Wrangle OpenStreet Map –SQL

MAP AREA

Ipswich, Suffolk, United Kingdom

- https://www.openstreetmap.org/relation/53324#map=12/52.0575/1.1654
- https://mapzen.com/data/metro-extracts/your-extracts/6d4dc79de0d8

DATA AUDIT

Unique Tags

In order to explore the data, it was important to know the different types of tags in this xml file. So, I used the *Iterparse* method to parse the Ipswich.osm file and count the number of unique tags. The python script *parsing_file.py* generates the following result:

- 'bounds': 1,
- 'member': 8737,
- 'nd': 345293,
- 'node': 268352,
- 'osm': 1,
- 'relation': 717,
- 'tag': 143538,
- 'way': 46572

Tag Types

Before processing the data and adding it into a database, I checked for the "k" values for each tag to identify potential problems. I have created a dictionary of four tag categories as keys and the count as values.

- "lower", for tags that contain only lowercase letters and are valid,
- "lower_colon", for otherwise valid tags with a colon in their names,
- "problemchars", for tags with problematic characters, and
- "other", for other tags that do not fall into the other three categories.

audit_tags.py generates the following output :

{'lower': 120482, 'lower_colon': 12223, 'other': 10833, 'problemchars': 0}

Street Names

1. Audit Process :

- After exploring a data a bit more, I found there abbreviations in street names such as, ‘Ave’ for Avenue, ‘Ln’ for Lane and ‘Rd’ for Road.

- Below is a snapshot of the output after running *audit_streetnames.py.*

```
{'Area': set(['Bentley Service Area']),
 'Ave': set(['Lloyds Ave']),
 'Boulevard': set(['Crane Boulevard',
                   'Jamestown Boulevard',
                   'Mansbrook Boulevard']),
 'Bungalows': set(['Hall Farm Bungalows']),
 'Buttermarket': set(['Buttermarket']),
 'Cornhill': set(['Cornhill']),
 'Court': set(['Samuel Court']),
 'Driftway': set(['The Driftway']),
 'East': set(['Goddard Road East', 'New Cut East', 'Woodbridge Road East']),
 'Europark': set(['Ransomes Europark']),
 'Gardens': set(['Reeve Gardens']),
 'Green': set(['Maidenhall Green', 'The Green']),
 'Hamlet': set(['Fore Hamlet']),
 'Havens': set(['The Havens']),
 'Hill': set(['Constitution Hill', 'Mow Hill']),
 'Hintlesham': set(['Hintlesham']),
 'Interchange': set(['Copdock Interchange']),
 'Ln': set(['Pannington Hall Ln']),
 'Market': set(['Old Cattle Market']),
u'Plain': set([u'St Margaret\u2019s Plain']),
 'Ramparts': set(['Tower Ramparts']),
 'Rd': set(['Foxhall Rd']),
 'Sandlings': set(['The Sandlings']),
 'South': set(['Anglia Parkway South']),
 'Thoroughfare': set(['The Thoroughfare', 'Thoroughfare']),
 'West': set(['New Cut West', 'Sidegate Lane West', 'Stone Lodge Lane West'])}
```

- Above output is a dictionary with key as the last word in the street name and the corresponding value is the entire street name.
- I have used the following regular expression to extract the last word from the street names: *r'\b\S+\.?$'*

2. Cleaning Process:
   - Using the following mapping technique I updated the street names in the xml file.

     mapping = { "Ln": "Lane",
     "Rd":"Road",
     "Rd.":"Road",
     "Ave":"Avenue"}
   - Below are some expected street name values (from my general awareness of the town) in the data set which we do not want to update.

expected=["Road","Lane","Avenue","Place","Quay","Street","Walk","Way", "Close","End","Drive","Approach","Centre","Park"]

