Area Dataset - Boston, Massachusetts

- boston_massachusetts.osm ... 211 MB
- boston_massachusetts.json ... 245 MB

Boston, MA

Boston is the capital city and the most populous municipality of the Commonwealth of Massachusetts, USA. With population over 670,000 in 2016, it is the largest city in the New England region of the Northeastern United States. Founded in 1630 by Puritan settlers from England, it is considered as one of the oldest cities in the country. With universities like MIT, Harvard University, Berklee College of Music, University of Massachusetts etc., Boston is considered as an international center of higher education and also a world leader in innovation and entrepreneurship with nearly 2,000 start-ups.

Why Boston?

For a long time, I have wished to go to Boston, and I'm taking my first step towards knowing the place, by choosing this particular dataset to learn more about it before hand.

Note:

The dataset can be found via this link: https://mapzen.com/data/metro-extracts/metro/boston_massachusetts/. The original dataset is over 400 MB and unfortunately my computer wasn't able to process the file of this magnitude, so I had to trim down the dataset to about 200 MB for it to function properly.

Skimming through it, I encountered few problems like:

- How to better understand and clean up the dataset of such large size.
- Several inconsistencies in zip codes, like the ones with more than five digits "02110-1301" or zip codes out of Boston area.
- Abbreviation of some street types, like "Ave" for Avenue, "St" for Street etc.
- The "address" section for some documents might differ from others, for instance, one way of representing the address in the documents is:

```
<tag k="address" v="40 Thorndike St, Cambridge, MA, 02141" />
```

having the entire address in one line. But some documents have separate entities describing an address like:

```
<tag k="addr:city" v="Hyde Park" />
<tag k="addr:street" v="Metropolitan Avenue" />
<tag k="addr:postcode" v="02136" />
<tag k="addr:housenumber" v="655" />
```

Having an address described the way above, makes it easy to go after cities, street or postcode separately. But with one-liner address, it might be difficult to get an accurate count, say most mentioned street or cities. So, I'm going to identify the zip code from the address part and save it separately like so, "address": {"location": "40 Thorndike Street,

Cambridge, MA 02141", "postcode": "02141"}, to easily analyze all the zip codes in one go.

After the initial observations, my next steps would be to:

- To count the tags, like the number of nodes, ways etc.

Query Snippet	Result
`for event, element in ET.iterparse(filename, events = ("start",)):` `  key = element.tag` `  if key in tag_dictionary:` `  tag_dictionary[key] += 1` `  else:` `  tag_dictionary[key] = 1`	Nodes = 976373 Ways = 155519 Relations = 663

- Identify any problem characters

Query Snippet	Result
`lower = re.compile(r'^([a-z]|_)*$')` `lower_colon = re.compile(r'^([a-z]\|_)*:([a-z]\|_)*$')` `problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')` `def key_type(element, keys):` `  if element.tag == "tag":` `  k_value = element.attrib['k']` `  if lower.search(k_value):` `  keys["lower"] += 1` `  elif lower_colon.search(k_value):` `  keys["lower_colon"] += 1` `  elif problemchars.search(k_value):` `  keys["problemchars"] += 1` `  else:` `  keys["other"] += 1` `  return keys`	problemchars -- 3 lower -- 398383 other -- 19787 lower_colon -- 37812

- Clean up the dataset, convert to JSON and load the data to mongoDB:

Query Snippets
`def street_cleanup(street):` `  name_split = street.split(' ')` `  for key in mapping:` `  for word in name_split:` `  if key == word:` `  street = re.sub(r'\b%s[.,\b]?' %key, mapping[key], street)` `  return street` `def zip_code_setup(address):` `  zip_code_1 = re.compile("MA, ")` `  zip_code_2 = re.compile("MA ")` `  zip_pos = 10000` `  if zip_code_1.search(address):` `  zip_pos = zip_code_1.search(address).start()+4` `  if zip_code_2.search(address):` `  zip_pos = zip_code_2.search(address).start()+3` `  zip_code = address[zip_pos:].strip(' ')` `  return zip_code`

```
# This condition will perform any kind of clean up related to street name abbreviation based on the
# "mapping" dictionary provided, like converting "St." to "Street" or "Ave" to "Avenue"
if k_attrib == "addr:street":
    value = street_cleanup(value)

# If the regexes don't match with the attributes
if addr_pattern.match(k_attrib) == None and colon_pattern.match(k_attrib) == None:

    # The following condition is to check for the attributes stored as "address" instead of "addr:",
    # like <tag k="address" v="888 Broadway, Everett MA 02149-3199" />, here we save the postcode as
    # a separate entity
    if k_attrib == "address":
        node['address']['location'] = value
        zip_code = zip_code_setup(value)
        if len(zip_code) >= 5:
            node['address']['postcode'] = zip_code
```

