OpenStreetMap Sample Project
Data Wrangling with MongoDB

Allan Jacobs

Map Area: Los Angeles, CA, United States

URL: *https://s3.amazonaws.com/metro-extracts.mapzen.com/los-angeles_california.osm.bz2*

1. Problems Encountered in the Map

Problematic characters in 'k' attributes of 'tag' nodes

The node 'tag' has attributes 'k' and 'v' that are to be converted into a MongoDB node and value, respectively. Some of the 'k' values contain characters that cannot be converted into a JSON attribute name. The problem tags were found using a variant of the script tags.py, changed to echo lines to the terminal when problematic characters were detected.

Some of the problem characters are trailing space characters. These were fixed by stripping the string before categorizing the tag. Examples included 'lane ' and 'building:levels:belowground '.

The other tags with problematic had internal spaces or punctuation in their 'k' attributes and can be ignored. The 'k' values that were discarded include: 'note:save for future use', 'lift:stop layers', 'est.', 'supressed name', and 'note:joke'.

Of these, <tag k="note: joke" v="Named for Dr. Albert Falls"/> belongs to the 'way' node (id=134991260) which also has <tag k="name" v="Schweitzer Falls"/>. It turns out that this is

probably in the Jungle Cruise ride in Disneyland. It's a poor joke. I did not feel bad leaving it out.

Mis-tagging in 'k' attributes of 'tag' nodes

'building:levels:belowground' should be 'building:levels:underground' and was converted.

Street name anomalies in the 'v' attribute of 'tag' nodes

Street name anomalies were discovered using a modified version of the script audit.py. This script is used to detect problems in 'v' attribute of 'tag' nodes when 'k' has the value 'addr:street' or the value 'name' inside 'way' nodes. audit.py also detected street names with leading blanks, like ' N Western Ave'.

* Trailing street types abbreviated in many different ways

For example: 'St', 'St.', 'Str', 'str', and 'street' for Street. Similar variability was seen and corrected for 'Avenue', 'Boulevard', 'Circle', 'Court', 'Drive', 'Highway', 'Interchange', 'Lane', 'Parkway', 'Place', 'Road', 'Terrace', 'Trail', 'View', and 'Way'. Interchanges were messy and left largely uncorrected. They used abbreviations internally as well as at the end of their strings. Some trailing substrings were corrected. For example, 'Ave Ic' for 'Avenue Interchange'.

* Intersection names

Nodes are sometimes located using street names that are street intersection names. I left these in. An example of such a tag is <tag k="addr:street" v="Orangethorpe Av @ Magnolia Av"/>.

* Leading direction abbreviations

North, South, East, and West in street names were often, and inconsistently, abbreviated. Three examples are: `<tag k="addr:street" v="N. Rosemead"/>`, `<tag k="addr:street" v="N Azusa Ave"/>`, and `<tag k="addr:street" v="North Brand Blvd"/>`. I tried cleaning these up, but gave up on the effort. There were legitimate non-abbreviations lurking in the data set, like `<tag k="name" v="N Street"/>`.

Street name mis-spells

I detected two spelling mistakes in street names while using a text editor: 'Commonwelth Avenue' (should be 'Commonwealth Avenue') and 'Lincon Boulevard' (should be 'Lincoln Boulevard').

Street name containing house numbers

An example is `<tag k="addr:street" v="360 Goddard"/>`. This should be broken up into an 'addr:street' k-attribute and an 'addr:housenumber' k-attribute. These, too, were left uncorrected.

addr:street_direction_prefix

Many addr:street_name assignments have N, E, W, or S prefixes. Some openstreet editors decided to break apart the address and employ a tag addr:street_direction_prefix instead. The openstreetmap wiki does not list the tag addr:street_direction_prefix.

Entries like

```
  <tag k="addr:street" v="Valley Blvd"/>
  <tag k="addr:street_direction_prefix" v="W"/>
```

were converted to

```
  <tag k="addr:street" v="West Valley Blvd"/>.
```

Address housenumbers containing street names

Address housenumbers should just be numbers but sometimes contain other address pieces. There are a handful of exceptions (6) in the data set. They generally occurred in 'node' nodes, as opposed to 'way' nodes. Judging that these would not be a problem in queries and I left them uncorrected. One example is

```
   <tag k="name" v="Mobile Oil 12746"/>
   <tag k="addr:housenumber" v="18744 Via Princessa"/>.
```

This is probably part of the address for a Mobil Oil gasoline station. The name is probably mis-spelled. The housenumber for a fast food restaurant in Murietta reads

```
   <tag k="addr:housenumber" v="39252 Winchester Rd Murrieta, CA 92563"/>
```

A phone number was used for the housenumber for Greenshine New Energy LLC. Of course, the street name contains the housenumber.

```
   <tag k="addr:street" v="360 Goddard"/>
   <tag k="addr:housenumber" v="9496099636"/>
```

Bad addr:postcode values in 'k' attributes of 'tag' nodes

Two sorts of errors are made in addr:postcode values. These were discovered by altering the audit.py code to examine the addr:postcode values, ensuring that they started with a number and started with a number in the range from 93010 to 93019.

Only two entries failed the range check: `<tag k="addr:postcode" v="62660"/>` and `<tag k="addr:postcode" v="722A"/>`. The first of these was corrected (to '92660') and the second was discarded.

The most frequent postcode mistake is to include the state in the code: `<tag k="addr:postcode" v="CA 93010"/>.`

2. Data Overview

This section contains basic statistics about the dataset.

