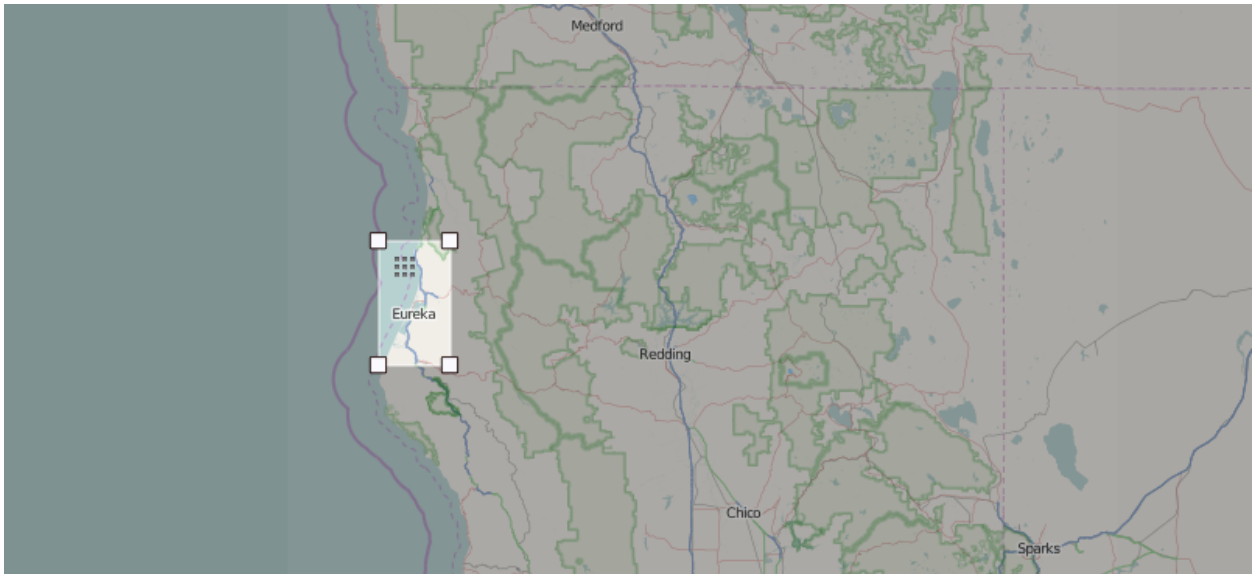
OpenStreetMap Data Wrangling Project with MongoDB

Kevin Palm

Introduction

This project pulled OpenStreetMap data from the Humboldt Bay area of Northern California. The dataset was downloaded from a mirror using Overpass API with the input of latitude 41.1851 through 40.5190, and longitude -124.4037 through -123.9062 (see Figure 1). The resulting dataset was 60.6 MB in raw XML format.

Figure 1: Area of Analysis



Problems with the Raw Dataset

Scanning through a dictionary of each unique key and the respective values occurring with that key in this dataset revealed a plethora of possibly incorrect tags. It is very likely that a programmer with more familiarity using OSM could greatly improve this dataset by consolidating redundant keys and rekeying incorrectly keyed values, but as a newcomer to OSM I focused instead primarily on consistency in formatting (unless with reference to the OSM Wiki I found blatantly erroneous usage of a key). Focusing on formatting also seemed to be more in line with the exercises and sample project. All the problems listed below were corrected programmatically.

Key:type Capitalization

Capitalized and uncapitalized entries were present in ‘type’ (e.g. 'public' and 'Public'). All entries were made lowercase (e.g. ‘public’).

Key:addr:state Capitalization

Capitalized and uncapitalized entries were present in ‘addr:state’ (e.g. 'Ca' and 'CA'). All entries were made uppercase (e.g. ‘CA’).

Key:fax Formatting

Multiple phone number formats were used in ‘fax’ (e.g. 'XXX-XXX-XXXX', '(XXX) XXX-XXXX', '(XXX)-XXX-XXXX', 'XXX.XXX.XXXX', '1-XXX-XXX-XXXX'). All entries were

changed to international format following the RFC 3966/NANP pattern[1] (e.g. +1-XXX-XXX-XXXX).

Key:phone Formatting

Same case as in ‘fax’. There were also values of ‘Yes’, rather than a listed number. Such values were removed.

