Analyzing the dublin_ireland.osm dataset

Udacity Data Wrangling Course

Anthony Munnelly

ABSTRACT

Udacity Data Science Nanodegree, May 2015 Cohort

Preparing the Dataset¶

I downloaded the Dublin dataset from https://mapzen.com/data/metro-extracts on July 22nd, 2015. Using the *tag_collector()* and *key_collector()* functions, I was able to identify three potential problem areas in the dataset that would have to be addressed before the data was uploaded to MongoDB.

Problem 1: Street Names¶

Cultural differences apply when it comes to a street name audit. The USA is very formalized in its street naming system, while Ireland and the UK are much less so. There is a wider selection of possible street endings to choose from and the elements that are difficult are singletons in the dataset.

Problem 2: Postcodes¶

While there was no real issue with street name endings, wrangling the postcodes proved a considerable task in this dataset. Some background: a new postcode system is being introduced in Ireland currently, but Dublin has had its own postcodes for many years.

The postcodes are important to Dublin's own sense of identity - all the even-numbered postcodes are to the south of the river Liffey, and all the odd-numbered postcodes to the north. Being a Northsider or Southsider is quite a point of pride in the city. However, there is no postcode key among the three million data points in the *dublin_ireland.osm* dataset. Not one.

There is, however, an *addr:city* key in the dataset, and this proved to be where most postcodes had been recorded. The recording was very haphazard though, and the *addr:city* key contained a mix of postcodes proper, districts of the city, the names of many of the satellite towns that surround Dublin, or else just the single word "Dublin" itself.

To address this problem, I wrote a function called *find_dublin_postcodes()* that took a link from the Irish Post Office (https://www.anpost.ie/AnPost/AnPostDM/ProductsAndServices/Publicity+Post/DublinDeliveryZones/Dublin+Delivery+Zones.htm)and used the *BeautifulSoup* module on data from that link to form an initial list of official Dublin post codes.

I then iterated that Post Office list against the values for *addr:city* to identify which districts or postcodes weren't on the offical list. I created a .json file of these outliers, read the .json back as a dict and used both dictionaries in another function, *fix_postcode()*, that identified the *addr:city* as a postcode, a Dublin district or a satellite town, and amended the *shape_element()* function appropriately.

Problem 3: Two Languages¶

The most intriguing feature of the dataset is the presence or absence of the Irish language. Although everybody in Ireland speaks English, the official first language of the state is Irish, or Gaelic. The *name:ga* key in the dataset means the value will be a placename's Gaelic translation, but it's also possible that the placename itself is in Gaelic. A Gaelic word has to be coded as Unicode which could have caused issues when writing to the .json file, but happily the codecs module took care of this.

Upload¶

It took four minutes to convert the XML to .json. From this .json file, 1,184,959 documents were created in a MongoDB dublin collection, itself part of a dublin database, using mongoimport:

```
mongoimport --db dublin -c dublin --file dublin_ireland.osm.json
```

Size of the files

dublin_ireland.osm	238.4 MB
dublin_ireland.osm.json	281.1 MB

The data having being uploaded, the next tasks were to fulfill the core requirements of the project rubric. All code is in python, taken from the *examining_dublin_ireland_osm.ipynb* program.

```
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017')
db = client.dublin
```

Number of Documents¶

```
db.dublin.find().count()
1184959
```

Number of Unique Users¶

```
distinct_users = db.dublin.distinct("created.user")
print len(distinct_users)

1032
```

Number of Nodes¶

```
db.dublin.find({"type":"node"}).count()

1009514
```

Number of Ways¶

```
db.dublin.find({"type":"way"}).count()

175413
```

Number of pubs¶

Leopold Bloom, the hero of James Joyce's novel *Ulysses*, reflects that it would a be a "pretty puzzle to cross ublin without passing a pub." Let's see if the OSM data would be any help to Mr Bloom.

```
import pprint

pub_pipeline = [{"$match":{"amenity":{"$exists":1}, "amenity":"pub",
             "amenity":"bar"}},
               {"$project":{"_id":False,"Name":"$name",
               "Type":"$amenity", "Location":"$address"}},
               {"$sort":{"Amenity":-1, "Name":1}}]


pubs = db.dublin.aggregate(pub_pipeline)

# Shortened to the first ten for convenience
for p in pubs['result'][:10]:
    pprint.pprint(p)

{u'Type': u'bar'}
{u'Type': u'bar'}
{u'Name': u'Alfie Byrnes', u'Type': u'bar'}
{u'Location': {u'county': u'Dublin', u'town': u'Dublin'},
 u'Name': u'Arlington Bar & Restaurant',
 u'Type': u'bar'}
{u'Name': u'Ballyfermot Sports and Social Club', u'Type': u'bar'}
{u'Name': u'Bar@Regency', u'Type': u'bar'}
{u'Name': u'Bar@Skylon', u'Type': u'bar'}
{u'Name': u'Bar@Tolka', u'Type': u'bar'}
{u'Location': {u'county': u'Dublin', u'town': u'Dublin'},
 u'Name': u'Berlin',
 u'Type': u'bar'}
{u'Name': u'C Central', u'Type': u'bar'}
```

As a reverse sort is applied in the aggregation query, we should see the pubs listed before the bars. There are no amenities tagged as "pubs" at all - everything is a bar or not at all. This, and the nature of the bars listed, points towards a younger demographic having compiled the data.

