BOSTON OpenStreetMap

This project aims at curating the Boston Open Street Map. The report is divided into the following sections:

I. Problems encountered in the map
   1. Audit type of tags
   2. Audit street names
   3. Audit postal codes
   4. Audit 'tourism' tags
II. Overview of data
   1. Size of the files
   2. Number of nodes and ways
   3. Number of Unique users
   4. Top 10 contributing users
   5. Top 10 tourism types
   6. Top 10 values and counts in leisure tags and tourism tags that the top contributor had curated
   7. Maximum number of nodes required for way elements with the top 10 tourism tags
III. Other ideas about the dataset
   1. More on curating the street names – street number in front of the street name
   2. Looking at 'addr:street' audit as a case study for the Boston OSM quality
   3. Looking at 'museum' tags as a case study for the Boston OSM quality

Problems encountered in the map

---

1. Audit type of tags

I first used audit_tag.py to look for potential problems in the tags. There were 0 tags with problematic characters, 791025 lower-cased tags, 71767 tags lower-cased tags with colon separation, and 39912 tags that are in the other categories.

For "address" tags, in addition to the conventional "address", "addr:street" and "addr:housenumber" tags there were two other tags, "addr:street_1" and "addr2:housenumber". I decided to also audit the street names in addr:street_1 but will ingore the addr2:housenumber since I will not look into the housenumbers.

2. Audit street names

I used audit_street_name.py and found that there were 67 street names that require revision. The problems encountered and the solution that I chose are summarized as follows:

A. Street names with abbreviations (Ave, Ave., Ct, Dr, HIghway, Hwy, Pkwy, Pl, place, Rd, rd., Sq., ST, St, St,, St., Street., st, street) –> change to standard street names
B. Street number included after “,”–> delete street numbers after “,”
C. Street name followed by # suite number –> delete suite number after “#”
D. Only name of the street but not street type specified (Elm, Cambridge, Broadway, Windsor, Holland, Newbury) –> Leave as is because not enough information here to curate. Maybe can be fixed in the database when there are more information to pull out together.
E. Unconventional address/street names (Kendall Square, Cambridge Center) –> leave as is

```
def clean_street_name(name, mapping):

    # delete number after street name,
    if "," in name:
        return name.split(",")[0]

    # delete suite number after street and fixed one abbreviated street type
    elif "#" in name:
        if "Harvard" in name:
            return "Harvard Street"
        else:
            name_split_pound = name.split("#")
            return name_split_pound[0]

    # map all street names in question to standard street names
    else:
        name_as_list = name.split(" ")
        if name_as_list[-1] in mapping.keys():
            name_as_list[-1] = mapping[name_as_list[-1]]
            name = " ".join(name_as_list)
            return name
        else:
            return name
```

3. Audit postal codes

I used audit_postcode.py and identified 125 unique postal codes in the area. The problems are mainly four types: postal codes with wrong

format, postal codes that are outside of the area, typo or postal codes that has been changed.

I decided to update the formats so that they all have five digits and return 00000 into the database for those postal codes that are outside the area for future curation. For the postal codes that have been changed over the years, because the correct information can still be found relatively easily online, I will leave them as is.

Details and the solutions are summarized as follows:

A. zip code that is outside of MA: 01125, 20052 --> will return 00000 in the database
B. zip code that is outside of the greater boston area: 01238, 01240 --> will return 00000 in the database
C. zip code that is outside of the USA: 01250 --> will return 00000 in the database
D. zip code change 02159 to 02459 for Newton Center, Newton (2002), 02174 to 02474 or 02476 for Arlington (1998), etc --> will leave it as is
E. zip code with 4 digit extension --> will remove all 4 digits and the dash
F. 4 digit zip code: 0239 --> will return 00000 in the database
G. 'MA' instead of the zip code --> will return 00000 in the database
H. 'MA' followed by the 5 digit zip code --> will return 00000 in the database

```
def clean_postcode(postcode):

    # delete -XXXX after the five digit postcode
    if "-" in postcode:
        return postcode.split("-")[0]

    # delete MA in the postcodes
    elif "MA" in postcode:
        new_postcode = postcode.replace("MA ", "")
        if len(new_postcode) == 5:
            return new_postcode
        else:
            return "00000"

    # return "00000" for postcodes that are less than 5 digits
    elif len(postcode) < 5:
        return "00000"

    # return "00000" for postcodes that are outside the area
    elif postcode == "01125" or postcode == "20052" or postcode == "01238" or
postcode == "01240" or postcode == "01250":
        return "00000"
    else:
        return postcode
```

4. Audit ‘tourism’ tags

I used audit_tourism.py to find out what are the value types for tags with ‘tourism’ key and did not find anything problematic.

