Data Wrangling with MongoDB

Udacity Nanodegree Project

Program: Data Analyst Nanodegree

Project: 2 of 5

Supporting course(s):

- Data Wrangling with MongoDB

Project Overview

You will choose any area of the world in https://www.openstreetmap.org and use data munging techniques, such as assessing the quality of the data for validity, accuracy, completeness, consistency and uniformity, to clean the OpenStreetMap data for a part of the world that you care about.

Project Depencies

Python 2.7x

Final Project Report

Task 1: Lesson 6 Programming Exercises

Lesson 6 programming exercise solutions can be found in the lessons directory.

Task 2: Process Dataset

Map area: Somerville, Massachusetts, United States

OpenStreetMap: View Map

Download source: Overpass API

Task 3: Document Findings

Data Overview

File sizes

- Size of XML file: 133.4 MB
- Size of JSON file: 150.7 MB

Note: A sample of the XML OSM data is available: finalproject/data/sample.osm

Reason for Selection

My two younger brothers attend Tufts University in Somerville, MA. I often visit and thought the dense urban neighborhood would be interesting to review.

XML Data Overview

Note: I wrote 4 modules to assist in data review and transformation:

- streetauditor.py
- summarize.py
- tagauditor.py
- transformer.py

Tag element "K" attribute analysis

I wrote a recursive algorithm to parse compund k attribute values (multple words separated by a ":") found in "tag" elements.

This breakdown provides an interesting look into the type and frequency of data provided in the "tag" element.

Note: the "root" property is the frequency count for the parent key.

See the k attribute analysis file in: finalprojects/data/k-breakdown.json

Top Level XML Elements

Module: summarize.py

Code:

```
summarize.get_top_level_tag_summary(OSM_FILE)
```

Results:

```
{
    'node': 596989,
    'member': 10171,
    'nd': 714378,
    'tag': 227679,
    'bounds': 1,
    'note': 1,
    'meta': 1,
    'relation': 534,
    'way': 95546,
    'osm': 1
}
```

Number of Unique Contributing Users

Module: summarize.py

Code:

```
summarize.get_number_of_contributors(OSM_FILE)
```

Result: 577

MongoDB Data Overview

Number of documents uploaded

MongoDB Query:

```
db.somer.count()
```

Result: 692535

Number of nodes

MongoDB Query:

```
db.somer.find( { 'type': 'node' } ).count()
```

Result: 596878

Number of ways

MongoDB Query:

```
db.somer.find( { 'type': 'way' } ).count()
```

Result: 95532

Top 5 Contributing Users

Query:

```
db.somer.aggregate([
    { "$group" : { "_id" : "$created.user", "total" : { "$sum" : 1 } } },
    { "$sort" : { "total" : -1 } },
    { "$limit" : 5 }
])
```

Result:

User	Count
crschmidt	370001
jremillard-massgis	93258
OceanVortex	83486
ingalls_imports	29064
morganwahl	27541

Top 10 amenity types

MongoDB Query:

```
db.somer.aggregate([
    { "$group" : { "_id" : "$amenity", "total" : { "$sum" : 1 } } },
    { "$sort" : { "total" : -1 } },
    { "$limit" : 10 }
])
```

Result:

Amenity	Count

Amenity	Count
parking	660
bench	415
restaurant	268
school	199
place_of_worship	184
bicycle_parking	166
hydrant	124
library	110
cafe	85
university	77

Top 10 Cuisine Types

MongoDB Query:

```
db.somer.aggregate([
    { "$group" : { "_id" : "$cuisine", "total" : { "$sum" : 1 } } },
    { "$sort" : { "total" : -1 } },
    { "$limit" : 10 }
])
```

Results:

Cuisine	Count
pizza	30
chinese	22
mexican	19
american	18
italian	16
sandwich	16
indian	15
coffee_shop	14

Cuisine	Count
burger	11
thai	8

Problems encountered in map

Problem #1: Street Names

Inconsistent abbreviations for streetnames exist in the data set.

Examples

For "Street" the following abbreviations were used:

- "St"
- "st"
- "ST"
- "St."

