PROJECT: Wrangle OpenStreetMap Data

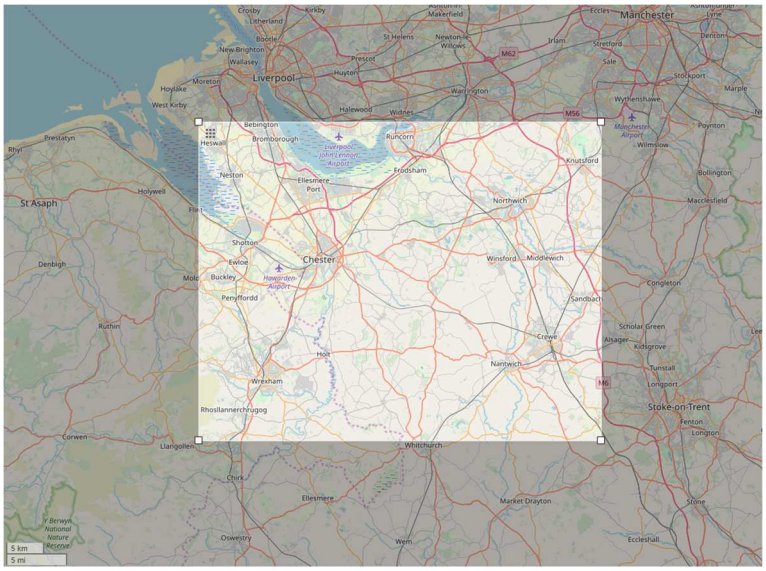
Map Area

The area to be analysed is the area encompassing the Borough of Cheshire West and Chester [https://www.openstreetmap.org/relation/153488]:

The following bounding box was selected to enclose the chosen area and the data was downloaded with the Overpass API.

- Latitude 52.9759 to 53.3530
- Longitude -3.1284 to -2.3346

[https://overpass-api.de/api/map?bbox=-3.1284,52.9759,-2.3346,53.3530]:



Datafile

Using the `audit_tags()` function the downloaded datafile was found to contain the following elements:

```
bounds ......... 1
member ..... 72718
meta ........... 1
nd ....... 2751965
node ..... 2039034
note ........... 1
osm ............ 1
relation .... 4615
tag ....... 845209
way ....... 323391
```

Problematic Keys

An initial investigation of the extracted node tags and way tags using the `audit_tags()` function revealed several items requiring attention:

- Keys containing '`fixme`'.
- '`todo`' keys.
- Keys beginning with '`not`'.
- Keys ending in '`_1`'.
- Keys containing capital letters.

To address these issues we will do the following:

- Keys containing '`fixme`' will be ignored.
- '`todo`' keys will be ignored.
- Keys beginning with '`not`' will be ignored. This includes notes but they will not be needed anyway.
- The trailing '_#' will be removed from all keys with that suffix, where # is any numeric character.
- While the convention is to use lower case keys, there are several valid keys which contain capital letters. For example, '`CEMT`' is the Classification of European Inland Waterways [https://wiki.openstreetmap.org/wiki/Key:CEMT]. We will therefore allow capital letters.

The node tags and way tags contain key-value pairs. These will need to be converted to key-value-type triplets, as per the supplied database schema. One solution used in the course was to split any tag keys containing a colon (`:`) into two parts, for example `key='aaa:bbb'` and `value='ccc'` becomes `key='bbb'`, `value='ccc'` and `type='aaa'`. We will use the same approach here. Keys that do not contain a colon will be assigned the type '`regular`' in

order to conform to the database schema. It is also noted that some keys contain multiple colons. In these instances, the first colon will be used to split the key, resulting in keys containing any remaining colons.

Problematic Values

Some of the more common tags were reviewed to decide what cleaning, if any, needs to be performed.

`'addr:street'` appears 1165 times as a node tag, and 49004 times as a way tag. Auditing with the `audit_street_types()` function revealed several items requiring attention:

- Problematic/unwanted characters.
- Abbreviated street names.
- Inconsistent use of capital letters, for example '`Lane`' and '`lane`'.
- Inconsistent use of hyphens in Welsh names, for example '`Pen-Y-Maes`' and '`Ael Y Bryn`'.

To address these issues we will do the following:

- Problematic/unwanted characters will be removed.
- Abbreviated street names will be replaced with the correct full name.
- Each word will modified to start with a capital letter.
- Welsh names containing 'Y', '`Yr`' and '`Yn`' will be hyphenated, for example '`Ael Y Bryn`' becomes '`Ael-Y-Bryn`'.

