Project 2 - Data Wrangling With MongoDB

Project Summary

What Is Your Name?

David Broadwater

What E-mail Address Do You Use To Sign In To Udacity?

david.broadwater@gmail.com

What Area Of The World You Used For Your Project? Post A Link To The Map Position And Write A Short Description. Note That The Osm File Of The Map Should Be At Least 50mb.

- URL: http://osm.org/go/TPTKToK
- Export bounding box: `32.6739 : 32.7738, -117.2030 : -117.0678`
- This area includes downtown San Diego, as well as portions of Coronado and National City. I chose this particular area because it is near where I currently live and where I hope to be moving within the next year.

Is There A List Of Web Sites, Books, Forums, Blog Posts, Github Repositories Etc That You Referred To Or Used In This Submission?

See the attached file `resources_used_ud032.txt` for a list of websites used for completing the project and problem sets.

Please Carefully Read The Following Statement And Include It In Your Email:

“I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, Github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate.”

Is There Any Other Important Information That You Would Want Your Project Evaluator To Know?

No, everything has been covered in the rest of the write-up.

Problems Encountered In The Map

First, `mapparser.py` was run to make sure it could parse the OSM (OpenStreetMap) xml file correctly and to see how many tags were in the dataset. As expected, there were a large number (207,353) of `node` tags in the file.

Next, the `tags.py` script was used to see how many kinds of each type of "k" tag were present. This script identified a large number of "other" keys (11,598) and two keys with problem characters. In order to see what kinds of formatting schemes were present in the data, the script was modified to add items identified in those categories to sets representing each category. Otherwise, most of the tags were in the expected lowercase format:

```
{'lower': 130900, 'lower_colon': 721621, 'other': 11598, 'problemchars': 2}
```

The problem characters (`problemchars`) set consisted of:

```
set(['color box', 'park trail31']))
```

Without additional knowledge about the origin of these fields, simply replacing the whitespace (" ") characters with an underscore seemed like the most appropriate fix and was implemented later in the data cleaning process.

The `others` set consisted of a wide range of name patterns (as expected, since by default this was the category for everything else). Most of the entries simply had one or more letters capitalized, which was easily fixed using the `str.lower()` method. There were also some keys with more than one colon, as well as some keys with numbers.

The keys with multiple colons were broken down into the appropriate dictionaries and sub-dictionaries, as implemented with the `addr:` fields in the Lesson 6 problem set. This entailed some modifications to the Lesson 6 code. Both of these changes complied with the general "guidelines" outlined on the OSM Wiki.

Running the `users.py` script revealed a large number of unique contributors (343). This was not surprising given the fact that San Diego is a large metropolitan area; as such, there were likely a significant number of people interested in map data for the area. While additional analysis could have been done in Python to determine more information about the contributors, for the purposes of this project it was done later in MongoDB.

Next, the street names were audited using the `audit.py` script. According to the "Names" page of the OSM Wiki, "if the name can be spelled without an abbreviation, then don't abbreviate it." This confirmed that the auditing and data cleaning stages of this effort conformed with the standards of the dataset.

The output of the `audit.py` script showed there were many issues with correct street types in the dataset. There were many unfamiliar street types, along with some common abbreviations that could be easily fixed. Since San Diego shares such a close proximity to Mexico, many of the street names are in Spanish (presumably since parts of California used to belong to Mexico). Consequently, some of the streets are named such that the street type is actually a prefix to the full name (e.g., Avenida Del Mundo, Camino Del Rio, Valle Vista, etc.), versus a suffix to the name (e.g., El Cajon Boulevard, Hotel Circle, etc.). Like their common English counterparts, these names also have their own abbreviations that needed to be fixed as well. Fortunately, a web search produced a pdf document entitled "Address Help List" and published by the city of San Diego which outlined common street type prefix and suffix abbreviations and their correct full name counterparts.

This document was the foundation for updating the dictionary of street type corrections, as well as a new dictionary of street type prefixes. In addition, the original code was modified to search for matches in the prefix dictionary when matches in the suffix dictionary were not found.

To complicate matters, one of the the Spanish street types which appeared in the dataset was a suffix (“Paseo”), so it was added to the list of expected suffix street types. Other unique street types were found (such as “Mews”), but were confirmed as correct street names through cross-referencing using Google Maps. There were other abbreviations that appeared in the dataset but did not conform to the abbreviations in the official document; these were corrected to the most reasonable common street type (e.g., “Pk” changed to “Parkway,” “Hw” changed to “Highway,” etc.).

“North” and “South” were added as expected street types for this dataset because the street names associated with them were otherwise in the correct format (e.g., Hotel Circle South).