- Using the above two and the regex in the step 1, I created a list of updated street names such that name => better name. Please see the output below:

```
Copdock Interchange => Copdock Interchange
Mansbrook Boulevard => Mansbrook Boulevard
Crane Boulevard => Crane Boulevard
Jamestown Boulevard => Jamestown Boulevard
Samuel Court => Samuel Court
Bentley Service Area => Bentley Service Area
Pannington Hall Ln => Pannington Hall Lane
Stone Lodge Lane West => Stone Lodge Lane West
Sidegate Lane West => Sidegate Lane West
New Cut West => New Cut West
Foxhall Rd => Foxhall Road
Constitution Hill => Constitution Hill
Mow Hill => Mow Hill
New Cut East => New Cut East
Woodbridge Road East => Woodbridge Road East
Goddard Road East => Goddard Road East
The Thoroughfare => The Thoroughfare
Thoroughfare => Thoroughfare
Hall Farm Bungalows => Hall Farm Bungalows
The Sandlings => The Sandlings
Reeve Gardens => Reeve Gardens
Buttermarket => Buttermarket
The Driftway => The Driftway
Tower Ramparts => Tower Ramparts
St Margaret's Plain => St Margaret's Plain
Maidenhall Green => Maidenhall Green
The Green => The Green
The Havens => The Havens
Anglia Parkway South => Anglia Parkway South
Ransomes Europark => Ransomes Europark
Hintlesham => Hintlesham
Cornhill => Cornhill
Fore Hamlet => Fore Hamlet
Lloyds Ave => Lloyds Avenue
Old Cattle Market => Old Cattle Market
```

3. I have updated these streets names and generated a cleaned xml (clean_ipswich.osm) file using *clean_file.py.* I performed ann audit again on the cleaned osm file to check for effectiveness. We can see below, the abbreviations ***Ln, Ave, Rd*** do not appear in the output because of the cleaning process in step 2. The street names now fall in the ***expected*** list!

```
{'Area': set(['Bentley Service Area']),
 'Boulevard': set(['Crane Boulevard',
                   'Jamestown Boulevard',
                   'Mansbrook Boulevard']),
 'Bungalows': set(['Hall Farm Bungalows']),
 'Buttermarket': set(['Buttermarket']),
 'Cornhill': set(['Cornhill']),
 'Court': set(['Samuel Court']),
 'Driftway': set(['The Driftway']),
 'East': set(['Goddard Road East', 'New Cut East', 'Woodbridge Road East']),
 'Europark': set(['Ransomes Europark']),
 'Gardens': set(['Reeve Gardens']),
 'Green': set(['Maidenhall Green', 'The Green']),
 'Hamlet': set(['Fore Hamlet']),
 'Havens': set(['The Havens']),
 'Hill': set(['Constitution Hill', 'Mow Hill']),
 'Hintlesham': set(['Hintlesham']),
 'Interchange': set(['Copdock Interchange']),
 'Market': set(['Old Cattle Market']),
 u'Plain': set([u'St Margaret\u2019s Plain']),
 'Ramparts': set(['Tower Ramparts']),
 'Sandlings': set(['The Sandlings']),
 'South': set(['Anglia Parkway South']),
 'Thoroughfare': set(['The Thoroughfare', 'Thoroughfare']),
 'West': set(['New Cut West', 'Sidegate Lane West', 'Stone Lodge Lane West'])}
>>>
```

DATA OVERVIEW

I exported the data into csv's using *data_csv.py* and created the database using *database.py*

File	Size
ipswich.osm	57.9 MB
clean_ipswich.osm	73.5 MB
clean_ipswich.db	42.1 MB
nodes.csv	20.6 MB
nodes_tags.csv	1.13 MB
ways.csv	2.62 MB
ways_nodes.csv	7.97 MB
ways_tags.csv	3.84 MB

Following SQL queries reveal some basic statistics about the data set.

Number of Nodes:

sqlite> SELECT COUNT(*)

...> FROM nodes;

268456

Number of Ways:

```
sqlite> SELECT COUNT(*)
   ...> FROM ways;
```

46572

Number of unique users:

```
sqlite> SELECT COUNT(DISTINCT(T.uid))
   ...> FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) T;
```

296

Top contributing users:

```
sqlite> SELECT T.user, COUNT(*) as num
   ...> FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) T
   ...> GROUP BY T.user
   ...> ORDER BY num DESC
   ...> LIMIT 10;
```

Rowland|219013

PeterEastern|32269

smsm1|10583

The Maarssen Mapper|6330

Robert Whittaker|4899

IpswichUser99|4696

BCNorwich|2783

AVERAGE_INDIVIDUAL|2549

ADDITION DATA EXPLORATION AND IDEAS:

‘fixme’ value

I ran a query to identify ‘shops’ in Ipswich and came across a certain ‘fixme’ associated with a key=’shop’.

sqlite> SELECT VALUE

...> FROM nodes_tags

...> WHERE key='shop'

...> GROUP BY VALUE;

Here is a snapshot of the partial result:

estate_agent

fishing

fixme

florist

funeral_directors

funeral_parlor

furnishings

I ran the following query to find out all the nodes that have key=’fixme’ or value=’fixme’ and noticed that value for ‘fixme’ keys contain interesting questions posed for a particular node. Here is the result of the query:

sqlite> SELECT id ,value

...> FROM nodes_tags

...> WHERE Key='fixme' OR value='fixme'

...> GROUP BY Id;

Result:

1069452238|incomplete

1073628197|Where does Public Footpath go from here?

