OpenStreetMap Project

Rupa Bisaria

September 7, 2017

Contents

OpenStreetMap Data

OpenStreetMap is an editable map of the world, with contribution from volunteers from all over the world. This project creates and distributes free geographic data of the world.

Quoting from it's wiki page:

OpenStreetMap represents physical features on the ground (e.g., roads or buildings) using tags attached to its basic data structures (its nodes, ways, and relations). Each tag describes a geographic attribute of the feature being shown by that specific node, way or relation.

OpenStreetMap's free tagging system allows the map to include an unlimited number of attributes describing each feature. The community agrees on certain key and value combinations for the most commonly used tags, which act as informal standards. However, users can create new tags to improve the style of the map or to support analyses that rely on previously unmapped attributes of the features.

Project Objective

The project required to select an *'area of the world in https://www.openstreetmap.org and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the OpenStreetMap data'*. This clean data was to be saved to either MongoDB or SQL database, as per a pre-specified data schema. Finally, data was to be analysed as per the Project Rubric.

Data Description

Data from Dubai-Abu Dhabi region from the following link: https://s3.amazonaws.com/metro-extracts.mapzen.com/dubai_abu-dhabi.osm.bz2 was used for the project.

Dubai and Abu Dhabi are part of a federation of emirates called United Arab Emirates (UAE). UAE is a federal absolute monarchy located at the southeast Arabian Peninsula in Western Asia. It is a federation of seven emirates, namely, Abu Dhabi, Ajman, Dubai, Fujairah, Ras Al Khaimah, Sharjah and Umm Al Quwain.

Data Wrangling

Since a large part of the data is user generated, there is a huge likelihood of finding many discrepancies and errors in the data, which might need to be cleaned.

There are 862 unique keys of the 'tag' elements of elements 'node' and 'way'. These are basically describing various map features in an open street map.

Here are the top twenty keys sorted in descending order of their occurence in the data:

Key	Count
highway	183227
source	91029
building	88605
name	77374
surface	73926
oneway	69162
addr:street	52428
lanes	35757

Key	Count
name:ar	31158
addr:housenumber	29740
amenity	23222
shop	20169
addr:city	19997
power	17391
addr:city:ar	16049
addr:street:ar	15889
landuse	14473
barrier	13353
maxspeed	10581
natural	10299

Few of these tag keys were further investigated. The golden list of expected values were taken from here and here

building

These were few values which didn't match the expected values:

```
['Office_And_entrance', 'office', 'Airport_terminal', 'Gate 3', 'Tourist_Exhibition',
'Offices', 'yes;mosque', 'MAJ Building', 'Complex_A_&_B']
```

As per wiki, building=commercial with office=* is to be used to describe the type of office. Also, building=yes is to be used where it is not possible to determine a more specific value.

For building value with 'yes;mosque', there are only 4 count of this value in the database as per taginfo web site, where most frequently used tag is 'mosque'. While wiki does allow such semi-colon separated values, it is also advised to use it only if necessary.

With these pointers, following mapping were done for the erroneous entries for building.

Dirty	Clean
Airport_terminal	terminal
MAJ Building	yes
Tourist_Exhibition	yes
Gate 3	yes
Offices	commercial
yes;mosque	mosque

surface

While most of the values were fine, there were few with spelling mistakes and some with extra backticks `.

```
['unpaveds', 'running surface', 'asphalt`', 'Elevated', 'unpaved`', 'paving stones',
'paving_stoness', 'pavin', 'paving`']
```

These were mapped against corrected values as follows:

Dirty	Clean
running surface	running_surface

Dirty	Clean
Elevated	elevated
paving_stoness	paving_stones
pavin	paving
paving stones	paving_stones
unpaveds	unpaved
‘asphalt	asphalt

oneway

Audit of ‘oneway’ values resulted in following values:

`['Street 43', 'yes', '-1', 'tertiary', 'no']`

while the expected values as per wiki are:

`['yes','no','-1','reversible','alternating']`

Since there are only 6 entries whose values did not match the expected set of values, they were taken as ‘None’ for this project.

addr:city

A list of most of the major cities in the region was created for mapping the cities in the data.