File size

```
los-angeles_california.osm ......... 1.1 GB
1,094,896,305 bytes, 1,094,889,640 chars, 70,994,458 words, and 15,096,156 lines.
```

Number of documents

```
> db.openstreet.find().count()
5106009
```

Number of nodes

```
> db.openstreet.find({"type":"node"}).count()
4621566
```

Number of ways

```
> db.openstreet.find({"type":"way"}).count()
483840
```

Number of unique users

```
> db.openstreet.aggregate(
{ $group : { "_id" : "$created.user" }},
{ $group: { _id : null, count: { $sum: 1 } } });

{ "_id" : null, "count" : 2548 }
```

Number of freeways, highways, and theaters in Los Angeles

```
> db.openstreet.find( { "type" : "way", "name" : /.* Freeway/i } ).count();
5972
> db.openstreet.find( { "type" : "way", "name" : /.* Highway/i } ).count();
1163
> db.openstreet.find( { "type" : "node", "amenity" : "theatre" } ).count();
119
> db.openstreet.find( { "type" : "node", "amenity" : "cinema" } ).count();
55
```

3. Other Ideas About the Datasets

Data Set Uses

The amenities and position fields can be used to look up shops, restaurants, and theaters in a box centered on a user's position, obtained with the GPS in their phones.

The 'relation', 'nodes' and 'ways' can be used by a graph algorithm (nodes are graph-nodes and a combination of relation and ways define directed and undirected edges) to provide optimal-distance routes from one position to another. Combined with traffic information, optimal-time routes can be supplied.

The automatic generation of lists of sites and shops in a given city within the map area for consumption by tourists is yet another application.

Data Set Improvements

The address fields can be made more uniform by processing leading directional information (N, S, E, W). Some of the missing 'k' tags in 'tag' nodes, like addr:city and addr:postcode, could be filled in automatically using latitude and longitude and a list of the nodes that define city and postal zone boundaries. The boundary:administrative tag is available for this purpose. This could be done automatically, and the program could sign its changes like any other editor. Some provision for manually fending off the program's changes would need to be provided.

A tool that might help improve the quality of the data set is a query facility that looks for deviations from openstreetmap 'standard's made since a given time, defaulting to one hour. Editors can then check if their latest submissions introduce bad address formats or newly invented tags. The sheer variety of good address formats is a major difficulty in this scheme.

Openstreetmap needs to define guidelines about what to do at nodes that are road interchanges.

There are only five different layers. If it was possible to supply an API to 'plug-in' GIS layers then demographic and economic information might be supplied. This would open up business opportunities to vendors of that demographic information. The legal restrictions on the use of the open data would need to be specified. Another legal question to be answered: Can the openstreetmap foundation itself take in money for this service? Specifying a common user interface for the plug-ins will prove difficult, as it is hard to know ahead of time what sort of map layers would be desired.

More Data Set Analysis Fun

Number of users having one post

```
> db.openstreet.aggregate(
{ $group : { "_id" : "$created.user" , "count" : { $sum : 1 } } },
 { $match : { "count" : 1 }},
 { $group : { "_id" : null,"count" : { $sum : 1 }}} );

{ "_id" : null, "count" : 519 }
```

Users with more than 200,000 posts and the number of their posts

```
> db.openstreet.aggregate(
{ $group : { "_id" : "$created.user" , "count" : { $sum : 1 } } },
{ $match : { "count" : {$gt : 200000} }},
{ $sort : { "count" : -1 }});

{"_id" : "woodpeck_fixbot",  "count" : 577072},
{"_id" : "AM909", "count" : 474533 },
{"_id" : "The Temecula Mapper", "count" : 468373},
{"_id" : "nmixter", "count" : 330454},
```

```
{"_id" : "Brian@Brea", "count" : 218371}
```

Top 10 amenities

```
db.openstreet.aggregate(
{ $match : { "amenity" : { $exists : true }}},
{ $group : { "_id" : "$amenity" , "count" : { $sum : 1 } } },
{ $sort : { "count" : -1}},
{ $limit : 10 });
```

Parking is at the top. Of course.

```
{"_id" : "parking", "count" : 7188},
{"_id" : "school",      "count" : 5080},
{"_id" : "place_of_worship", "count" : 4158},
{"_id" : "restaurant", "count" : 2112},
{"_id" : "fast_food", "count" : 1665},
{"_id" : "fuel", "count" : 979},
{"_id" : "toilets",     "count" : 666},
{"_id" : "fire_station", "count" : 660},
{"_id" : "cafe", "count" : 596},
{"_id" : "swimming_pool", "count" : 546}
```

Number of food vendors that do not declare their cuisine

```
> db.openstreet.aggregate(
  {$match : {"amenity": {$in : ["cafe","restaurant","fast_food"]}}},
  {$match : {"cuisine" : {$exists:0}}},
  {$group:{"_id":"$cuisine", "count":{"$sum":1}}});

[ { "_id" : null, "count" : 1965 } ]
```

Top 10 cuisines

```
> db.openstreet.aggregate(
  {$match : {"amenity": {$in : ["cafe","restaurant","fast_food"]}}},
  {$match : {"cuisine" : {$exists:1}}},
  {$group:{"_id":"$cuisine", "count":{"$sum":1}}},
  {"$sort":{"count": -1}},
  {$limit : 11});
[{"_id" : "burger", "count" : 496},
{"_id" : "mexican", "count" : 305},
{"_id" : "american", "count" : 251},
{"_id" : "pizza", "count" : 154},
{"_id" : "coffee_shop", "count" : 136},
{"_id" : "sandwich", "count" : 111},
{"_id" : "chinese", "count" : 85},
{"_id" : "italian", "count" : 81},
{"_id" : "chicken", "count" : 73},
{"_id" : "japanese", "count" : 57},
{"_id" : "deli", "count" : 45}]
```