Data Wrangle OpenStreetMaps Data

by Rica Enriquez, July 1, 2015

In this project, the OpenStreetMap data for Cambridge, United Kingdom is explored. Having lived there for a few years, I found it an intersting city. It has a good blend of history and modernity. The data was downloaded from https://mapzen.com/data/metro-extracts on July 1, 2015. It was prepared for MongoDB using "nanoproject_2.py" and the resulting JSON file was added into a local "udacity" database as the "cambridge" collection.

Section 0. References

- Udacity sample document
  - https://docs.google.com/document/d/1F0Vs14oNEs2idFJR3C_OPxwS6L0HPliOii-QpbmrMo4/pub
- MongoDB group values by multiple fields
  - http://stackoverflow.com/questions/22932364/mongodb-group-values-by-multiple-fields

Section 1. Problems Encountered in the Map

Removing Underutilized Labels

When importing the data, all labels were included at first. Many of these labels were only available for a small fraction of the documents. So, underutilized upper labels and lower lables were removed. If a subset of the dataset is used in the future, removal of labels may need to be done more judiciously. The "created" and "position" labels, along with their sublabels, are kept for the purposes of this exercise.

Only upper labels were used in at least 1000 documents were kept. This was found previously via MongoDB queries. Similarly, only lower labels that had at least 500 documents were kept. The cleaning of the data is done before the JSON file is created. The code for cleaning can be found in "nanoproject_2_prep.py."

The final structure of the collection is seen in the following dictionary, where the sublabels are defined as "None" or an array:

```
{"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"],
 "created": ["version", "changeset", "timestamp", "user", "uid"],
 "position": ["lat", "lon"]}
```

Capitalize Street Names

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

Remove Documents with Invalid Postcodes

All postcodes in Cambridge start with CB. Listing the postcodes in the collection, documents with a postcode of "SG8 5TF" is discovered and should not be included since it is for a place in Royston and the Stevenage postcode area. Since there are only two documents in the collection with this postcode, it is an error and not an approach to extend the collection the surrounding area. Additionally the postcode "CB1" is incomplete - there should be a second set of three characters. Documents with this postcode are not included.

Update Cities to Cambridge

Some cities were "cambridge" and not "Cambridge", overspecified to "Girton" or "South Cambridgeshire", or listed as "11". However, other information shows that each entry is in Cambridge. Therefore, the "city" is updated to "Cambridge". This was fixed using a mapping scheme and the function "update_entry" in "nanoproject_2_prep.py."

Pare Down Barrier, Entrance, Highway, Landuse, and Operator

Some entries for these sublabels were the same, but in a different format. They were updated to be more consistent. These errors were also fixed using a mapping scheme and the function "update_entry" in "nanoproject_2_prep.py."

Section 2. Overview of the Data

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

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

# Connect to database
client = MongoClient()
db = client["udacity"]
```

In [2]:
```
print "The size of 'cambridge_england.osm' is", os.stat("cambridge_england.osm").st_size / 1e6, "M
B."
print "The size of 'cambridge_england.osm.json' is", os.stat("cambridge_england.osm.json").st_size
/ 1e6, "MB."
N = db.cambridge.find().count()
print "There are", N, "documents in the set."
pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}}]
print "There are", len(list(db.cambridge.aggregate(pipeline))), "unique users."
print "There are", db.cambridge.find({"type": "node"}).count(), "nodes."
print "There are", db.cambridge.find({"type": "way"}).count(), "ways."
```

```
The size of 'cambridge_england.osm' is 61.828774 MB.
The size of 'cambridge_england.osm.json' is 70.833688 MB.
There are 300419 documents in the set.
There are 453 unique users.
There are 252027 nodes.
There are 48392 ways.
```

```
In [3]: pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 1}]
        print list(db.cambridge.aggregate(pipeline))[0]['_id'], "contributed the most to this collection w
        ith", \
            list(db.cambridge.aggregate(pipeline))[0]['count'], "documents."
        pipeline = [{"$group": {"_id": "$created.user", "count": {"$sum": 1}}},
                    {"$group": {"_id": "$count", "num_users": {"$sum": 1}}},
                    {"$sort": {"_id": 1}},
                    {"$limit": 1}]
        print list(db.cambridge.aggregate(pipeline))[0]['num_users'], "users contributed once."
        pipeline = [{"$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.aggregate(pipeline))[1][
            "_id"], "are the top two amenities."
```