There were lot of problems in the data entries, starting from spelling errors, uppercase letters, non-city areas marked as city, arabic names and just a digit as an entry for a city. There were also few arabic entries too.

```
Business Bay, Dubai
abu dhabi
1
5
Al Rahba DUBAI
```

There were some values like `town` and `ME-12` or even `AE`. How mistakes can creep in into user-generatde data can be highlighted by the fact that there is an entry as `San Diego, CA` in Dubai-Abu Dhabi region! All such values were mapped to ‘None’. All the arabic names and numbers were also set to ‘None’

Although the map name suggests data from Abu Dhabi and Dubai, there were city data from other emirates too, which if fine.

Names with spelling errors or with all uppercase letters were mapped accordingly.

Dirty	Clean
DUBAI	Dubai
abu dhabi	Abu Dhabi

Entries were city names were accompanied with stree or other information were corrected as follows:

Dirty	Clean
Dubai Media City, Dubai	Dubai
Jumeirah Beach, Dubai	Dubai

addr:street

Since many of the street names had spelling and typing errors, these values were mapped against correct values as follows:

Eg:

Dirty	Clean
St	Street
st.	Street
sweet	Street
rd	Road
Roud	Road
Streetrr	Street
blvd	Boulevard
Atreet	Street

Further, there were other issues which were sorted out one by one by using regex. Since the issues were so inter-twined, regex was required to be applied in a particular sequence to finally get the desired result.

Eg:

Dirty	Clean
Street 6a	Street 6A
11 B	Street 11B
2-A	Street 2A
12 D St	Street 12D
6b St	Street 6B
Sa'ada Street (19th)	Sa'ada Street 19
55	Street 55
Jumeirah Village Triangle, District 2, Street 5	Street 5
Al Satwa StreetAl Bandoq Vegetable & Fruits	Al Satwa Street
13C street , Al Quoz Industrial 3	Street 13C

There were entries like

```
Al khail rd. business bay. Al khaleej al tejari 2 st. Al amal st. Churchill towers
```

```
Rose 2 - 17a St - Dubai 17a St Dubai United Arab Emirates
```

which although look like correct entry but has so many extraneous information, it was very difficult to understand which is the correct name of the street.

There were lot of errors in data as not all the data entry were pertaining to the street name.

```
khawaneej
```

```
Paragon Mall
```

```
No Fear Cafe
```

```
Arabian Ranches
```

```
Ibn Sina Medical Centre
```

```
24°26'24.5"N 54°27'03.8"E
```

```
P.O. Box 34429
```

`Bahar 7 Jumeirah Beach Res, Marsa, Dubai, AE`

`Plot No. M-26, Area 54`

`Mezzanine Level, Spinneys Supermarket`

`residential`

`High Bay Business Center`

These values look very obviously to be wrong entries, hence such data entry were ignored and set as 'None'.

There were values like

`(Inside of the Hotel The Cove Rotana Resort)`

`Camel race track near Sweihan`

where user has taken pains to put some description, but it may not be good enough. Here I chose to keep the later as `Camel Race Track Near Sweihan`

arabic entries

Keys like 'addr:street:ar' and 'name:ar' with suffix 'ar' were all with values in arabic and have been ignored for this project.

Data Exploration

Data file size

Uncompressed data file: 'dubai_abu-dhabi.osm' is approximately 500MB in size.

Number of nodes

There are in total 2431529 node elements.

```
# SQL query:
cursor = conn.execute("SELECT COUNT(*) FROM nodes")
print "Node count  = ", cursor.fetchone()[0]
```

Number of ways

There are in total 324346 ways elements.

Sql query for counting the number of ways:

```
# SQL query:
cursor = conn.execute("SELECT COUNT(*) FROM ways")
print "Ways count  = ", cursor.fetchone()
```

Unique users

There are 2450 unique user who contributed to the data.

