OpenStreetMap Project
Data Wrangling with MongoDB

Andrius Dalisanskis
andrius.dalisanskis@gmail.com

Map Area: Dublin, Ireland
https://s3.amazonaws.com/metro-extracts.mapzen.com/dublin_ireland.osm.bz2

1. Problems Encountered in the Map

After downloading the Dublin area map we run it against a provisional audit.py file to check the values for the keys that we are interested in. I noticed that values for keys amenity, shop, cuisine, phone have some issues:

- Letter cases. For example we can find keys with the values named hardware and Hardware. We will fix this by using all values in lower case.
- Using/not using underscores or other characters to separate words. For example for the shop key we can find following values: Shopping Centre and shopping_centre. To make values follow the same format we will join words with an underscore.
- Using singular/plural forms of words for same values. For example we can find 7 carpet and 1 carpets values for the key shop.
- Grammatical mistakes as in **Chineese**_&_Thai
- Different ways to separate words if there is more than one value in the field as in:

Sandwiches,_Bakery,_Salads,_Frozen_Yoghurt
Thai_&_Vietnamese
chinese;thai

We will fix this by separating values with a comma sign:

Sandwiches,Bakery,Salads,Frozen_Yoghurt

To fix the grammatical mistakes and to separate values with comma sign only we will create a dictionary with wrong and correct word/symbol pairs and will use it to make replacements as needed.

- Phone number values are written in many different ways as in:

+353 1 6684364 (superintendent herbert park)
+353 1 465 1964
(01) 219 5966
(003531) 6683075
+ 353 (0)1 8332321
+353-1-8436346
+353 1 ____4061

We will make phone numbers to follow same format - 00353 xx xxx xxxx. First we will remove any non digit characters. After doing so we can see that we get several big groups:

0035318730606	// 00 353 1 xxx xxxx
012028300	// 01 xxx xxxx
353012826283	// 353 01 xxx xxxx
35318031004	// 353 1 xxx xxxx
353831171786	// 353 [83] xxx xxxx, 83 85 86 87 89 *are possible codes for mobile operators*

We can extract the required digits using the regular expressions to recognise which group the phone number belongs to. We will discard the rest few of the numbers that can not be grouped into these larger groups.

2. Data Overview

File size
dublin_ireland.osm .... 219 MB

- Number of documents

```
db.cleaned_data.count()
1121887
```

- Number of nodes

```
db.cleaned_data.find({"type":"node"}).count()
957096
```

- Number of ways

```
db.cleaned_data.find({"type":"way"}).count()
164791
```

- Number of unique users

```
len(db.cleaned_data.distinct('created.user'))
1003
```

- Top 1 contributing user

```
q = [{'$group':{'_id':'$created.user', 'count':{'$sum':1}}},{'$sort' : {'count' : -1}},{'$limit' : 1}]
db.cleaned_data.aggregate(q)['result']

[{u'count': 228243, u'_id': u'Nick Burrett'}]
```

- Number of users that have only one edit

```
q = [{'$group':{'_id':'$created.user', 'count':{'$sum':1}}},
   {'$group':{'_id':'$count', 'user_count':{'$sum':1}}},
   {'$sort':{'_id':1}},
   {'$limit':1}]
db.cleaned_data.aggregate(q)['result']

[{u'_id': 1, u'user_count': 244}]                  # _id is the number of contributions
```

3. Additional Ideas

The map data is entered in many different formats as we could see in the cleaning part of the project. To improve the data format the values which are allowed to be typed in manually could be programmatically cleaned up before saving them to the map. For example letter cases could be checked and fixed as required or formatting the value so that words are separated using only one standard way (like an underscore or space). The values that should have a strict format (for example phone numbers) could be checked against regular expressions, in case of mismatch a user should see a message with tips how to improve his entry.

We can find values that are common for few keys. For example keys `shop` and `amenity` have few values that they share:

```
[{u'commonToBoth': [u'taxi', u'ice_cream', u'yes', u'pub', u'fuel']}]
```

Introducing on screen better explanatory tips for users while entering data could reduce this effect.

The completeness of entries is quite an issue as well. For example we can find 316 shops that have `name` field missing. One way of encouraging users to provide more details about nodes would be to introduce scores or ranks and leaderboards. Competition could have a positive impact on users by pushing them to have their entries as full as possible.

Additional data exploration using MongoDB queries

- Number of `shops` that have name missing

```
>db.cleaned_data.find({'shop':{'$exists':1}, 'name':{'$exists':0}}).count()
316
```

- List of values that we can find in shop AND amenity fields

```
q = [{'$group':{'_id':None, 'shops':{'$addToSet':'$shop'},
'amenities':{'$addToSet':'$amenity'}}},
{'$project':{'_id':0, 'commonToBoth':{'$setIntersection':['$shops','$amenities']}}}]

>db.cleaned_data.aggregate(q)['result']

[{u'commonToBoth': [u'taxi', u'ice_cream', u'yes', u'pub', u'fuel']}]
```

- Top 2 cuisines in Dublin

```
q = [{'$unwind':'$cuisine'},
{'$group':{'_id':'$cuisine', 'count':{'$sum':1}}},
{'$sort':{'count':-1}},
{'$limit': 2}]
>db.cleaned_data.aggregate(q)['result']

[{u'count': 100, u'_id': u'chinese'}, {u'count': 83, u'_id': u'fish_and_chips'}]
```

Conclusion

After reviewing the map data we can see the Dublin area is incomplete. There are consistency issues - users do mix up different fields when entering data. Many different formats are chosen as well to fill certain data types (for example phone numbers). Adding additional functionality and auditing features to the tools that are used to enter map data could help users to add more relevant and clean data. The gamification of the process (giving scores/ranks) could motivate users as well to give more detailed descriptions of map elements.