`'name'` appears 7991 times as a node tag, and 41265 times as a way tag. In addition to the problem areas described above, it was also noted that names appear in several languages, for example:

```
<tag k="name" v="Tuttle Street" />        (English)
<tag k="name:cy" v="Stryt Y Twtil" />     (Welsh)
```

We will address this issue as follows:

We will replace the 2-character language code with the full language name, in accordance with ISO-639.1 [https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes].
We will treat any such keys according to the following mapping:

```
key="name:cy",                  ⇨     key="name",
value="Stryt Y Twtil"                 value="Stryt Y Twtil",
                                      type="Welsh"
```

Multiple Values

It was noticed that several tags contain multiple values separated by a semi-colon (`;`), for example:

```
k='cuisine',
v='International;Burger;Coffee_Shop;Sandwich;Fish_And_Chips'
```

We will address this issue by splitting up the multi-value tag into several single-value tags. The example above then becomes:

```
k='cuisine', v='International'
k='cuisine', v='Burger'
k='cuisine', v='Coffee_Shop'
k='cuisine', v='Sandwich'
k='cuisine', v='Fish_And_Chips'
```

Data Overview

A smaller sample datafile was created from the main datafile using the `create_sample()` function with a seed of `k=50`. This was used to test the data cleaning functions.

The main datafile was processed using the `process_map()` function to produce several `.csv` files containing the extracted and cleaned `xml` data.

The `.csv` files were imported into a SQLITE Database using the following command:

```
sqlite3.exe < create_database.sql
```

The resulting database contained the following:

```
TABLE NAME    NO OF ENTRIES
----------    -------------
Nodes               2039034
nodes_tags           121859
ways                 323391
ways_tags            700173
ways_nodes          2751965
```

Users

The total number of unique users who contributed to the selected area of the map:

```
SELECT COUNT(*) FROM (SELECT uid FROM nodes UNION SELECT uid FROM ways);
```

```
1212
```

The top 10 contributing users, with their contributions as both count and percentage of total:

```
SELECT user, ROUND((100.0*cnt/(SELECT COUNT(*) AS total FROM (SELECT user
FROM nodes UNION ALL SELECT user FROM ways))),2) FROM (SELECT user,
COUNT(*) AS cnt FROM (SELECT user FROM nodes UNION ALL SELECT user FROM
ways) GROUP BY user ORDER BY cnt DESC LIMIT 10);
```

```
daviesp12 .............. 1093258 ... 46.28%
Dyserth ................ 124589 .... 5.27%
kjnpbr ................. 100390 .... 4.25%
Nathan2006 ............. 92797 .... 3.93%
RichardB ............... 79106 .... 3.35%
blackadder ............. 75965 .... 3.22%
Former OSM contributor ... 60557 .... 2.56%
James Derrick ........... 57262 .... 2.42%
mikh43 ................. 52388 .... 2.22%
Colin Smale ............. 41701 .... 1.77%
```

The number of users with a single contribution:

```
SELECT COUNT(*) FROM (SELECT user, COUNT(*) AS cnt FROM (SELECT user FROM
nodes UNION ALL SELECT user FROM ways) GROUP BY user HAVING cnt=1);
```

```
196
```

Multi-language names

Selected entries with names entered in both English and Welsh:

```
SELECT DISTINCT en.value AS English, cy.value AS Welsh FROM (SELECT
id,value FROM ways_tags WHERE key='name' AND type='English') AS en JOIN
(SELECT id,value FROM ways_tags WHERE key='name' AND type='Welsh') AS cy
ON en.id=cy.id ORDER BY RANDOM() LIMIT 10;
```

```
Station Road ............. Ffordd-Yr-Orsaf
Forest Walk .............. Ffordd-Y-Goedwig
Beech Tree Road .......... Ffordd Ffawydden
River Dee ................ Afon Dyfrdwy
Albert Grove ............. Gelli Albert
School Road .............. Ffordd-Yr-Ysgol
Browns Lane .............. Lon Brown
Chapel Street ............ Stryd-Y-Capel
Gorse Close .............. Clos Eithin
Blackbrook Drive ......... Lon Nant Ddu
```