Key:cuisine Capitalization and Formatting

There were inconsistencies in ‘cuisine’ (e.g. ‘pizza’ and ‘_pizza’, ‘american’ and ‘Family Style American’). Capitalization was removed, spaces were replaced with underscores, and commas were replaced with semi-colons (e.g. ‘pizza’ and ‘family_style_american’).

Key:brand Capitalization

Not all ‘brand’ names were capitalized (e.g. ‘subaru’). These names were capitalized (e.g. Subaru).

Key:addr:street Abbreviations and Incorrect Values

There were inconsistencies in abbreviations in ‘addr:street’ (e.g. ‘St’ and ‘Street’). Some values were not actually street names (e.g. ‘Nw Cnr Trinidad Int’ and '1835 6TH Street'). Abbreviations for types of streets were changed to full words (e.g. ‘Street’). Specific fixes were applied from information gathered using OpenStreetMap and Google Maps, and information in the value (e.g. ‘Nw Cnr Trinidad Int’ became ‘Patricks Point Road’ and '1835 6TH Street' became ‘6th Street’).

Key:natural Incorrect Value

There was a web URL value found in ‘natural’. The incorrect value was removed.

Key:route Incorrect Values

There were non access related values in ‘route’ (e.g. ‘299’ and ‘101’). The incorrect values were removed.

Key:addr:city Formatting

There were dissimilar values for the same cities (e.g. ‘Arcata,’, ‘Arcata’, and ‘Arcata, CA’). Each was changed to its simplest complete name (e.g. ‘Arcata’).

---

[1] From http://wiki.openstreetmap.org/wiki/Key:phone#Usage

Key:is_in:state Abbreviations

There were inconsistencies in abbreviations in ‘is_in:state’ (e.g. ‘CA’ and ‘California’). States were changed to their abbreviated forms (e.g. ‘CA’).

Overview of the Data

The JSON generated containing the corrected ‘node’ and ‘way’ elements and their sub-elements was 63.5 MB in size. Some basic statistics about the dataset:

Number of Documents:

```
> db.project2.count()
305134
```

Number of Nodes:

```
> db.project2.count({"type":"node"})
292292
```

Number of Ways:

```
> db.project2.count({"type":"way"})
12839
```

Number of Unique Users:

```
> len(db.project2.distinct('created.user'))
166
```

Unique Towns Tagged:

```
> db.project2.distinct('address.city')
['Eureka', 'Arcata', 'Trinidad', 'Bayside', 'Fortuna', 'Loleta', 'McKinleyville']
```

Earliest Timestamp in Dataset:

```
> stringdates = db.project2.distinct('created.timestamp')
> listdt = []
> for item in stringdates:
>       listdt.append(datetime.datetime.strptime(item, "%Y-%m-%dT%H:%M:%SZ"))
> min(listdt)
2007-10-24 01:45:48
```

Additional Ideas

Skewed Contributor Statistics

The same phenomenon with the sample data was existent in this dataset. 44% of the entire dataset was created by the top contributing user, 82% was created by the top two contributing users, and 92% was was the work of the top ten contributors. The word 'bot' appeared in top contributor user names in this dataset as well.

'Empty' Elements

Currently, a huge proportion of elements in this data set are 'empty' - containing nothing but 'id', 'lat', 'lon', 'version', 'timestamp', 'changeset', 'uid' and 'user' fields. Going off the idea that the vast majority of this dataset was created by map editor bots, consequently the vast majority of this dataset is 'empty'. I do not understand the usefulness of including free-floating GPS points in the dataset without any indication of what they reference. (Unless they are there to provide coordinates for inputting data, so as to escape incorrect values from lower quality GPS devices by users. In either case, for purposes of data analysis, and empty nodes seem useless.) I think that a reasonable first step to any study analyzing this data would be to remove these free-floating points from the dataset, and as a result it seems that OSM could save a lot of data transfer resources by offering datasets already free of them.

Redundant Tag Keys

This dataset seems to contain a lot of redundant keys, some sanctioned by the OSM Wiki, some created erroneously by users. A good example of this is the 'is_in' keys. The wiki points out that, even if the data is redundant, it can allow simpler searching and easy disambiguation between two similar objects[2]. Redundancy is completely understandable for such a project created by many different people of a long period of time. However, redundancy creates the problem that a querier cannot be certain that he or she is utilizing all of the data until each such key has been aggregated. If, for instance, and element has an 'is_in' value containing the city, but not an 'addr:city' value, a researcher could miss a data point by using the wrong key to query. As a result, it seems the the basic OSM data shape is not optimal for attempting comprehensive analysis - trying to get a comprehensive snapshot in time would require all redundant keys aggregated or their values shared across each redundant key.