Overview of the data

1. size of the files

```
boston_massachusetts.osm: 414.7 MB
small_sample.osm: 9.3 MB
nodes.csv: 150.2 MB
nodes_tags.csv: 16.6 MB
ways.csv: 19.7 MB
ways_tags.csv: 20.7 MB
ways_nodes.csv: 50.2 MB
```

2. Number of nodes and ways

```
sqlite> SELECT count(*) FROM nodes;
```

```
1933986
```

```
sqlite> SELECT count(*) FROM ways;
```

```
309389
```

3. Number of unique users

```
sqlite> SELECT count(*)
        FROM (SELECT COUNT(uid) FROM (SELECT uid FROM nodes UNION ALL
          SELECT uid FROM ways) GROUP BY uid) as all_uid;
```

```
1339
```

4. Top 10 contributing users

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

```
crschmidt           |  1202522
jremillard-massgis  |   429870
OceanVortex         |    92041
wambag              |    80100
morganwahl          |    69511
ryebread            |    67015
MassGIS Import      |    63230
ingalls_imports     |    32461
Ahlzen              |    27132
mapper999           |    14960
```

5. Top 10 tourism types

```
sqlite> SELECT all_tags.value, COUNT(*)
        FROM (SELECT key, value FROM nodes_tags UNION ALL SELECT key, value
         FROM ways_tags) as all_tags
        WHERE all_tags.key = 'tourism'
        GROUP BY all_tags.value
        ORDER BY COUNT(*) DESC
        LIMIT 10;
```

```
hotel          |  101
museum         |   56
artwork        |   53
attraction     |   33
viewpoint      |   31
picnic_site    |   27
information    |   25
guest_house    |    8
hostel         |    4
motel          |    3
```

6. Top 10 values and counts in leisure tag and tourism tag that the top contributor had curated

```
sqlite> SELECT ways_tags.value, COUNT(*)
        FROM ways_tags, (SELECT user, uid FROM ways WHERE user = 'crschmidt') as crs
        WHERE ways_tags.key = 'leisure' OR ways_tags.key = 'tourism'
        GROUP BY ways_tags.value
        ORDER BY COUNT(*) DESC
        LIMIT 10;
```

```
pitch               |  842240
park                |  825600
playground          |  448000
recreation_ground   |  410880
nature_reserve      |  142080
swimming_pool       |   69120
hotel               |   62720
garden              |   52480
museum              |   28160
sports_centre       |   25600
```

7. What is the maximum number of nodes required for way elements with the top ten tourism tags?

```
sqlite> SELECT ways_tags.value, MAX(ways_nodes.position) as largest_ways, COUNT(*)
        FROM ways_tags, ways_nodes
        WHERE ways_tags.id = ways_nodes.id AND ways_tags.key = 'tourism'
        GROUP BY ways_tags.value
        ORDER BY count(*) DESC
        LIMIT 10;
```

```
hotel         | 120   | 897
attraction    | 176   | 345
museum        |  48   | 303
zoo           | 155   | 156
picnic_site   |  60   |  78
information   |  16   |  60
aquarium      |  23   |  29
artwork       |  12   |  28
guest_house   |  10   |  25
viewpoint     |  12   |  22
```

Other ideas about the dataset

1. More on curating the street names – street number in front of the street name

```
sqlite> SELECT value
        FROM (SELECT key, value, type from nodes_tags UNION ALL SELECT key, value,
          type FROM ways_tags)
        WHERE key = 'street' AND type = 'addr'
        ORDER BY value
        LIMIT 10;
```

```
1 Kendall Square
1629 Cambridge Street
3rd Street
5th Street
61 Union Square
738 Commonwealth Avenue
A Street
A Street
A Street
A Street
```

I notice that in addition to having street numbers after the street names, there were also a few with the street number inserted before the street names. A minor modification to clean_street_name.py should take care of cleaning these street names.

2. Looking at 'addr:street' audit as a case study for the Boston OSM quality

```
sqlite> SELECT count(*)
        FROM (SELECT * FROM ways_tags UNION ALL SELECT * FROM nodes_tags)
        WHERE key = 'street' AND type = 'addr';
```

```
5998
```

From audit_street_name.py there were 67 street names that needed cleaning, which was 1% of the total 5998 tags with 'addr:street'. The result suggests that the quality of data imported is pretty good.

