Open Street Map Project

Data Wrangling with MongoDB

Michael Mitchell

Map Area: Indianapolis, IN

www.openstreetmap.org/?bbox=-86.485,39.454,-85.746,40.134

https://s3.amazonaws.com/metro-extracts.mapzen.com/indianapolis_indiana.osm.bz2

This map area was chosen because it's the author's hometown and thus he has familiarity with the area.

1. Problems Encountered in the Map
   1.1 Tag key illegal characters
   1.2 Abbreviated street names
   1.3 Incorrect city names
2. Data Overview
3. Additional Ideas
4. Conclusion

1. Problems Encountered in the Map

1.1 Tag key illegal characters

As part of the OSM data modeling standard, an xml element <tag> can be included in a node, way, or relation in order to tag them with a descriptive key-value pair (e.g. a highway key with a residential value). As part of the data shaping import process to mongo, the key-value pairs are going to be directly placed in the node or way document (instead of nested inside a <tag>), so all of the keys need to be checked to make sure they contain legal characters for mongo. After the check, no issues were found.

```
Potential problem characters include [=\+/&<>;\'"\?%#$@
\,\. \t\r\n]
```

1.2 Abbreviated street names

Most of the data contains full suffixes for street names (e.g. Street, Avenue, Parkway, etc.) but some entries have shorted them to abbreviations. Worse, some abbreviations have punctuation while others omit punctuation. A function was created to check street name suffixes while data was being streamed into mongo, and update abbreviated names when encountered.

```
Problem street suffixes encountered: Ave, Blvd.,
Dr ,Dr., Ln, Pkwy, Rd, Rd., St
```

1.3 Incorrect city names

Some of the address info contained incorrect or poorly formatted city names; the issues were incorrect capitalization, extra white space, misspellings, and inclusion of state name. A function was created to check city names and correct them if needed while the data was being streamed into mongodb.

```
Example bad city names for 'Indianapolis' include
'Indianapolis, Indiana', 'Indianapolis ',
'indianapolis', 'Indianapoliss'
```

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

Stats:

File size indianapolis_indiana.osm: 312 MB

```
$ stat -f%z indianapolis_indiana.osm
```

Number of documents: *1,583,107*

```
> indianapolis.count()
```

Number of nodes: *1,440,335*

```
> indianapolis.find({"type":"node"}).count()
```

Number of ways: *142,764*

```
> indianapolis.find({"type”:”way”}).count()
```

Top 5 streets in addresses: *South Mooresville Road, South Holt Road, East Washington Street, East 96th Street, West Perry Street*

```
> indianapolis.aggregate([{"$match": {"address.street":
{"$exists":1}}},
                        {"$group": {"_id":
"$address.street", "count":{"$sum":1}}},
                        {"$sort": {"count": -1}},
                        {"$limit": 5}])
```

Top 5 cities in addresses: *Indianapolis, Brownsburg, Fishers, Greenwood, Noblesville*

```
> indianapolis.aggregate([{"$match": {"address.city":
{"$exists":1}}},
                        {"$group": {"_id":
"$address.city", "count":{"$sum":1}}},
                        {"$sort": {"count": -1}},
                        {"$limit": 5}])
```

3. Additional Ideas

Cross validation

The data in OSM could be improved by cross validating it with other mapping databases such as google maps, apple maps, or Garmin. Issues with this approach include formatting the data in a compatible way (e.g. another dataset may favor street abbreviations over full words), error in GPS measurements or choice of points for ways (e.g. another dataset could use a completely different set of points to mark out a road, even though the final shapes are similar), and access to those databases (e.g. will Apple let you have access to their data).

King of the hill

All of the user contributions to a particular geographic area could be counted and the contributor with highest count could be the 'King of the hill' for that region. This could add gamification to the contribution process and potentially increase engagement. Issues with this approach could include increased incentives for non-relevant or useless edits as people would be incentivized only for changes not necessarily correctness.

Conclusion

After reviewing the data, it's obvious that the data is generated by humans due to the inconsistencies in format and misspellings. The mistakes themselves are not too difficult to correct programmatically, but it seems like a better idea to use the issues found in the data to create validations run on the data when the user submits it in the first place. This way bad data never makes it to the database, and the user is given an error message and a chance to correct the data and resubmit.