```
smb1001 contributed the most to this collection with 79851 documents.
120 users contributed once.
university bicycle_parking are the top two amenities.
```

Section 3. Additional Ideas

The Cambridge, England OpenStreetMap dataset is full of information. However, it can be cumbersome to analyze. During the cleaning up stages, many labels and sublables were removed. It may be useful to move the information to an exisiting label rather than having it removed. For example, the information in "have_riverbank" and "trees" can be moved as sublabels of the "natural" label. By moving labels rather than not including them, prematurely removing useful information can be prevented.

Since all the labels are known, structuring the labels is possible, though tedious. One way to reduce the labels without excessive information removal is to lower the limit of documents in which the labels need to appear (e.g., from 1000 to 10). However, one must be careful in properly creating the new tree structure. Samples of the label information from the dataset would be useful in making appropriae sublabels and upper labels. After the labeling structure is created, it can be submitted as a proposal for future OpenStreetMap users to follow, prior to submission of new information.

Additonal Data Exploration Using MongoDB Queries

Some addresses have house names. It would be interesting to know if there's a certain postal code with the most and if there is an operator that is popular. Additionally, it would be interesting where the top amenities are located.

```
In [4]: # Find the top postcodes with housenames
        pipeline = [{"$match": {"address.housename": {"$exists": True}}},
                    {"$match": {"address.postcode": {"$ne": None}}},
                    {"$group": {"_id": "$address.postcode", "count": {"$sum": 1}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 5}]
        pprint.pprint(list(db.cambridge.aggregate(pipeline)))

        # Find the top operators with housenames
        pipeline = [{"$match": {"address.housename": {"$exists": True}}},
                    {"$match": {"operator": {"$ne": None}}},
                    {"$group": {"_id": "$operator", "count": {"$sum": 1}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 5}]
        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}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 10},
                    {"$group": {"_id": "$_id.amenity",
                                "breakdown": {"$push": {
                                    "postcode": "$_id.postcode",
                                    "count": "$count"},},
                                "count": { "$sum": "$count"}}},
                    {"$sort": {"count": -1}},
                    {"$limit": 3}]
        pprint.pprint(list(db.cambridge.aggregate(pipeline)))
```

```
[{u'_id': u'CB4 1HG', u'count': 17},
 {u'_id': u'CB3 0EY', u'count': 11},
 {u'_id': u'CB1 2LJ', u'count': 9},
 {u'_id': u'CB4 1HH', u'count': 8},
 {u'_id': u'CB1 2LG', u'count': 7}]
[{u'_id': u'University of Cambridge', u'count': 52},
 {u'_id': u'Clare College (University of Cambridge)', u'count': 6},
 {u'_id': u'Anglia Ruskin University', u'count': 4},
 {u'_id': u'Selwyn College (University of Cambridge)', u'count': 4},
 {u'_id': u'Riverside ECHG', u'count': 3}]
[{u'_id': u'restaurant',
  u'breakdown': [{u'count': 8, u'postcode': u'CB2 1DP'},
                 {u'count': 6, u'postcode': u'CB1 2AS'},
                 {u'count': 5, u'postcode': u'CB2 1AB'}],
  u'count': 19},
 {u'_id': u'fast_food',
  u'breakdown': [{u'count': 4, u'postcode': u'CB1 7AW'},
                 {u'count': 4, u'postcode': u'CB4 1JY'},
                 {u'count': 4, u'postcode': u'CB1 2AD'},
                 {u'count': 3, u'postcode': u'CB1 7AA'}],
  u'count': 15},
 {u'_id': u'cafe',
  u'breakdown': [{u'count': 5, u'postcode': u'CB1 3NF'},
                 {u'count': 3, u'postcode': u'CB2 1LA'}],
  u'count': 8}]
```