Result

{"amenity": "place_of_worship", "gnis:state_id": "25", "name": "Quincy Point Congregation Church",
"created": {"uid": "2063579", "changeset": "22175812", "version": "4", "user": "JGS12", "timestamp": "2014-05-06T21:08:19Z"},
"pos": ["42.2488205", "-70.983316"], "ele": "12", "religion": "christian", "node_type": "node", "node_id": "358279337",
"address": {"city": "Quincy", "street": "Washington Street", "housenumber": "444", "postcode": "02169"},
"gnis:county_id": "021", "gnis:created": "01/15/2003", "gnis:feature_id": "1974492", "denomination": "Congregational"}

Loading data into MongoDB:

```
\3.4\bin>mongoimport -d osmstreetmap --collection boston_massachusetts --type json --file C:\Users\unayak2\Desktop\boston_massachusetts.json
connected to: localhost
[###.....................] osmstreetmap.boston_massachusetts    41.7MB/256MB (16.3%)
[#######.................] osmstreetmap.boston_massachusetts    84.1MB/256MB (32.9%)
[###########.............] osmstreetmap.boston_massachusetts    127MB/256MB (49.6%)
[###############.........] osmstreetmap.boston_massachusetts    169MB/256MB (66.1%)
[###################.....] osmstreetmap.boston_massachusetts    211MB/256MB (82.4%)
[########################] osmstreetmap.boston_massachusetts    256MB/256MB (100.0%)
imported 1132555 documents
```

- Calculate the total number of documents

Query Snippet	Result
`# Total no. of documents` `total = db.boston_massachusetts.find().count()` `print "Total no. of documents:", total`	Total no. of documents: 1132555

- Number of unique contributors

Query Snippet	Result
`# No. of unique contributors` `unique_contributors = len(db.boston_massachusetts.distinct("created.uid"))` `print "No. of unique contributors:", unique_contributors`	No. of unique contributors: 1197

- Users with highest contributions

Query Snippet	Result

```
db.boston_massachusetts.aggregate([{"$group" : {"_id" : "$created.user",
                                                "contributions" : {"$sum" : 1}}},
                                   {"$sort" : {"contributions" : -1}},
                                   {"$limit" : 10}])
```

```
DOCS SUBMITTED, (contribution %ge) -->  USERS
-----------------------------------     -----
597677 , (52.77)                  -->    crschmidt
214329 , (18.92)                  -->    jremillard-massgis
55587 , (4.91)                    -->    wambag
45447 , (4.01)                    -->    OceanVortex
33690 , (2.97)                    -->    morganwahl
32991 , (2.91)                    -->    ryebread
29291 , (2.59)                    -->    MassGIS Import
16220 , (1.43)                    -->    ingalls_imports
14125 , (1.25)                    -->    Ahlzen
7363 , (0.65)                     -->    mapper999

TOTAL DOCS SUBMITTED BY THE TOP 10 USERS, (contribution %ge)
---------------------------------------------------------------
1046720 , (92.42)
```