Finally, some custom mappings were added to the correction dictionaries based on looking up the road name in Google Maps and determining the omitted street type based on the search results. There were five street names with multiple possible street types in the San Diego area (according to Google Maps), so they were left unchanged. A total number of 1,576 street names were corrected; this number was calculated using a counter which incremented each time a street name was updated.

Next, a modified version of `data.py` was used to perform the street type corrections previously mentioned and shape the data in the manner outlined in the Lesson 6 problem set. In order to perform the street type corrections, the `update_name` function, expected street type lists, street type corrections libraries, and regular expression searches were modified. This script produced a JSON file as output, shaped in the desired format with street name corrections applied. The original code to output the JSON document had to be modified to include the enclosing square brackets and commas between entries for proper MongoDB importation.

Lastly, the JSON file was imported into MongoDB using the `dbinsert.py` script. Upon initial analysis, it was discovered that many tags were not transferred over to the JSON file due to an omission in the `data.py` script. Specifically, non-address tags were not transferred over; these issues were corrected and the JSON file was re-imported into MongoDB after deleting all of the data from the original transfer. All of the San Diego map information was imported into a new collection called `sd` in a new database called `osm`.

Overview Of The Data

Methodology

For the data analysis portion of this project, MongoDB was used to query the database of Open Street Map data. All of the queries were performed in the MongoDB shell (version 2.6.7 running on Mac OS X 10.10.2). For each of the sections that follow, a brief description is provided of the reason for the query, followed by the query itself (preceded by “>”), query results (one line below the query), and a brief analysis. The term “document” as used below adopts the MongoDB definition of the term, corresponding to database “entry” or in this case, OSM feature.

The Size Of The File:

The `osm` database, which only contained the `sd` collection, had a file size of 196.6 MB, as reported by MongoDB. This was much larger than the file sizes of the associated XML (78.7 MB) and JSON (78.3 MB) files as reported by Mac OS X.

Number Of Nodes:

This query returned a count of all of the documents in the `sd` collection labeled as `node`.

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

207348
```

There were 207,348 `node` documents in the `sd` collection.

Number of Ways:

This query returned a count of all of the documents in the `sd` collection labeled as `way`.

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

16410
```

There were 16,410 `way` documents in the `sd` collection.

Number Of Unique Users:

This query returned the number of unique users who edited or added OSM map features in the area chosen for this effort.

```
>db.sd.distinct('created.user').length

339
```

There were 339 unique contributors for this dataset. Since San Diego is a major metropolitan area and the map area for this effort contains a large number of features, it is not surprising there were a large number of contributors.

Number Of Documents With A User Listed

This query returned the number of documents with a user listed. This number was calculated to assist in determining the top contributors in the dataset (see below).

```
>db.sd.find({"created.user": {"$exists": "True"}}).count()

223768
```

Top Contributors In The Dataset:

This query found all of the documents with a user listed, counted the number of documents they created, calculated the percentage of overall documents they created, and returned the count and percentage for the top 5 most frequently occurring users.

```
>db.sd.aggregate([{"$match": {"created.user": {"$exists": "True"}}},
                 {"$group": {"_id": "$created.user",
                             "count": {"$sum": 1}}},
                 {"$project": {"_id": 1, "count": 1,
                               "percentage": {"$divide": ["$count", 223768]}}},
                 {"$sort": {"count": -1}},
                 {"$limit": 5}])

{ "_id" : "Adam Geitgey", "count" : 123451, "percentage" : 0.5516919309284616 }
{ "_id" : "Sat", "count" : 20176, "percentage" : 0.09016481355689822 }
{ "_id" : "woodpeck_fixbot", "count" : 15644, "percentage" : 0.06991169425476386 }
{ "_id" : "javbw", "count" : 11608, "percentage" : 0.05187515641199814 }
{ "_id" : "evil saltine", "count" : 7042, "percentage" : 0.03147009402595546 }
```

Adam Geitgey was by far the most active contributor, with 123,451 documents created, accounting for 55.2% of the dataset. A quick Google search and review of his LinkedIn profile revealed that he worked for a company in San Diego which developed "geospatial software products that provide a common operating picture to crisis response teams," which likely explains why he was so active in contributing to OSM data in the San Diego area. Due to the cryptic user names of the other top contributors, it was more difficult to determine their origin. However, the OSM Wiki included Adam Geitgey and "woodpeck_fixbot" on a list of "users that have run imports or Automated_Edits on a significant scale". This page indicated that Adam Geitgey's contributions mainly involved importing information from SanGIS, a "regional geographic information system (GIS) landbase and data warehouse" run by the City of San Diego and County of San Diego. According to an OSM wiki about SanGIS written by Adam Geitgey, the import included public/commercial building footprints, military bases, national parks, bus stops, and addresses. According to the wiki,

> "Data is being imported only as determined to be useful. There is currently no plan to replace the streets in OSM with streets from SanGIS. While the SanGIS data quality is excellent, OSM already has good coverage with many edits."

