Udacity Data Analyst Nanodegree
Data Wrangling Project

Richard Dean

Overview

This project required the student to obtain OpenStreetMap data, perform some initial data cleansing, load it into MongoDB, and run some basic data analyses. This document covers the steps I took in completing these tasks.

Data selection, initial audits, early statistics and loading into MongoDB

File source:	https://s3.amazonaws.com/metro-extracts.mapzen.com/liverpool_england.osm.bz2
File size (uncompressed):	272MB
Sample size:	5.5MB
Processed JSON:	310MB
Records loaded:	1436842

The OpenStreetMap data was sourced from https://mapzen.com/data/metro-extracts. I specifically chose Liverpool (UK), as it was my home for 20 years, thus any insights gained from the data analysis would be particularly interesting to me. A sample of the full dataset was created, containing 1/30 elements from the original, to ease the process of the initial audits.

Street Names

For the purposes of audit, a street name is defined as the final word in the addr:street <tag> key. My initial list of valid street names was simply the ones I could think of in a few minutes - I used this list against the sample file to find any unknowns. The sample file contained 13 additional names, all of which were valid UK-style names.

Postcodes

The UK postcodes are particularly detailed, so the regular expression to match them is also complex:
^(GIR ?0AA|[A-PR-UWYZ]([0-9]{1,2}|([A-HK-Y][0-9]([0-9ABEHMNPRV-Y])?)|[0-9][A-HJKPS-UW]) ?[0-9][ABD-HJLNP-UW-Z]{2})$
Every one of the postcodes in the sample file matched the regular expression.

The accuracy of the above audits suggests an earlier data cleaning operation has been performed on this data. I chose to reassess this data once the complete file had been imported into MongoDB.

Analysis - general subjects and a few specifics

Total number of records

Query	db.liverpool.count()
Result	1436842

Occurences of ‘type’ elements

Query	[{'$match':{'$or':[{'type':"way"},{'type':"node"}]}}, {'$group': {'_id': '$type', 'count': {'$sum':1}}}, {"$limit": 2}]	
Result	way	242005
	node	1194827

Number of unique contributors

Query	len(db.liverpool.distinct(‘created.user’))
Result	693

Top 5 contributors as percentage of contributions

Query	totalUserPosts = db.liverpool.count({"created.user": {"$exists": True}}) [{"$match": {"created.user": { "$exists": True }}}, {"$group":{"_id":"$created.user","count":{"$sum":1}}}, {"$project": {"count":1,"percentage":{"$multiply":[{"$divide":[100,totalUserPosts]},"$count"]} }},{"$sort" : {"count": -1}}, {"$limit": 5}]		
Result	**Username**	**Edits**	**Percentage**
	daviesp12	1066921	74.25
	jrdx	36075	2.51
	UniEagle	34564	2.41
	F1rst_Timer	22301	1.55
	duxxa	20953	1.46

Postcodes: Re-assessment

Query	*postcode_re* is a compiled regular expression, defined as the postcode matching regex shown above. [{'$match': {'address.postcode':{'$exists':True}, }}, {'$match': {'address.postcode':{'$not': postcode_re}}}, {'$project': {'postcode':'$address.postcode'}}, {'$group': {'_id': '$postcode', 'count':{'$sum':1}}}, {'$project': {'postcode':'$_id', 'count':'$count'}}, {'$limit': 20}]	
Result	**Postcode**	**Count**
	L35 9JY.	1
	L1 4LN,L1 3DN	3
	L1	1
	CH43	1
	L17	1
	CH63	1
	L18	1

The results above show that in the entire dataset (over 14k postcodes), there are only 7 which need to be corrected, and only one could possibly benefit from an automated cleansing process (removing the full stop from L35 9JY.) This was added to the import script, to demonstrate the use of a cleansing process.

Further analysis to be considered

The first piece of extended analysis that I considered was the distribution of edits over time. This would clearly show if the edits by **daviesp12** were made in one update or many, and also if they were was any pattern.

Query	[{"$match": {"created.user": { "$exists": True }}}, {'$project': {'username': '$created.user', 'year': { '$year': "$created.timestamp" }, 'month': { '$month': "$created.timestamp" } }}, {'$group':{'_id': {'year':'$year', 'month':'$month', 'username':'$username'}, 'count':{'$sum':1}}},

	{'$sort': {'count':-1}}, {'$limit': 10}]			
Result	**Year**	**Month**	**Username**	**Edits**
	2013	8	daviesp12	184933
	2013	12	daviesp12	183008
	2015	1	daviesp12	170871
	2013	9	daviesp12	108725
	2014	6	daviesp12	96320
	2014	5	daviesp12	60663
	2013	11	daviesp12	56609
	2015	2	daviesp12	49488
	2013	10	daviesp12	43747
	2014	1	daviesp12	42738

These results show no clear pattern, and cover several years - daviesp12 is clearly a regular updater of OpenStreetMaps. It would be interesting to look into neighbouring areas to find if he also updated there - or perhaps across the entire country.

Ideas for Improvement

Several potential methods of improving the data quality have occurred to me during this project:

Enforcing the *name* field for particular *amenity* types. Certain classes of amenity should (it would seem) should always have a name - for instance pubs, schools, restaurants, etc. By making the *name* compulsory, the general data quality could be improved. In the same way, postcodes for all addresses could be enforced.

In order to encourage users to correct existing data (like the 15 pubs with no name), I would propose ‘gamifying’ fixes. Various automated methods could be used to identify potential data issues (such as missing postcodes or amenity names), and list these problem items. Users could then be ranked on fixes they make to the data. Naturally, some double-checking method would be required to validate the submitted corrections to ensure that genuine data had been supplied.

One particular bulk improvement would be to validate address data against the UK PAF - the official Postal Address File listing every valid postal address in the country. This data is supplemented by the Ordnance Survey coordinate file, which maps postcodes to location - another useful validation check to ensure that the address/postcode information was in the correct geographic location. Unfortunately, the PAF is a commercial product (exact pricing data is not know), and thus could prove expensive for the

OpenStreetmaps project - or any user who chose to use it in a bot. The Ordnance Survey data, however, has been made available free-of-charge, so the postcode-location mapping could potentially be performed.

Conclusion

The Liverpool area of the OpenStreetMap data seems remarkably clean, based on the two simple checks I performed. The edits are dominated by a single user who has made nearly 75% of all current edits. Looking at the data, I found several other areas that could be checked or investigated - website URLs, amenity names (there are 15 pubs with no name!) and more detailed user analysis.

Appendix

Valid street Names
Street, Avenue, Boulevard, Drive, Court, Place, Square, Lane, Road, Close, Terrace, Grove, Crescent, Way, Mews, View

Additional names found in sample
Croft, Shorefields, North, Parade, Rise, Park, Dell, Beechway, Flaxhill, Summerwood, Hill, East, Pipers