Data Science Nano Degree –

Project 1 –

Data Wrangling with Mongo DB

Project Report

# Table of Contents

Table of Contents

[Table of Contents 1](#_Toc417684927)

[1. The Map Extract 2](#_Toc417684928)

[1.1 Initial Analysis of XML Data 2](#_Toc417684929)

[1.2 Reshaping Map Data 3](#_Toc417684930)

[1.3 Identification of Data Problems 3](#_Toc417684931)

[2. Data Over*v*iew and Analysis 6](#_Toc417684932)

[3. Additional Ideas 8](#_Toc417684933)

[4. Conclusion 9](#_Toc417684934)

# The Map Extract

The map area extracted is for a rectangle enclosing Bangor, Maine in the northwest and Mount Desert Island in the southeast. Specifically, the bounding box for the extract encloses latitude from 44.2083 to 44.8948, and longitude from -68.9063 to -67.9999. My original plan was to analyze data for Mount Desert Island, a large costal island just of the coast of Maine, and home to Acadia National Part, and towns including Bar Harbor and Southwest Harbor, and a personal favorite of mine for vacations. The extract for this area, though, was less than 16 MB, so the rectangle was increased to include the city of Bangor, about 40 miles northwest of Mount Desert Island. The final extract file size was 57 MB.

The extract was downloaded from <http://overpass-api.de>.

## Initial Analysis of XML Data

Some quick measurements of the map extract were generated using the python code written for Lesson 6, Quiz 1. A count of XML element types encountered was determined with the following results:

|  |  |
| --- | --- |
| **Element type** | **Count** |
| bounds | 1 |
| member | 6341 |
| meta | 1 |
| nd | 288171 |
| node | 266323 |
| note | 1 |
| osm | 1 |
| relation | 139 |
| tag | 151668 |
| way | 17556 |
| Total number of tags | 730202 |

Of these, we are interested only in the following:

* Node – encodes a location. Child of the top level “OSM” element
* Way – encodes extended line or area. Child of the top level “OSM” element
* Type – encodes a keyword/value pair attached to the enclosing “node” or “way” element
* Nd – encodes the id for a node associated with the enclosing “way” element(s) – segmented line or polygon.

A number of quality checks were run on the map extract

|  |  |  |
| --- | --- | --- |
| Data quality check | Results | Comments |
| Do all <tag> elements contain exactly 1 ‘k’ and one ‘v’ attribute? | All <tag> elements have both a ‘k’ and a ‘v’ attribute |  |
| Are all postcodes (ZIP codes) legitimate for this area, and are all 5 digit strings? | Found:   * Zip+4 code * ZIP preceded by “ME “ * ZIP legal but incorrect (e.g. ZIP code is for area outside of map | Very few addresses contain zip codes.  Difficult to assure ZIP code is the right one without having a validated “gold” source for data. |
| Are all street names spelled out (no abbreviations) | Various abbreviations used for street type | Corrected in MapCleaner script. |
|  |  |  |
|  |  |  |
|  |  |  |
| Are there any keywords with problematic characters | No illegal characters were found in keywords (!!!). |  |
| Some amenities (such as restaurants, cafes, etc.) should have a name specified | Three locations were identified as restaurants, but had no name specified. | These were actually detected while running Mongo queries. These records were manually removed from the DB using the Mongo shell. If this had been a more wide-spread problem, it could have been coded into the MapCleaner script. |
| Extraneous data in fields | One case if addr:city that contained comma and state code | For the limited cleaning needed in this data, the MapCleaner removes the comma and anything following it. |

## Reshaping Map Data

Following the model of the Lesson 6 quizes, we will be mapping the OSM XML structure into a collection of JSON objects where the dictionary keword/value pairs are determined by the OSM k=\* and v=\* fields. We will also be following the course instructions to reshaping the data in the following ways:

* “Addr:\*” fields will be collected into a “address” substructure
* Several k/v pairs from the <node> and <way> elements will be collected into a “created” substructure
* The “lat” and “lon” attributes of the <node> elements will be combined into a list containing the two values.

## Identification of Data Problems

The map area was audited using a Python program (based on and extending the code created for Lesson 6 problems) to collect statistics to help identify problem entries. This includes:

* the set of unique values for various keywords, so that they could be easily evaluated for quality,
* geographic information that could be assessed independently – e.g. addr:city, addr:state, addr:postcode – these can be evaluated in terms of format, and accuracy of content.
* counts of unique values for keywords that seem likely to have lots of user generated values: “amenities”, “cuisine”, “religion”, “tourism”
* counts of unique values for several “tiger:” fields. tiger:county, tiger:zip\_left\_1, tiger:zip\_right\_1 (to evaluate the value of using tiger:data as a basis for filling in missing fields).