Ideas for Improving the Dataset

I think it would create an improvement on the datasets produced if OpenStreetMap required that each data element be assigned at least one tag at the time of creation. I also think that it would be good if it were difficult for users to create a new key (one not well established through use and the Wiki) when tagging. Both changes could probably be achieved easily in the OSM app with programmatic checks and extra menus.

---

[2] http://wiki.openstreetmap.org/wiki/Key:is_in

In downloading the data, it would be nice to be able to pull data from more complex geometries than a square. Since OSM offered data downloads in shapefile format, however, this problem would be easily fixed by first tailoring the dataset in QGIS or some other GIS program before generating a JSON file.

Finally, the dataset used in this project seems to shine a light on how sparsely updated the Humboldt Bay area is in OpenStreetMap. Presumably, this is a problem in many rural areas. A lot more information is already available elsewhere on the internet - business websites, government websites, and online journals. The City of Arcata hosts a shapefile of all the addresses inside its boundaries, for instance - it's be extremely easy to update all the Arcata gps points inside this .OSM with address information. Webcrawlers or some other kind of data wrangling technique designed at populating elements with tags would probably also go a long way in getting this dataset beyond mostly empty elements. No doubt, however, with machine generated tags purely with information on the internet there would be plenty of errors.

Additional Data Exploration using MongoDB

Most Common Leisure Tags

```
> pprint.pprint(list(db.project2.aggregate([{"$group":{"_id":"$leisure", "count":{"$sum":1}}},
{"$sort":{"count":-1}}, {"$skip": 1}, {"$limit":3}])))
[{'_id': 'park', 'count': 83},
 {'_id': 'pitch', 'count': 37},
 {'_id': 'playground', 'count': 8}]
```

address.city Tags per Town

```
> pprint.pprint(list(db.project2.aggregate([{"$group":{"_id":"$address.city",
"count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$skip": 1}])))
[{'_id': 'Eureka', 'count': 407},
 {'_id': 'Arcata', 'count': 12},
 {'_id': 'Loleta', 'count': 10},
 {'_id': 'Fortuna', 'count': 5},
 {'_id': 'Trinidad', 'count': 3},
 {'_id': 'McKinleyville', 'count': 2},
 {'_id': 'Bayside', 'count': 1}]
```

Cuisine by Town

```
> pprint.pprint(list(db.project2.aggregate([{"$group":{"_id":"$address.city",
"cuisine":{"$addToSet": "$cuisine"}}}, {"$limit": 7}])))
[{'_id': 'McKinleyville', 'cuisine': []},
 {'_id': 'Fortuna', 'cuisine': []},
 {'_id': 'Bayside', 'cuisine': ['coffee_shop']},
 {'_id': 'Trinidad', 'cuisine': ['american']},
 {'_id': 'Loleta', 'cuisine': []},
```

{'_id': 'Arcata',
 'cuisine': ['mexican',
             'mediterranean',
             'donuts;pizza;ice_cream;sandwiches']},
{'_id': 'Eureka',
 'cuisine': ['bagel',
             'thai',
             'pizza;seafood',
             'sushi',
             'burger',
             'asian',
             'pizza',
             'mexican',
             'italian',
             'seafood',
             'american',
             'regional']}]

Conclusions

The Humboldt Bay Area is relatively rural, so this dataset is pretty sparse. While it's still interesting to look at what has and hasn't been tagged in this area, this dataset does not seem to contain remotely enough data to back a comprehensive study.

For this exercise, the dataset has been well cleaned of formatting inconsistencies and nested in a logical style when converted to JSON format, making queries upon it much more effective. As a newcomer to OpenStreetMap, the amount of content-based alterations that I did make (primarily removing values which were not entered under an appropriate key) already feels like I've overstepped my familiarity. However, I feel that for this dataset to be fully useful even more data reshaping is needed. Either a strict understanding of 'good' OpenStreetMap form or a more specific research aim would be necessary to determine how best to reshape the data, so for this exercise I feel that the data's current form is as good as possible.