2035446350|Where does Public Footpath go from here?

2096821362|Where does Public Footpath go from here?

2096862722|Where does Public Footpath go from here?

2138824383|unsafe, you cannot lock more than one bike properly

2361007180|fixme

24950236|Where does Public Footpath go from here?

27125326|is footpath reachable from this point?

2973094047|Buttermarket redevelopment has removed bulding here, and no sign of

this postbox. Check where postbox is now.

3078362453|incomplete

3078362455|incomplete

3078362458|incomplete

3078362523|incomplete

3078999822|incomplete

3078999953|incomplete

3079000007|incomplete

3079000027|incomplete

3079163912|location uncertain -- on this side of the road between memorial and Barley Mow pub.

3079248462|check location

3118026161|position approx

3118026180|position approx

3268569798|incomplete

3269314050|incomplete

3377504526|incomplete

3377504527|incomplete

3736539751|incomplete

4707949504|Can you get out to the road here?

546463707|Does Public Footpath continue?

694593417|Where does Public Footpath go from here?

For example, the id for key='shop' and value='fixme' is 2361007180. I checked the latitude and longitude values on Google map which shows a restaurant 'K Bar and grill' at this location which is not a shop!

https://www.google.co.uk/maps/place/52%C2%B003'18.5%22N+1%C2%B009'09.4%22E/@52.055169,1.1525533,79m/data=!3m1!1e3!4m5!3m4!1s0x0:0x0!8m2!3d52.0551406!4d1.1526219

It may be useful to create a list of such nodes with 'fixme' tags and encourage contributors to update the data. I have created a csv file including the latitude and longitude values for these nodes *fixme_locations.csv* using the following query:

sqlite> .headers on

sqlite> .mode csv

sqlite> .output fixme_locations.csv

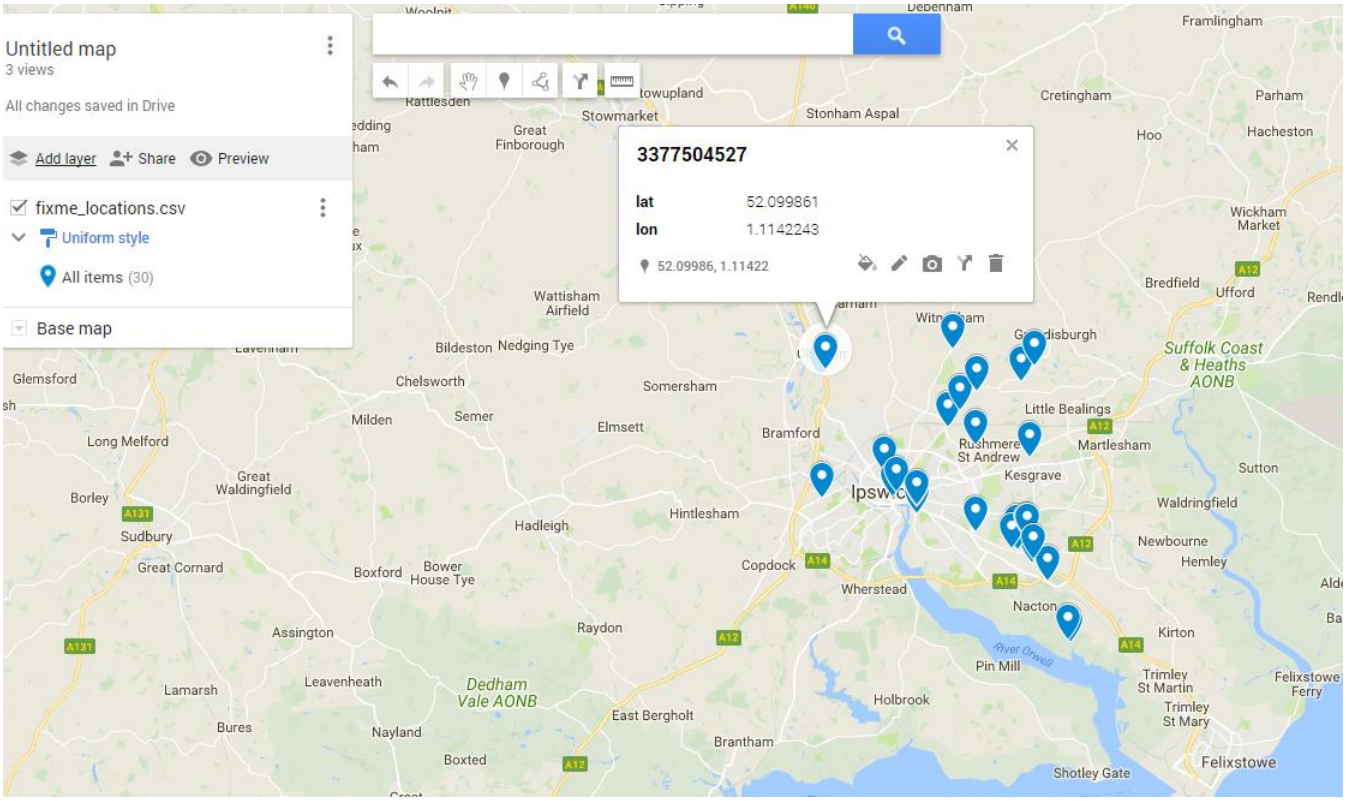
sqlite> SELECT lat,lon,nodes_tags.id

...> FROM nodes_tags

...> JOIN nodes ON nodes_tags.id=nodes.id

...> WHERE Key='fixme' OR Value='fixme';

I have plotted these locations on google map to cross check the missing values. The map suggests some of these places are in the country side for example on a farm.



It may be helpful to create a cross validation tool using the list of ‘fixme’ node locations list. This tool can be an in-built feature within maps and can also be used to encourage users to update any information they know. For example, there can be a dialog box where users can enter values for ‘fixme’ locations: shops, roads etc.

‘maxspeed’ value

Next I checked for the various speed limits on roads in Ipswich

```
sqlite> SELECT Value, COUNT(*) AS num
   ...> FROM ways_tags
   ...> WHERE Key='maxspeed'
   ...> GROUP BY Value
   ...> ORDER BY num DESC;
```

Result:

30 mph|2597

20 mph|224

40 mph|208

60 mph|196

70 mph|169

UK:nsl_dual|90

GB:nsl_dual|63

15 mph|40

50 mph|40

UK:nsl_single|40

100 mph|28

survey 2013-08-09|27

Stats19 and interpolation|20

GB:nsl_single|8

stats19|8

survey 2013-08-30|8

2013-08-30|6

survey|4

10 mph|3

this odd section is indeed 70mph - I believe the legals for restricting the speed failed to include this setion|3