3. Looking at 'museum' tag as a case study for the Boston OSM quality

The 'museum' tags are present in both nodes_tags and ways_tags.

```
sqlite> SELECT nodes_tags.value
        FROM nodes_tags, (SELECT DISTINCT(id) FROM nodes_tags WHERE
          value='museum') as museum
        WHERE nodes_tags.id = museum.id AND nodes_tags.key = 'name';
```

```
Somerville Historical Museum
Museum of Transportation
Captain Robert Bennet Forbes House
Forbes House Museum
Peabody Museum of Archaeology and Ethnology
New England Sports Museum
MIT Museum
Harvard University Museum of Comparative Zoology
Pierce/Hichborn House Museum
Harvard University Museums of Natural History
Winthrop Public Library and Museum
Alexander Graham Bell Room Museum
Boston Fire Museum
Ames Mansion Museum
Burage Mansion Museum
Boston Marine Society Museum
Harvard University Geological and Mineral Museum
First Corps of Cadets Museum
Josiah Quincy House Museum
Nichols House Museum
Gibson House Museum
Ancient and Honorable Artillery Company Museum
Busch-Reisinger Museum
Boston Children's Museum
Hayden Planetarium
Dallin Art Museum
Griffin Museum at Digital Silver Imaging
Fogg Museum
Arthur M. Sackler Museum
Warren Anatomical Museum
Eustis Estate Museum and Study Center
USS Constitution Museum
The Sports Museum
```

```
sqlite> SELECT ways_tags.value
        FROM ways_tags, (SELECT DISTINCT(id) FROM ways_tags WHERE
          value='museum') as museum
        WHERE ways_tags.id = museum.id AND ways_tags.key = 'name';
```

```
Museum of Science
The Battle of Bunker Hill Museum
Old State House
Old South Meeting Place
Boston Fire Museum
Isabella Stewart Gardner Museum
Cooper-Frost-Austin House
Armenian Library and Museum of America
Longfellow Natl Historic House
Dorothy Quincy House
Longyear Museum
Semitic Museum
Jason Russell House
Abigail Adams House Museum
Smith Museum Archives
The Institute of Contemporary Art
MGH Museum of Medical History and Innovation
Adams National Historical Park
Armory Museum of the Ancient and Honorable Artillery Company
Paul Revere House
```

From looking at the list of museums, I noticed that Museum of Fine Arts, a prominent tourism spot in the area, is missing, which suggests that the list of museums is incomplete. Additionally, although the Hayden Planetarium is a part of Museum of Science, it is annotated as a node tag and not associated with Museum of Science, which is a way tag.

```
sqlite> SELECT *
        FROM nodes_tags, (SELECT id FROM nodes_tags WHERE value = 'Hayden
          Planetarium' GROUP BY id) as hayden
        WHERE nodes_tags.id = hayden.id;
```

```
367781422|ele|4|regular|367781422
367781422|name|Hayden Planetarium|regular|367781422
367781422|source|USGS Geonames|regular|367781422
367781422|tourism|museum|regular|367781422
367781422|state|MA|addr|367781422
367781422|reviewed|no|gnis|367781422
367781422|feature_id|600091|gnis|367781422
367781422|county_name|Suffolk|gnis|367781422
367781422|import_uuid|57871b70-0100-4405-bb30-88b2e001a944|gnis|
          367781422
```

This type of missing or incorrect information is harder to target in a systemic cleaning effort. One way to find out and curate it would be to incorporate ‘user experience’ into updating map information in

which a person that actually uses the map and living or sightseeing the area report back errors or discrepancies.

There are a few ways to obtain input from these users. For example, a hashtag #bostonOSM could be created for people to include when they make comments on social media networks. The hashtag can serve as a bait for the admins to find out what are created to be curated. Alternatively, a user interface can be incorporated to the OSM website so that when a user looks for a piece of information and didn't find something satisfying, they can immediately report back to the website. Both ways can generate a log of problems for curation.

An advantage of generating a user interface over hashtag method is that additional effort can be put in for the users to update the information themselves, thus leaving less work for the admins to do the actual curation. The proposed update by the user can be validated first by the admin before publication on the website to make sure that the information is accurate.

This type of curation can be done over a long period of time or by an annual competition held over a weekend. Constant curation is required because it's foreseeable that things can change over time (eg. new parks in the neighborhood, restaurants going out of business, etc.).

One anticipated problem using these methods is that for areas that don't attract a lot of visitors, it would be harder to generate 'user experiences'. With the increasing interest in adding coding into the school curriculum, one way to overcome this could be to set up outreach projects for local school students to do map curation of their hometown.

Conclusion

After this initial review I would conclude that the Boston OSM is quite complete with only few errors left for systemic fix. However, there might still be large quantity of detailed information missing. To connect the dots, I would suggest incorporation of user experience to manually look at the missing details.