P3_Project_Report

February 24, 2016

1 Data Wrangling with MongoDB - OpenStreetMap Project

1.1 Map Area: Cheshire, England

1.1.1 1. Problems Encountered With the Data

Using a sample of the data for an initial audit, I noticed three main problems with the data: 1. Abbreviations in the street names, for example 'Rd', 'St' etc. 2. Incorrect street names. A large number of unusual, possibly incorrect street names. 3. Incorrect or incorrectly formatted postcodes.

I will discuss each of the above issues in detail below. Please note that my analysis of the sample data and the conversion of the complete file from xml into json format can be found in the attached document entitled P3_Project_Initial_Audit.

I imported the complete dataset into Mongodb in order to complete the following analysis. Please see attached document P3_Project_Wrangling_Stage for details.

1. Abbreviations in Street Names I began by converting the frequently occuring street name abbreviations I had seen in the sample data. Following this, querying the data and excluding all street names with the correct street type I converted the remaining street names. These included street types that were given in lower case or appeared to have been spelled incorrectly. In total there were 17 conversions:

```
In [ ]: mapping = { "St": "Street",
                    "Rd":"Road",
                    "Sqaure":"Square",
                    "Steet":"Street",
                    "street":"Street",
                    "road":"Road",
                    "crecent":"Crescent",
                    "sq":"Square",
                    "cheadle":"Cheadle",
                    "close":"Close",
                    "crescent":"Crescent",
                    "drive":"Drive",
                    "green":"Green",
                    "grove":"Grove",
                    "level":"Level",
                    "lightoaks":"Lightoaks",
                    }
```

2. Incorrect street names After correcting for the street types mentioned above there were a number of street names that still looked incorrect. Namely:

```
In [ ]: need_to_clean = ['Level', 'lon','1DB','1LE','A','A5027']
```

Looking at the complete address entries for each of these street types showed that in the case of '1DB' and '1LE' the postcode had been entered in the street name field. In each of these cases I moved the values into the correct field.

With 'A5027' the street name and street number (since all major roads have a street name and street number) was given. I deleted the street number in order to avoid confusion.

With 'A' the complete street name was 'Harding A' with postcode CH63 3H. After checking to see if the street name 'Harding' exists in the postcode CH63, I found that the correct name should be 'Harding Avenue' which I amended accordingly in the dataset.

'Level' could have been correct, I did not have any further information, such as a postcode in order to verify this.

Finally the full street name for the 'lon' street type was 'Pen-y-lon'. More closely examining the map of the area I selected showed that it included a small part of Wales in the data, therefore some of the data is in Welsh.

The main issue with the data (which still remains) is that there are a number of unusual steet names (as is common in England) and therefore it is difficult to identify which name could be incorrect. However all abbreviations and obvious mistakes were corrected.

3. Incorrect or Incorrectly Formatted Postcodes I checked for postcodes that did not take the following regex format:

```
In [ ]: postcode_type_re = re.compile(r'[A-Z]{1,2}\d{1,2}\s\d[A-Z]{2}')
        short_postcode_type_re = re.compile(r'[A-Z]{1,2}\d{1,2}$')
```

While complete postcodes have two parts to them, I also decided to take into account the postcodes with just the first part given, since this is still correct (although incomplete).

The result was a long list of incorrect postcodes. One in partcular contained street name data instead of postcode data, I therefore moved this entry into the street name field.

The two common issues with the resulting data were that firstly the letters in the postcode were in lowercase, which I changed to uppercase, and secondly the postcode was not split into two different parts, which I also corrected.

```
In [ ]: no_space_re = re.compile(r'[A-Z]{1,2}\d{1,3}[A-Z]{2}')

        clean_these = set()

        postcode_data = db.cheshire_map.find({"address.postcode":{"$exists":1}})

        for item in postcode_data:
            m = postcode_type_re.search(item['address']['postcode'])
            n = short_postcode_type_re.search(item['address']['postcode'])
            if not m and not n:

                pprint.pprint(item['address']['postcode'])
                clean_these.add(item['address']['postcode'])

        for post_code in clean_these:
            db.cheshire_map.update_many({"address.postcode" : {"$exists" : 1},
                                         "address.postcode":post_code},
                { "$set": { "address.postcode":post_code.replace(post_code, post_code.upper()) }})

            pc = no_space_re.search(post_code)
            if pc:

                db.cheshire_map.update_many({"address.postcode" : {"$exists" : 1},
                                             "address.postcode":post_code},
```

```
                { "$set":
                 {"address.postcode":post_code.replace(post_code, post_code[:-3]+" "+post_code[-3:]) }}
```