Unlike Adam Geitgey, the previously mentioned significant contributor wiki page indicated "woodpeck_fixbot"'s contributions primarily involved automated edits throughout the world. A review of the edit log for "woodpeck_fixbot" linked from the significant contributor wiki page indicated the automatic fixes which were applied to the United States map area involved correcting the tags associated with "highway" ways, and removing superfluous tags. Further information could not be found for the additional top contributors for this dataset, but given the significant number of their contributions, it is likely they were performed using some sort of import or correction script.

Number Of Documents With A Zip Code Listed

This query returned the number of documents with a zip code listed. This number was calculated to assist in determining the most frequently occurring zip codes in the dataset (see below).

```
>db.sd.find({"address.postcode": {"$exists": "True"}}).count()

66364
```

Zip Codes Listed:

This query found all of the documents in the collection which had a zip code (or postcode, using OSM terminology), counted how many times each zip code appeared in documents in the collection, and returned the zip codes and counts in descending order. The results presented here were altered slightly to include the preferred city name for each zip code according to the US Postal Service using this website. These city names were appended to each line of the results in the format of Python comments (i.e., `# San Diego`).

```
>db.sd.aggregate([{"$match": {"address.postcode": {"$exists": "True"}}},
                 {"$group": {"_id": "$address.postcode",
                             "count": {"$sum": 1}}},
                 {"$sort": {"count": -1}}])

{ "_id" : "92105", "count" : 10136 } # San Diego
{ "_id" : "92104", "count" : 8446 } # San Diego
{ "_id" : "92113", "count" : 7222 } # San Diego
{ "_id" : "92116", "count" : 6662 } # San Diego
{ "_id" : "92102", "count" : 6498 } # San Diego
{ "_id" : "92103", "count" : 6363 } # San Diego
{ "_id" : "92115", "count" : 4831 } # San Diego
{ "_id" : "92114", "count" : 4004 } # San Diego
{ "_id" : "91950", "count" : 3709 } # National City
{ "_id" : "92118", "count" : 3476 } # Coronado
{ "_id" : "92101", "count" : 2062 } # San Diego
{ "_id" : "92110", "count" : 1352 } # San Diego
{ "_id" : "92111", "count" : 847 } # San Diego
{ "_id" : "92139", "count" : 321 } # San Diego
{ "_id" : "92108", "count" : 288 } # San Diego
{ "_id" : "92136", "count" : 77 } # San Diego
{ "_id" : "92135", "count" : 26 } # San Diego / NAS North Island
{ "_id" : "CA 92101", "count" : 18 } # San Diego
{ "_id" : "92154", "count" : 4 } # San Diego
{ "_id" : "92140", "count" : 4 } # San Diego
{ "_id" : "92182", "count" : 3 } # San Diego
{ "_id" : "92100", "count" : 3 } # NOT VALID ZIPCODE
{ "_id" : "92103-3609", "count" : 2 }  # San Diego
{ "_id" : "92102-4810", "count" : 1 } # San Diego
{ "_id" : "92110-9998", "count" : 1 } # San Diego
{ "_id" : "91932", "count" : 1 } # Imperial Beach
{ "_id" : "92037", "count" : 1 } #La Jolla
{ "_id" : "92103-3607", "count" : 1 }  # San Diego
{ "_id" : "92024", "count" : 1 } # Encinitas
{ "_id" : "92020", "count" : 1 } # El Cajon
{ "_id" : "92108-3803", "count" : 1 }  # San Diego
{ "_id" : "CA 92101-6144", "count" : 1 }  # San Diego
{ "_id" : "92137", "count" : 1 }  # San Diego
```

As expected, most of the zip codes listed (89.1%) were associated with San Diego. Most of the other zip codes belonged to the surrounding communities of National City (5.6%) and Coronado (5.2%). Imperial Beach, La Jolla, Encinitas, and El Cajon each had a single document associated with their zip codes, despite the fact their city limits (according to Google Maps) did not overlap the geographic area used in this effort. Lastly, three documents were incorrectly labeled with an invalid zip code (92100), likely due to human error, as zip codes 92101-92109 are all associated with San Diego and only differ by one digit.