Sql query for getting the number of unique user ids:

```
# SQL query:
cursor = conn.execute("SELECT COUNT(DISTINCT(uid)) from nodes")
print "Unique uid count  = ", cursor.fetchone()[0]
```

Top ten users

List of top ten users with most contibution:

No.	user	uid	num
0	eXmajor	561234	403358
1	chachafish	2237750	141500
2	Seandebasti	550560	137851
3	Alex111X	3451386	136847
4	csdf	28794	61949
5	SEVEN	127989	60409
6	hellosun	6127501	58903
7	AndrewBuck	214969	53321
8	gellazp	2199587	52969
9	MozlTosh	6127498	43238

```
# SQL query:
df = pd.read_sql_query("SELECT user, COUNT(user) as num FROM nodes \
                       GROUP BY uid ORDER BY num DESC;", conn)
df[0:10]
```

User 'eXmajor' has made highest contribution to the data. His top ten contributions are:

key	num	
0	highway	17105
1	building	4550
2	oneway	3892
3	barrier	2468
4	access	2458
5	amenity	2322
6	landuse	2242
7	leisure	1206
8	natural	1059
9	aeroway	564

```
# SQL query:
df = pd.read_sql_query("SELECT ways_tags.key, COUNT(ways_tags.key) as num FROM ways_tags \
                       JOIN ways ON ways_tags.id = ways.id \
                       WHERE ways.uid = '561234' \
                       GROUP BY ways_tags.key ORDER BY num DESC;", conn)
```

Top ten keys

For Nodes

```
# SQL query:
df = pd.read_sql_query("SELECT key, COUNT(key) as num FROM nodes_tags \
                       GROUP BY key ORDER BY num DESC;", conn)
df[0:10]
```

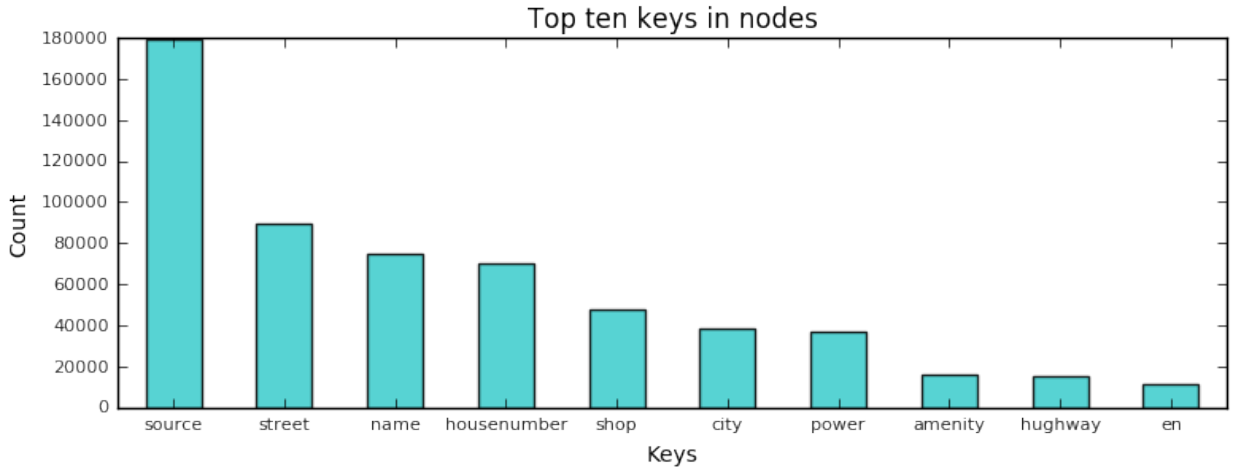
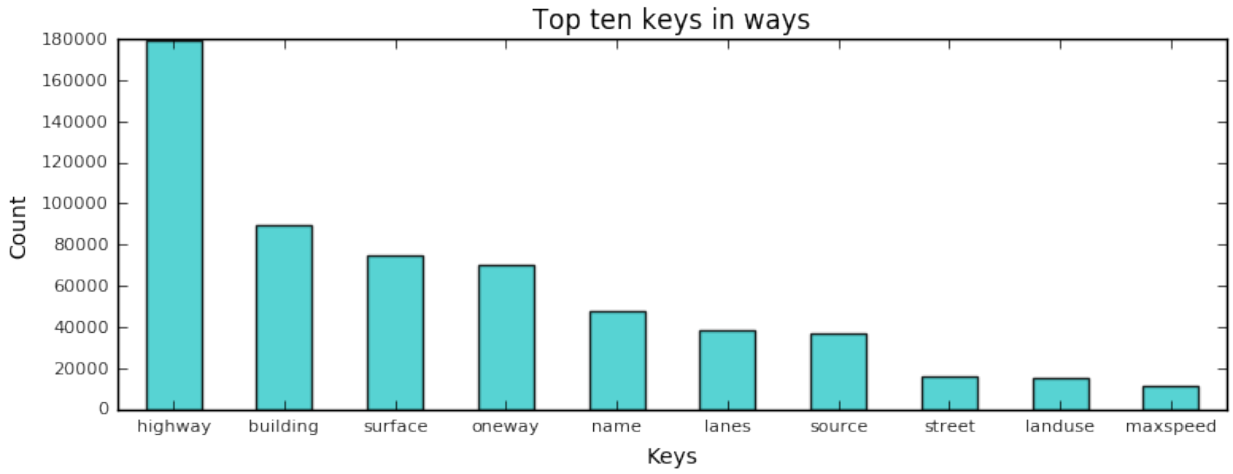