Amenities

There appears to be an inconsistency in tag placement. For example, the top 10 amenities found in `nodes_tags`:

```
SELECT value, COUNT(*) AS cnt FROM nodes_tags WHERE key='amenity' GROUP BY
value ORDER BY cnt DESC LIMIT 10;
```

```
      Post_Box ......... 630
      Pub .............. 439
      Parking .......... 261
      Bench ............ 227
      Place_Of_Worship . 161
      Telephone ........ 154
      Grit_Bin ......... 149
      Restaurant ....... 145
      Fast_Food ........ 143
      Cafe ............. 115
```

The top 10 amenities found in `ways_tags`:

```
SELECT value, COUNT(*) AS cnt FROM ways_tags WHERE key='amenity' GROUP BY
value ORDER BY cnt DESC LIMIT 10;
```

```
      Parking .......... 2791
      School ............ 424
      Place_Of_Worship .. 306
      Pub ............... 236
      Grave_Yard ........ 111
      Fast_Food ......... 105
      Restaurant ......... 75
      Cafe ............... 51
      Fuel ............... 51
      Community_Centre ... 38
```

We will address this issue by creating a database view combining the contents of both the `nodes_tags` table and the `ways_tags` table:

```
CREATE VIEW all_tags AS SELECT * FROM nodes_tags UNION ALL SELECT * FROM
ways_tags;
COMMIT;
```

We will use this view in subsequent queries. The total number of entries in the view:

```
SELECT COUNT(*) FROM all_tags;
```

```
      822032
```

The top 10 amenities found in `all_tags`:

```
SELECT value, COUNT(*) AS cnt FROM all_tags WHERE key='amenity' GROUP BY
value ORDER BY cnt DESC LIMIT 10;
```

```
Parking .......... 3052
Pub ............... 675
Post_Box .......... 630
Place_Of_Worship .. 467
School ............ 450
Fast_Food ......... 248
Bench ............. 232
Restaurant ........ 220
Cafe .............. 166
Telephone ......... 155
```

Pubs

The most popular pub names in our selected area:

```
SELECT value AS pubname, COUNT(*) AS cnt FROM all_tags WHERE key='name'
AND id IN (SELECT id FROM all_tags WHERE key='amenity' AND value='Pub')
GROUP BY pubname ORDER BY cnt DESC LIMIT 10;
```

```
The Nags Head ......... 6
The Red Lion .......... 5
The Travellers Rest ... 5
The White Lion ........ 5
Red Lion Public House . 4
The Bridge Inn ........ 4
The Railway Inn ....... 4
The Royal Oak ......... 4
The Swan .............. 4
Cross Keys ............ 3
```

At first glance, the fact that `Red Lion` appears as both `The Red Lion` and `Red Lion Public House`, hints at inconsistency errors in data entry. However, a quick internet search for pubs named Red Lion in Cheshire showed several with different versions of the name. For example:

```
The Red Lion - [http://www.redlionlowerwithington.co.uk]
Red Lion Inn - [https://www.redlionlittlebudworth.com]
```

We will look at a few of the names in the list above to see what other configurations we have.

Nags Head

```
SELECT value AS pubname, COUNT(*) AS cnt FROM all_tags WHERE key='name'
AND pubname LIKE '%Nags Head%' AND id IN (SELECT id FROM all_tags WHERE
key='amenity' AND value='Pub') GROUP BY pubname ORDER BY cnt DESC;
```

```
      The Nags Head . 6
      Nags Head ..... 3
```

Red Lion

```
SELECT value AS pubname, COUNT(*) AS cnt FROM all_tags WHERE key='name'
AND pubname LIKE '%Red Lion%' AND id IN (SELECT id FROM all_tags WHERE
key='amenity' AND value='Pub') GROUP BY pubname ORDER BY cnt DESC;
```

```
      The Red Lion .......... 5
      Red Lion Public House . 4
      Red Lion .............. 3
      The Red Lion Inn ...... 2
      Old Red Lion .......... 1
      Red Lion Hotel ........ 1
      The Olde Red Lion ..... 1
```

White Lion

```
SELECT value AS pubname, COUNT(*) AS cnt FROM all_tags WHERE key='name'
AND pubname LIKE '%White Lion%' AND id IN (SELECT id FROM all_tags WHERE
key='amenity' AND value='Pub') GROUP BY pubname ORDER BY cnt DESC;
```

```
      The White Lion . 5
      White Lion ..... 3
      White Lion Inn . 1
```

Cuisine

The top 10 types of cuisine available in our selected area:

```
SELECT value, COUNT(*) AS cnt FROM all_tags WHERE key='cuisine' GROUP BY
value ORDER BY cnt DESC LIMIT 10;
```

```
      Fish_And_Chips . 48
      Chinese ........ 45
      Indian ......... 45
      Coffee_Shop .... 31
      Pizza .......... 31
      Sandwich ....... 25
      Burger ......... 23
      Italian ........ 19
      Chicken ......... 9
      Regional ........ 9
```

As expected, the most popular type of cuisine is the traditional English 'chippy' (Fish and Chips), followed closely by the Asian cuisines Chinese and Indian. Since Pizza is Italian, we could add these together which would put them at the top of the list with 50. However, Pizza is sold in many restaurants and take-aways, for example most Greek Kebab shops also serve Pizza.