30|2

45 mph|2

5 mph|2

Sats19|2

national|2

25 mph|1

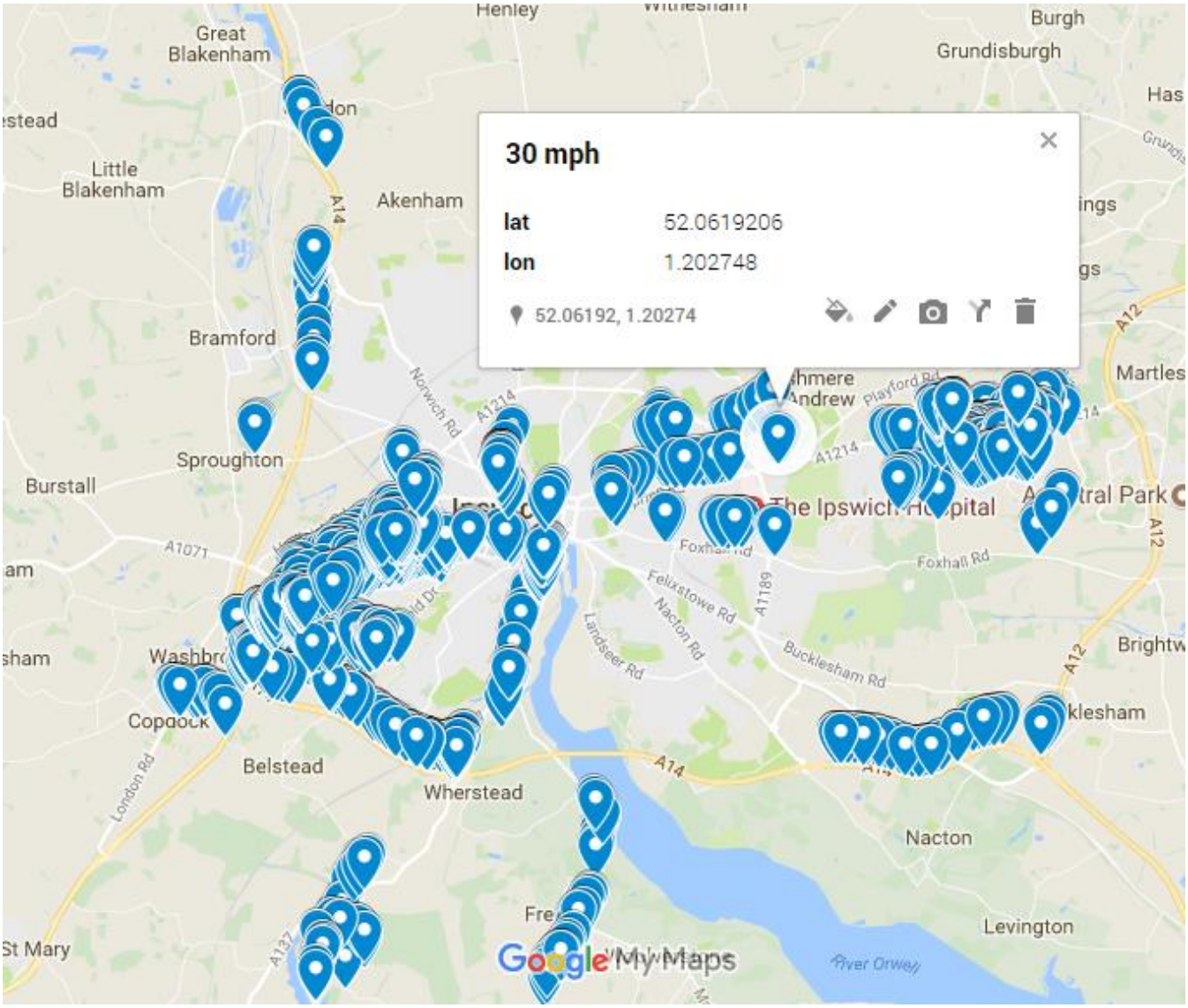
We can clearly see inconsistencies and wrong information presented above. The UK national speed limit is 70 mph so there is no way we have 100 mph roads in Ipswich. Also, multiple abbreviations for UK /GB is used. Improving this data can be useful in building an Openstreet map navigation app for customers warning them of speed limits on various roads. I did some more research on how the speed limits in UK are represented in Open street map.

http://wiki.openstreetmap.org/wiki/Key:source:maxspeed

http://wiki.openstreetmap.org/wiki/Speed_limits

There is some argument on the representation of these speed limits in the UK. Hence, I chose to use the values as it is to generate the following map:

```
sqlite> .headers on
sqlite> .mode csv
sqlite> .output speed_locations.csv
sqlite> SELECT lat,lon,value
   ...> FROM (SELECT * FROM ways_tags JOIN ways_nodes ON ways_tags.id=ways_nodes
.id) e
   ...> JOIN nodes on e.node_id=nodes.id
   ...> WHERE key='maxspeed';
```



This data can be used to build apps for customers warning them of speed limits or can be integrated with the car navigation system etc.

Conclusion

The OpenStreet Map data of Ipswich is fairly clean and with less typos. There is a lot of scope for additional improvements in this dataset and making it suitable for building apps or adding features in this map. As suggested above, we can use this data in the following ways:

1. Creating a cross validation tool (I used google maps for now) to get rid of 'fixme' values or missing values by encouraging users to input data. We can also have an in-built parser within the cross validation tool to restrict typos etc. This tool can also include scripts to regularly audit and clean the data.
   Benefits:
   - This will reduce dependency on contributors to audit/clean the data.
   - The rules and patterns to input information will restrict use of typos or abbreviations.

- Help improve accuracy and usability of the map

Anticipated issues:

- Although the tool is somewhat automated but still depends on users to input information. Therefore, the users must be motivated to enter information.
- Time and cost constraints are associated as it requires a developer to build this script which is customised for every town/city and country.

2. We can use the 'maxspeed' data to build additional functionality in the maps to warn drivers of the speed limits.

   Benefits:

   - There are maps in the market that already have this feature and have been integrated with the car navigation system. Therefore, adding this feature will enable the Openstreet Maps to be at par with others in the market.

   Anticipated issues:

   - There is missing information and possibly wrong information presented in this data set. This problem is presented in 'maxspeed' value.
   - Periodical data updates are required for this feature to work perfectly.