For "Avenue" the following abbreviations were used:

- "ave"
- "Ave."
- "Ave"

To resolve these inconsistencies I utilized the module streetauditor.py to normalize street names.

Problem #2: Outdated Data

Somerville is a densely nested active area surrounded by colleges and universities. Construction is continous and local businesses rise and fall. With this turnover, it is important to have "fresh" data. See below for analysis:

Total number of nodes with a timestamp:

Query:

```
db.somer.find({"created.timestamp" : { "$exists" : 1 } } ).count()
```

Result: 692535

Breakdown:

Query:

```
db.somer.find( { "created.timestamp" : { "$gte" : "2015-01-01T00:00:00Z" }
```

Created since	Number of nodes	% of nodes
2015	5930	0.85 %
2014	26408	3.81 %
2013	143707	20.75 %

The table above shows that only a fraction of data is current within the past year. Going back a full two years only provides ~20% of the data. Flip that around ~80% of the map data is more than two years old and may be no longer useful.

Task 4. Additional Ideas

Partner with local Universities and Government Organizations

In the analysis below there is a significant amount of GIS data present in this data set. GIS data ranks as the second most common k value and contains the second most number of variations. This could be due to the number of Universities in Boston who teach GIS mapping classes as well as active government organizations who use this type of data.

It may be possible to reach out to the surrounding academic and government organizations and create some type of partnership. Students/employees would be given a set of standards to follow and ask to upload any data they work with to the OSM Street Map site. This effort would assist in solving the problem of "stale/old" data as well as improve standards and consistency.

This idea does contain potential draw backs in that it requires continual over site to ensure quality standards are being met. With a potentially large amounts of contributions coming in, if standards are not met the dataset could become "polluted" quickly.

Types of similar data

I wrote a recursive algorithm to count the variations of compound k attribute values in "tag" elements.

Example:

- addr:street
- addr:zip

Above "addr" would be reported as having two variations and a count of two.

I uploaded the summary data (see file data/k-summary.json) into MongoDB and performed a query to identify the most variable k values and most popular.

Top 10 most frequently occurring k values

MongoDB query:

```
db.ksum.find({ }, { "_id" : 0 } ).sort( { "count" : -1 } ).limit( 10 )
```

Result:

K value	Variations	Count
building	8	81467
massgis	57	17791
source	10	16026
highway	1	14609
addr	13	14281
name	258	11892
attribution	1	10975
lanes	2	5961
condition	1	5883
width	1	5770

Top 10 K values with the most variations

MongoDB query:

```
db.ksum.find({ }, { "_id" : 0 } ).sort( { "variations" : -1 } ).limit( 10 )
```

Results:

K value	Variations
name	258
massgis	57
ngis	18
addr	13
source	10
roof	9
contact	9
building	8
service	7
seamark	7

Adding gamification to incentivize quality of submission

In the analysis below roughly 63% of users submitting to the dataset add 10 or fewer contributions. Rather than focus on frequency of submission, it would be valuable to incentivize the quality of submission. By recognizing and affirming users for the quality and inclusion of "must have" data in each submission, each contribution becomes more meaningful both to the user and to the dataset.

Number of Contributions Per User

MongoDB Query:

```
db.somer.aggregate([
    { "$group" : { "_id" : "$created.user", "numContribs" : { "$sum" : 1 }
    { "$group" : { "_id" : "null", "avgContribPerUser" : { "$avg" : "$numCo
])
```

Result: Avg. number of contributions per user: ~1238.8

This average is very high. When observing the top 10 contributors have a range from ~370000 to 5325. This indicates that we have a skewed data set with a few outliers on the upper end of the distribution that are pulling our average up.

Running variations of the following query shows us that of the 577 unique contributors the majority of them submitted ten or fewer times.

```
db.somer.aggregate([
    { "$group" : { "_id" : "$created.user", "total" : { "$sum" : 1 } } },
    { "$match" : { "total": { "$lte" : 10 } } },
    { "$group" : { "_id" : "null", "numOfUsers" : { "$sum" : 1 } } }
])
```

Number of users	% of users	Contributions
143	~25%	1
295	~50%	<= 5
365	~63%	<= 10

Resources

All resources used/referenced are listed in the file resources.txt.