A number of problems were found. The list below includes a description of each problem, *the corrective action to be takes, or the reason for not making corrections*, and the aspect of data quality (Validity, Accuracy, Completeness, Consistency, and Uniformity) involved.

* Address subfields: House/building addresses should contain the following subfields:
  + Housenumber, street, city, postcode, state, country
  + (The city, postcode, state, and country fields may be omitted if the location is within a boundary box that identifies these fields). There are many examples of missing fields. *Add boundary ways around all cities, counties, states, and postcode areas to associate values with all objects within the boundaries. This was not done due to apparent lack of open data. Boundaries for cities in this map extract are present in the downloaded extract, based on the US Census Tiger shapefile data. This data is not in MongoDB*, as it is captured in <relationship> elements, which we are not using for this project. (Completeness)
* Addr:city: City names generally don’t contain punctuation. Issue: one case of addr:city populated with appended comma and state name. *Remove extraneous state from addr:city names.* (Uniformity)
* addr:state – For state names, the ISO 3166-2 abbreviations should be used. In the area being audited, any state fields should contain “ME”. Some entries contained the string “Maine”. *Translate alternate formats for addr:state entries to ISO 3166-2 region codes.* (Uniformity)
* addr:country – For this data extract, country fields should contain “US”. *Use standard ISO 3166-1 alpha-2 rather than ISO 3166-1 alpha-3 for addr:country* (Validity, Uniformity)
* addr:street – form of street field should use unabbreviated street type (e.g. Street, Avenue, etc.). *All street types will be spelled out, with initial capitalization.* (Uniformity)
* addr:postcode – postcodes (Zip codes) should be 5 digits. Codes should correspond to ZIP code areas within the map area. Some entries incorrectly were prefixed with the state (“ME “), were XIP+4 codes, or had a code that was not appropriate for the location. *Eliminate extra pre-fixed or post-fixed characters, and drop verifiably incorrect code*. (Uniformity, Accuracy)
* cuisine –
  + The normal format is lower case, words separated by an underscore. *Map all entries to lower case.*(Uniformity)
  + In OSM XML format, multiple values are separated by a semicolon. For the JSON format to load in Mungo DB, we will use a list of values. *Map all entries to lower case, split value string separated by one or more separator characters (e.g., comma, whitespace, semicolon) into a list.* (Uniformity)

Finally, there are problems with completeness of the map data that will be described in the “Additional Thoughts” section below.

# Data Over*v*iew and Analysis

Once the data was examined and the problems to be fixed identified and remediated, the XML data was recast a JSON data set, which was loaded into a MongoDB instance. A series of queries was run to characterize the contents of the map. The following table summarizes this data. Note – map data was loaded into the ‘maine’ collection.

In order to process geospatial queries, a geospatial index had to be added to the collection before querying:

db.maine.createIndex( { pos: ‘2dsphere’ } )

**Table of Mongo queries and results**

|  |  |  |
| --- | --- | --- |
| **Description** | **Query** | **Result** |
| Number of documents | db.maine.count() | 283879 |
| Number of nodes | db.maine.find( {"type": "node"} ).count() | 266321 |
| Number of ways | db.maine.find( {"type":"way"} ).count() | 17551 |
| Number of unique contributors | db.maine.distinct("created.user").length | 250 |
| Name and reference count of most referenced street names | db.maine.aggregate( [  {'$match' :  {address.street : {'$exists' : 1}}},  {'$group' :  {'\_id' : '$address.street, 'count' : {'$sum' :1}}},  {'$sort' : {'count' : -1}},  {'$limit' : 10}  ]) | { "\_id" : "West Main Street", "count" : 22 }  { "\_id" : "Main Street", "count" : 14 }  { "\_id" : "Broadway", "count" : 14 }  { "\_id" : "State Street", "count" : 7 }  { "\_id" : "Cottage Street", "count" : 6 }  { "\_id" : "Main Road North", "count" : 4 }  { "\_id" : "High Street", "count" : 4 }  { "\_id" : "Church Street", "count" : 3 }  { "\_id" : "Bar Harbor Road", "count" : 2 }  { "\_id" : "Longview Drive", "count" : 2 } |
| Locations on “Cottage Street” | db.maine.find(  {‘address.street’: ‘Cottage Street’},  {‘name’:1, ‘\_id’:0}) | { "name" : "Acadia Bike Rentals" }  { "name" : "Bar Harbor Bicycle Shop" }  { "name" : "Hannaford's Pharmacy" }  { "name" : "Two Cats Cafe" }  { "name" : "Two Cats Inn" }  { "name" : "Hannaford's Supermarket" } |
| Restaurants on “Cottage Street” | db.maine.find(  {‘address.street’: ‘Cottage Street’,  ‘amenity’: ‘restaurant’},  {‘name’:1, ‘\_id’:0}) | { "name" : "Two Cats Cafe" } |
| Restaurants within 20 KM of Bar Harbor Village Green (Lat/Lon = [44.3879396, -68.2055718]) in increasing order of distance | db.maine.find(  { pos: { $near: {  $geometry: { type: ‘Point’,  coordinates:  [44.3879396, -68.2055718],  $minDistance: 10,  $maxDistance: 20000 } },  anemity: ‘restaurant’ },  { \_id: 0, name: 1} ) | { "name" : "Two Cats Cafe" }  { "name" : "Beal's Lobster Pound" }  { "name" : "Scotty's Dockside Motel & Restaurant" }  { "name" : "Red Sky" }  { "name" : "Grumpy's Breakfast" }  { "name" : "Sips" }  { "name" : "Gilley's Head Of The Harbor" }  { "name" : "Eat-A-Pita" }  { "name" : "Mainely Meat Barbecue" }  { "name" : "Thurston's Lobster Pound" }  { "name" : "Trenton Bridge Lobster Pound" } |