Number of Postcodes¶

As most of the wrangling had been done on postcodes, it's worthwhile to see how they worked out.

```
postcode_pipeline = [{"$match":{"address.postcode":{"$exists":1}}},
                     {"$group":{"_id":"$address.postcode",
                                "count":{"$sum":1}}},
                     {"$project":{"_id":"$_id",
                     "count":"$count"}},
                     {"$sort":{"count":-1}}]

db.dublin.aggregate(postcode_pipeline)
```

```
{u'ok': 1.0,
 u'result': [{u'_id': u'D6', u'count': 1013},
  {u'_id': u'D15', u'count': 990},
  {u'_id': u'D2', u'count': 589},
  {u'_id': u'D8', u'count': 448},
  {u'_id': u'D1', u'count': 370},
  {u'_id': u'D6W', u'count': 228},
  {u'_id': u'D7', u'count': 166},
  {u'_id': u'D12', u'count': 159},
  {u'_id': u'D11', u'count': 151},
  {u'_id': u'D3', u'count': 150},
  {u'_id': u'D9', u'count': 105},
  {u'_id': u'D4', u'count': 50},
  {u'_id': u'D5', u'count': 44},
  {u'_id': u'D14', u'count': 20},
  {u'_id': u'D16', u'count': 19},
  {u'_id': u'D20', u'count': 17},
  {u'_id': u'D13', u'count': 17},
  {u'_id': u'D10', u'count': 15},
  {u'_id': u'D18', u'count': 10},
  {u'_id': u'D17', u'count': 7},
  {u'_id': u'D24', u'count': 3}]}
```

Auditing the Users¶

An .osm map is only as good as those who upload the data. Let's look at the users of dublin_ireland.osm.

```
users = db.dublin.aggregate([{"$group":{"_id":"$created.user",
                                        "posts":{"$sum":1}}},
                             {"$project":{"_id":"$_id", "posts":"$posts",
                                          "percentage":
                                                 {"$divide":["$posts",
                                                 1184959]}}},
                                  {"$sort":{"posts":-1}}])

for i in users['result'][:10]: # That is, the top ten users
    print "{} made {:,} posts, {:.3}% of the total."
            .format(i['_id'],
                    i['posts'],
                    i['percentage']*100)
```

```
Nick Burrett made 229,079 posts, 19.3% of the total.
mackerski made 181,682 posts, 15.3% of the total.
Dafo43 made 150,454 posts, 12.7% of the total.
brianh made 133,700 posts, 11.3% of the total.
Conormap made 55,202 posts, 4.66% of the total.
Ignobilis made 54,606 posts, 4.61% of the total.
VictorIE made 40,891 posts, 3.45% of the total.
wigs made 21,137 posts, 1.78% of the total.
Blazejos made 19,935 posts, 1.68% of the total.
ManAboutCouch made 19,000 posts, 1.6% of the total.
```

Nick Burrett, *mackerski*, *Dafo43* and *brianh* have contributed 52% of the dataset between the four of them. The remaining 48% of the data was posted by 1,028 other users.

The Use of Gaelic¶

One of the most interesting aspects of the dataset is the use of Irish in some of the placenames. Let's look at how many translations exist in the dataset.

```
gaelic = db.dublin.aggregate([{"$match":{"name:ga":{"$exists":1}}}])
print "There are {:,} placenames translated into Irish in the
dataset.".format(len(gaelic['result']))
```

```
There are 11,835 placenames translated into Irish in the dataset.
```

```
gaelic['result'][0]
```

```
{u'_id': ObjectId('55c8feb9450fbe23a5b0a25e'),
 u'created': {u'changeset': u'27519679',
  u'timestamp': u'2014-12-17T01:04:53Z',
  u'uid': u'2008037',
  u'user': u'VictorIE',
  u'version': u'13'},
 u'id': u'661291',
 u'name': u'Sutton',
 u'name:en': u'Sutton',
 u'name:ga': u'Cill Fhionntain',
 u'operator': u'Irish Rail',
 u'pos': [53.3920249, -6.1160842],
 u'railway': u'station',
 u'type': u'node',
 u'visible': None}
```

What sort of data is being recorded here? Are the Gaelic names in the *dublin_ireland.osm* dataset there because of the work of Gaelic enthusiasts? Let's examine the keys.

```
translations = {}
for item in gaelic['result']:
    for key, value in item.iteritems():
        if key not in translations:
            translations[key] = 1
        else:
            translations[key] += 1

counter = translations.values()
counter = sorted(counter, reverse=True)
repeating_keys = ["created", "visible", "type", "id", "name:ga",
"_id", "name", "node_refs"]
for c in counter[:15]:
    for key, value in translations.iteritems():
        if c == value and key not in repeating_keys:
            print key, value
```

```
highway 10839
maxspeed 9698
name:en 5229
ref 2419
oneway 1737
cycleway 646
lanes 499
```

The examination of the keys proves to be a disappointment. Evidence that the Gaelic was being either recorded or translated on the spot would indicate a community of Irish speakers who are active online and in the Open Street Map community. However, the choice of location does not fit with this idea. The occurrence of tags like *highway*, *maxspeed* and *oneway* suggests that, rather than working at the language, the *dublin_ireland.osm* users are simply writing down what they see in front of them. In Ireland, all street signs are bi-lingual.

Improving the Database¶

The *dublin_ireland.osm* dataset is not very good. The nomenclature is inconsistent and the geographic sourcing of the data is haphazard. We can see this level of randomness in the compilation of the bar list, where very few are given with an address, and some aren't even named. The addressing methodology is inconsistent at best. However, the users are much more engaged with pinpointing locations using latitude and longitude.

```
db.dublin.find({"address":{"$exists":1}}).count()
12425

db.dublin.find({"pos":{"$exists":1}}).count()
1009525
```

Is it possible, then, to use this GPS to search another dataset, and fill out missing fields that way? For instance, Dublin bus records the GPS for all of the bus stops on its routes (http://dublinked.com/datastore/datasets/dataset-254.php). It should be possible to use this to fill in some missing data. It's not much, but it's a start.