I was left with the following list:

```
In [ ]: u'LL'
        u'CH1 34JS'
        u'SK237DG'
        u'SK23 123'
        u'WA42HF'
        u'CH7 1B'
        u'ST101JZ'
        u'ST101JZ'
        u'ST101RX'
        u'CW84DB'
        u''
        u'CH63 3H'
        u'CH16AZ'
        u'M22-9QY'
        u'ST9 9 HH'
        u'CW& 2AU'
        u'CW& 2AN'
        u'CW10 ODD'
        u'WA8 3'
        u'SK23 123'
        u'WA42HF'
```

I included a space in the postcodes that had been converted to uppercase, and removed the '&' from the two postcodes that had included it, replacing it with a 7 which is what the correct postcode should have been. I also removed the '-' from one of the postcodes, corrected those with incorrecting spacing and in the case of the 'SK23 123' postcode, I shortened it to just 'SK23' which falls in the Stockport area of the map. I was left with the following, which I left as they were.

```
In [ ]: u'LL'
        u'CH7 1B'
        u''
        u'CH63 3H'
        u'WA8 3'
```

1.1.2 2. Data Analysis

File size:

```
In [112]: import os
          print 'cheshire_england.osm'
          os.path.getsize('C:\Users\owner\Desktop\udacity\P3 Wrangling\P3 Project\cheshire_england.osm')

cheshire_england.osm

Out[112]: 776004005L

In [113]: print 'cheshire_england.osm.json'
          os.path.getsize('C:\Users\owner\Desktop\udacity\P3 Wrangling\P3 Project\cheshire_england.osm.

cheshire_england.osm.json

Out[113]: 890613419L
```

In [3]:
```
import pprint

from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.examples.cheshire_map

# Number of documents

db.find().count()
```

Out[3]: 4037005

In [4]:
```
# Number of nodes

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

Out[4]: 3497482

In [5]:
```
# Number of ways

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

Out[5]: 539474

In [88]:
```
# Number of unique users and names of top 10 contributors

#len(db.find({'created.user':{"£ne": None}}).distinct('created.user'))
total = db.aggregate([
        {"$group" : {"_id" : "$created.user", "count": {"$sum":1}}},
        {"$group" : {"_id" : "Total", "ttl" : {"$sum" : "$count"}}}])

for thing in total:
    total_contrib = thing['ttl']

users = db.aggregate([
        {"$group" : {"_id" : "$created.user", "count": {"$sum":1}}},
        {"$project" : {"_id" : "$_id", "count" : "$count", "percentage" :
                        {"$multiply" : [{"$divide" : ["$count",total_contrib]},100]}}},
        {"$sort" : {"count": -1}}])
counting = 0
number = 0

for thing in users:
    if counting < 10:
        print thing
        counting +=1
    number += 1

print 'Number of unique users:',number
```

```
{u'count': 1693381, u'percentage': 41.94646774031739, u'_id': u'daviesp12'}
{u'count': 324502, u'percentage': 8.03818672505979, u'_id': u'RichardB'}
{u'count': 203186, u'percentage': 5.033087647897389, u'_id': u'Former OSM contributor'}
{u'count': 179879, u'percentage': 4.4557537085042, u'_id': u'Mauls'}
{u'count': 167691, u'percentage': 4.153846725480895, u'_id': u'kjnpbr'}
```

```
{u'count': 94626, u'percentage': 2.3439653901840596, u'_id': u'dudone'}
{u'count': 68240, u'percentage': 1.6903620381941562, u'_id': u'mikh43'}
{u'count': 65953, u'percentage': 1.6337111299094254, u'_id': u'Dyserth'}
{u'count': 63343, u'percentage': 1.5690592406994788, u'_id': u'richardwest'}
{u'count': 44370, u'percentage': 1.09908211656909, u'_id': u'Chris Morley'}
Number of unique users: 1667
```

As above, the majority of the contributions to the Cheshire map have come from one user, 'daviesp12' who accounted for almost 42% of contributions.