- Users with 10 or less contributions

Query Snippet	Result

```
db.boston_massachusetts.aggregate([{"$group" : {"_id" : "$created.user",
                                                "contributions" : {"$sum" : 1}}},
                                   {"$group" : {"_id" : "$contributions",
                                                "users_count" : {"$sum" : 1}}},
                                   {"$sort" : {"_id" : 1}},
                                   {"$limit" : 10}])
```

```
DOCUMENTS --> NO. OF USERS, (%ge of users)
---------     ----------------------------
1         -->         338 , (28.24)
2         -->         128 , (10.69)
3         -->         88 , (7.35)
4         -->         60 , (5.01)
5         -->         50 , (4.18)
6         -->         36 , (3.01)
7         -->         24 , (2.01)
8         -->         23 , (1.92)
9         -->         20 , (1.67)
10         -->         15 , (1.25)

TOTAL, (%ge of users with 10 docs or less)
------------------------------------------
782 , (65.33)
```

- Drilling down on the zip codes, we'll examine if the zip codes are valid and whether they all belong to the Boston area. And the most used zip code:

Result

```
Documents with "postcode" in the "address" section: 1593
Documents with "postcode" in the Boston area: 833
Documents with "postcode" outside the Boston area: 760

Unique postcodes in the documents: 75
All of the Boston area postcodes: 62
Postcodes from the documents belonging to the Boston area: 40
Postcodes not belonging to the Boston area: 35

10 most mentioned zip codes:
----------------------------
02139  -->  218
02135  -->  161
02130  -->  105
02144  -->  63
02474  -->  63
02114  -->  49
02215  -->  46
02116  -->  43
02143  -->  40
02138  -->  39
```

- Most mentioned cities

Query Snippet	Result
`db.boston_massachusetts.aggregate([{"$match" : {"address.city" : {"$exists" : True}}},` `{"$group" : {"_id" : "$address.city",` `"count" : {"$sum" : 1}}},` `{"$sort" : {"count" : -1}},` `{"$limit" : 10}])`	`10 Most mentioned cities` `----------` `Boston --> 572` `Cambridge --> 287` `Malden --> 191` `Arlington --> 136` `Somerville --> 134` `Jamaica Plain --> 50` `Chelsea --> 37` `Quincy --> 28` `Medford --> 25` `Brookline --> 22`

- List of amenities and their count

Query Snippet	Result
`db.boston_massachusetts.aggregate([{"$match" : {"amenity" : {"$exists" : True}}},` `{"$group" : {"_id" : "$amenity",` `"count" : {"$sum" : 1}}},` `{"$sort" : {"count" : -1}},` `{"$limit" : 10}])`	`LIST OF AMENITIES AND COUNT:` `----------` `parking --> 742` `bench --> 527` `restaurant --> 376` `school --> 339` `parking_space --> 224` `place_of_worship --> 212` `library --> 157` `bicycle_parking --> 155` `cafe --> 151` `fast_food --> 127`

Ideas for improvement:

- For addresses, there should be one standard way to store the data, like the following:

```
<tag k="type" v="multipolygon" />
<tag k="building" v="commercial" />
<tag k="addr:city" v="Boston" />
<tag k="addr:state" v="MA" />
<tag k="addr:street" v="Summer Street" />
<tag k="building:height" v="34.1376" />
<tag k="building:levels" v="10" />
<tag k="addr:housenumber" v="280" />
```

This way of representing the address is quite convenient, not only it details everything clearly but also it would be easy to examine each of the entities separately.

- It would be a good addition to have the date or year of establishment for buildings, universities or libraries or places of interest or historical value.
- Also, adding the capacity attribute to restaurants, buildings or necessary public places.

  Although it might be difficult for some to gather such minutiae details, and ofcourse to edit the already existing data, but these improvements might help people looking for specific data at times.

Conclusion:

The dataset is quite well maintained for most part and it was fairly interesting & challenging to work with data of such large scale and examining the outcomes. There's always a room to improve the quality of content for hightened accessbility even by a novice.

Before this lesson, I had little to no idea what OpenStreetMap was but these lessons along with this project have got me several steps closer to understanding the fundamentals of this phenomenal opensource collaborative mapping service. OpenStreetMap today has over 2 million registered users (Res: OpenStreetMap Wiki) and with more awareness, people all over the globe can have the knowledge to successfully build the map of the world.

Resources used:

- https://en.wikipedia.org/wiki/OpenStreetMap
- https://api.mongodb.com/python/2.0/tutorial.html
- https://docs.mongodb.com/manual/reference/method/
- https://en.wikipedia.org/wiki/Boston