# 

# 3. Additional Ideas

From the viewpoint of a consumer of OSM data, e.g. as an input source for a location-aware mobile app, or as location data to be used in a larger analysis, there are several problems that must be dealt with.

**Multiple ways to specify the same data**

Some map features have more than one accepted forms. For example, a postal code can be entered as the value of ‘address.postcode=nnnn’ on a node, or as ‘name=nnnn’ on a way or relation with ‘boundary=postcode’ . This adds complexity in creating generally useable queries. The uer of the data must be aware of this kind of variation, and to deal with it for those properties that will be used in a given application.

**Impediments to assessing data quality and completeness**

While providing a great deal of flexibility in handling newly identified data attributes, the OSM model does create some obstacles to assessing and fixing data quality issues, as well as in formulating generalized data queries.

Key/value pairs can be added as the contributor sees fit. There is an expectation that the contributor will find and use existing keys and values where they are already available, but this is not directly enforceable. It is expected that keys and values that do not make sense will be reconciled eventually through community consensus. Again, it will be necessary for the user of such data to audit the properties used, and to deal with any problematic entries.

**Sparseness of location data**

For the map extract studied here, OSM data is sparse in several regards.

1. The actual number of amenities such as restaurants is a small fraction of what exists. Querying for restaurants in the town of Bar Harbor and surrounding communities yields a list of only 11 entries, while in fact this tourism oriented area has hundreds of such establishments. This is obviously an extremely difficult problem to solve, given the voluntary and informal contribution model. There is no obvious way to evaluate just how complete the data is, since there is no overarching control of what areas or establishments are to be included.
2. For locations that are in the map, many have very few attributes populated. As an extreme example, this map extract contains three separate entries with attribute “amenity: ‘restaurant’“ do not have a name populated.
3. There is a lack of uniformity in entering equivalent data. For example, a ZIP code (in the US) can be attached to the addr.postcode field, or the boundries of a ZIP code area can be represented as a ***way***, or a ***relation*** consisting of more than one ***way***. There are problems and benefits to each approach: attaching the ZIP to an attribute of a location makes ZIP based queries fast and simple, compared to turning each into a geo based query. On the other hand, entering the ZIP on each location created additional overhead in space and time, and also increases the chance for data error. In fact, most locations on the map extract used here do not have a ZIP code reported at all.

Alleviating the lack of data depends on encouraging users to make contributions based on their own observations and/or other existing data sources. One probable impediment is the lack of tools for easy data capture. Providing a smart phone app, for example, that allowed for capture of current location (lat/long) and provided a simple entry interface for the relevant data could encourage a higher level of participation by OSM users.

**Inaccurate data**

In some cases, map data is simply incorrect. This is difficult to detect, but can be seen in the map extracts in some cases. For example, the map extract used here contained a ZIP code (address.postcode) for a location on Mount Desert island, that was actually the code for a location some 50 miles further to the north-west, and outside of the boundaries of the map extract. Many such errors likely exist.

# 4. Conclusion

The use of OSM data requires attention to several aspects. The sparseness of the map extracts with regard to amenities limits the usability of the data as the basis of data analysis, which can only be alleviated by encouraging wider community involvement in contributing map data. The existence of multiple ways of coding a feature makes it difficult to create queries that will work across the ful set of map data. Finally, it is difficult to assess the accuracy of the data unless errors are inconsistent with other map data.