Speed Cameras

The roads with known speed cameras in our selected area:

```
SELECT DISTINCT n.value AS name, r.value AS ref, m.value AS maxspeed FROM
(((SELECT id,value FROM ways_tags WHERE key='name') AS n JOIN (SELECT
id,value FROM ways_tags WHERE key='ref') AS r ON n.id=r.id) AS nr JOIN
(SELECT id,value FROM ways_tags WHERE key='maxspeed') AS m ON nr.id=m.id)
WHERE n.id IN (SELECT id AS camera_way_id FROM ways_nodes WHERE node_id IN
(SELECT id AS camera_node_id FROM nodes_tags WHERE key='highway' AND
value='Speed_Camera'));
```

```
Chester Road ..... A56 ... 40 Mph
New Chester Road .. A41 ... 40 Mph
Sandy Lane ........ A534 .. 40 Mph
Welsh Road ........ A550 .. 60 Mph
Gladstone Way ..... A550 .. 30 Mph
Chester Road East . B5129 . 30 Mph
```

Additional Ideas

Postcodes

In addition to the cleaning operations performed, we could add additional data validation checks. For example, we could check whether each recorded post code is valid using Postcodes.io (a free alternative to accessing the Royal Mail PAF).

Postcodes.io [https://postcodes.io] is a postcode & geolocation API for UK postcodes. There is also a useful Python wrapper for the API at [https://github.com/previousdeveloper/PythonPostcodesWrapper].

We can test this idea using `postcode.py`:

For a known valid postcode, `validate_postcode('CH4 0DR')` returns `True`.

For a known fictitious postcode, `validate_postcode('CH9 0ZZ')` returns `False`.

We could also add to the dataset by providing a suggested postcode base on proximity using latitude & longitude. Again, we can test this idea using `postcode.py`:

The official address for Airbus UK in Broughton is

> **Airbus Ltd,**
> **Chester Road,**
> **Broughton,**
> **CHESTER,**
> **CH4 0DR**

The main reception building is shown below:

Choosing one of the nodes defining the reception building (`3063208341`) and passing its latitude and longitude to the `nearest_postcode()` function:

> `nearest_postcode(longitude=-2.9800493,latitude=53.1721903)`
>
> returns a suggestion of '`CH4 0DR`' `32.65`m away, which is correct.

However, there are several possible drawbacks to this. Which postcode would you select if the node is equidistant between 2 roads. And what if the node is closer to a road other than the road its parent way is located, for example when the building is on the corner of two roads.

Conclusions

The OpenStreetMap project will always be a work-in-progress as the landscape is always changing. Consequently, we find many tags named `fixme` & `notes` for entries that require clarification or amendment. We also find that most of the data is supplied by only a few users. The user `daviesp12` provided 46.28% of the data for our selected area.

The data was found to contain an inconsistency in tag placement, with numerous tags found as both node tags and way tags. Node tags should be reserved for point locations such as a crossing on a road, or public telephone location. Way tags should be reserved for descriptions of ways such as type of road, or description of an enclosed area such as the purpose of a building. Unfortunately, due to the open-source nature of the OpenStreetMap project, there are only guidelines and not strict controls over the tag format. This can sometimes lead to problems such as those described here.

For our selected area, we found that the most common pub name is `Red Lion` in some form or other, accounting for 17 out of all the pubs found, followed by `Nags Head` and `White Lion`, each appearing 9 times.

We also found that the most popular type of cuisine is the traditional English `Fish and Chips`, followed closely by the Asian cuisines `Chinese` and `Indian`.

Appendix – Datafiles

```
area.osm .......... 446MB
sample.osm ........ 8.92MB

nodes.csv ......... 160MB
nodes_tags.csv .... 4.23MB
ways.csv .......... 18.3MB
ways_tags.csv ..... 62.9MB
ways_nodes.csv .... 22.8MB

area.db ........... 243MB
```