Further Data Analysis: Amenities Firstly I will look at the top 10 listed amenities in the data set.

```
In [90]: amenities = db.aggregate([{"$match":{"amenity":{"$exists":1}}},
                                   {"$group":{"_id":"$amenity","count":{"$sum":1}}},
                                   {"$sort":{"count":-1}}, {"$limit":10}])
         for thing in amenities:
             print thing
```

```
{u'count': 4732, u'_id': u'parking'}
{u'count': 1559, u'_id': u'pub'}
{u'count': 1468, u'_id': u'post_box'}
{u'count': 1081, u'_id': u'school'}
{u'count': 1049, u'_id': u'place_of_worship'}
{u'count': 511, u'_id': u'fast_food'}
{u'count': 506, u'_id': u'bench'}
{u'count': 443, u'_id': u'telephone'}
{u'count': 354, u'_id': u'restaurant'}
{u'count': 291, u'_id': u'cafe'}
```

Pubs Aside from parking spaces, the most commonly appearing amenity is a pub. The 5 most popular pub names are as follows:

```
In [116]: pubs = db.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity" : "pub",
                                           "name" : {"$ne" : None}}},
                               {"$group":{"_id":"$name","count":{"$sum":1}}},
                               {"$sort":{"count":-1}}, {"$limit":5}])
          for thing in pubs:
              print thing
```

```
{u'count': 11, u'_id': u'The Red Lion'}
{u'count': 10, u'_id': u'Red Lion'}
{u'count': 10, u'_id': u'The Royal Oak'}
{u'count': 9, u'_id': u'The Crown'}
{u'count': 9, u'_id': u'The White Lion'}
```

Whilst there are 1559 pubs in the map area, the most common pub name, 'The Red Lion'/'Red Lion' only appears 21 times.

Below is a list of 10 original (possibly obscure!) pub names.

```
In [115]: pubs = db.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity" : "pub",
                                           "name" : {"$ne" : None}}},
                               {"$group":{"_id":"$name","count":{"$sum":1}}},
                               {"$sort":{"count":1}}])
          count = 0
          for thing in pubs:
```

```
            if thing['count'] == 1 and count < 10:
                print thing
                count += 1
```

```
{u'count': 1, u'_id': u'Rising Sun PH'}
{u'count': 1, u'_id': u'The Pheasant'}
{u'count': 1, u'_id': u'Trentham Village Toby Carvery'}
{u'count': 1, u'_id': u'The Mersey Clipper'}
{u'count': 1, u'_id': u'Oak PH'}
{u'count': 1, u'_id': u'Pool Dole'}
{u'count': 1, u'_id': u'The Nant Inn'}
{u'count': 1, u'_id': u'Vale Royal Abbey Arms'}
{u'count': 1, u'_id': u'The Golden Lion (closed)'}
{u'count': 1, u'_id': u'The Calveley Arms'}
```

Place of Worship Looking at places of worship, the 3 most popular religions are as follows:

```
In [109]: religions = db.aggregate([{"$match":{"amenity":{"$exists":1}, "amenity" : "place_of_worship",
                                                "religion" :{"$ne" : None}}},
                                    {"$group":{"_id":"$religion","count":{"$sum":1}}},
                                    {"$sort":{"count":-1}}, {"$limit":3}])
          for thing in religions:
              print thing
```

```
{u'count': 911, u'_id': u'christian'}
{u'count': 6, u'_id': u'muslim'}
{u'count': 5, u'_id': u'jewish'}
```

1.1.3 3. Additional Ideas

As mentioned in section 1, the street types for the data set needs further cleaning. There are a number of different street types in the data set, which need to be audited more thoroughly to ensure all are correct. Further, all major A and B roads have road numbers which should be included in the maps.

One further potential improvement could be in the labelling of amenities in the data. For example there is an amenity labelled 'yes', and also one labelling 'bicyckle_parking'.

There are also a vast number of amentities with no lables, its clear that the map is incomplete.

1.1.4 Conclusion

The Cheshire openstreetmap data has been partially cleaned in the completion of this report. However there is much more work to be done with the dataset, not only correcting further errors in the data, but also completing missing data in the map.

One solution for verifying the street names would require a person with local knowledge to provide/update this information. This solution would ensure that the quirkier street types could be verified. However this could introduce more human error to the map, and to find a person/people with such extensive knowledge of this region which covers England and a part of Wales may prove difficult. The data could also be cross-referenced with other street name/mapping resources, if there are any that would be willing to easily share their resources. The UK Post Office provides postcode information which could be used to check postcodes and street names.