Figure 1:

For Ways

```
# SQL query:
df = pd.read_sql_query("SELECT key, COUNT(key) as num FROM ways_tags \
                       GROUP BY key ORDER BY num DESC;", conn)
df[0:10]
```



]

Maximum nodes in a way element

One of the way element has 1960 node tags.

```
# SQL query:
cursor = conn.execute("SELECT MAX(A.node_count) FROM \
           (SELECT COUNT(node_id) AS node_count FROM ways_nodes \
           GROUP BY id) A")
print "(Tag Id, Node Count)   = ", cursor.fetchone()[0]
```

Further investigating the dataset yielded more of such tags with more than 1000 nodes.

```
# SQL query:
cursor = conn.execute("SELECT A.id, A.node_count FROM \
              (SELECT id, COUNT(node_id) AS node_count FROM ways_nodes \
              GROUP BY id) A WHERE A.node_count > 1000")
print "(Tag Id, Node Count) = ", cursor.fetchall()
```

```
(Tag Id, Node Count) =  [(170139278, 1086), (171171012, 1075), (194470882, 1094), (205958924,
1226), (216720271, 1628), (393225171, 1408), (393249907, 1434), (393254003, 1180), (393420807,
1266), (402883870, 1592), (402884861, 1450), (404074917, 1960), (404089376, 1469), (440574399,
1062), (440574403, 1133), (440574568, 1524), (440574586, 1264)]
```

Checking for the details of the associated feature yielded a coastline, which explains the number of nodes.

```
# SQL query:
cursor = conn.execute("SELECT key, value FROM ways_tags WHERE id = '402884861'")
print "Associated keys and values  = ", cursor.fetchall()
```

```
Associated keys and values  =  [(u'natural', u'coastline')]
```

Improvement Suggestions

While wrangling the dataset, one things that came out very starkly was user's confusion about the stated key and it's required value. One could actually see sometimes users struggling to enter relevant data. Here are few suggestions that came in my mind in the process:

- Since many a values were required to be all in lowercase, text box for the same can be constrained accordingly. Also, although semi-solons are allowed, it has been advised to use it sparingly in the wiki. There could be a suggestion to the user for alternate stragegy if too many semi-colons are observed in the text box. This solution though could be applicable only to few of the data entries.
- There were quite a few arabic entries in english language fields, which can also be constrained at the time of data entry.
- Since there are lot many spelling mistakes, and also, multiple versions of the same word like for eg: paved_stones and paved_stone, why not have a list of suggestions, a la Google, as a user is typing in, thereby, eleminating such scenario. There could be a spell check as well if that helps.
- There were few numeric entries for city names, and that seems little unlikely. An entry for such a field can be flagged as error to the user.

- There could be drop down menu of values if the list is reasonably small, for example, for oneway expected values are [‘yes’,‘no’,‘-1’,‘reversible’,‘alternating’] which can be easily picked from a drop down menu. This solution is of course constrained by the length of dropdown.

Since, the open street map is used by people from world over, entering data in different languages, all these suggestions might not be applicable to all for various technical and other reasons. All said and done, getting a clean data at such a huge scale is an arduous task.

Conclusion

As expected from any user generated data, there were quite a few errors and discrepancies in the data. While some had very few errors, other keys like street oy city names had lot of dirty entries. Also, while there were gold standards for some data entries which could be clean accordingly, there were no such standards easy available for, for example, city names. This project was a just a small attempt at cleaning a huge dataset like an open street map data, and wrangling such a data set for a clean version is indeed an enormous task.