Number Of Documents With An `amenity` Field Listed

This query returned the number of documents with an `amenity` field. This number was calculated to assist in determining the most frequently occurring amenity types in the dataset (see below).

```
>db.sd.find({"amenity": {"$exists": "True"}}).count()

1685
```

Most Popular Type Of Amenity:

This query found all of the documents with an `amenity` field, counted how many times each amenity type appeared in those documents, and returned the top 20 most frequently occurring amenity types and counts in descending order.

```
>db.sd.aggregate([{"$match": {"amenity": {"$exists": "True"}}},
                 {"$group": {"_id": "$amenity",
                             "count": {"$sum": 1}}},
                 {"$sort": {"count": -1}},
                 {"$limit": 20}])

{ "_id" : "place_of_worship", "count" : 364 }
{ "_id" : "parking", "count" : 339 }
{ "_id" : "restaurant", "count" : 152 }
{ "_id" : "school", "count" : 149 }
{ "_id" : "fast_food", "count" : 115 }
{ "_id" : "bar", "count" : 99 }
{ "_id" : "cafe", "count" : 40 }
{ "_id" : "fuel", "count" : 27 }
{ "_id" : "bench", "count" : 24 }
{ "_id" : "toilets", "count" : 23 }
{ "_id" : "drinking_water", "count" : 21 }
{ "_id" : "library", "count" : 21 }
{ "_id" : "bank", "count" : 20 }
{ "_id" : "parking_entrance", "count" : 20 }
{ "_id" : "hospital", "count" : 20 }
{ "_id" : "post_box", "count" : 16 }
{ "_id" : "atm", "count" : 16 }
{ "_id" : "post_office", "count" : 14 }
{ "_id" : "theatre", "count" : 13 }
{ "_id" : "fountain", "count" : 12 }
```

The two amenity types which occurred the most in the dataset were `place_of_worship` (21.6%) and `parking` (20.1%). Restaurants (9.0%), schools (8.8%) and fast food places (6.8%) rounded out the top five most popular amenity types for the dataset.

Number Of Documents With A `name` Field And Labeled As `fast_food`

This query returned the number of documents with a `name` field and labeled as `fast_food`. This number was calculated to assist in determining the most frequently occurring fast food name (i.e., chain) in the dataset (see below).

```
>db.sd.aggregate([{"$match": {"name": {"$exists": "True"},
                              "amenity": "fast_food"}},
                 {"$group": {"_id": "Fast Food Places",
                             "count": {"$sum": 1}}}])

{ "_id" : "Fast Food Places", "count" : 115 }
```

There were 115 fast food amenities with a name listed, which matched the total number of fast food amenities in the dataset.

Most Popular Name Of Fast Food:

This query found all of the documents with a `name` field and labeled as `fast_food`, counted how many times each name appeared in those documents, and returned the top 20 most frequently occurring fast food names and counts in descending order.

```
>db.sd.aggregate([{"$match": {"name": {"$exists": "True"},
                              "amenity": "fast_food"}},
                  {"$group": {"_id": "$name",
                              "count": {"$sum": 1}}},
                  {"$sort": {"count": -1}},
                  {"$limit": 20}])

{ "_id" : "Jack in the Box", "count" : 16 }
{ "_id" : "Subway Sandwiches", "count" : 13 }
{ "_id" : "McDonald's", "count" : 9 }
{ "_id" : "Carls Jr.", "count" : 5 }
{ "_id" : "Dominos Pizza", "count" : 4 }
{ "_id" : "Taco Bell", "count" : 4 }
{ "_id" : "Wendy's", "count" : 3 }
{ "_id" : "Papa Johns", "count" : 3 }
{ "_id" : "Burger King", "count" : 3 }
{ "_id" : "Panda Express", "count" : 3 }
{ "_id" : "KFC", "count" : 3 }
{ "_id" : "Del Taco", "count" : 3 }
{ "_id" : "Subway", "count" : 3 }
{ "_id" : "Quiznos Subs", "count" : 2 }
{ "_id" : "Bruegger's Bagels", "count" : 2 }
{ "_id" : "Rubio's Baja Grill", "count" : 2 }
{ "_id" : "Wetzel's Pretzels", "count" : 2 }
{ "_id" : "Cotijas", "count" : 1 }
{ "_id" : "Cold Stone Creamery", "count" : 1 }
{ "_id" : "Dairy Queen", "count" : 1 }
```

Jack in the Box was the most frequently occurring fast food restaurant (13.9%), likely due to the fact they are based in San Diego. Otherwise most of the major fast food chains were well represented. Rubio's Baja Grill is also based in the San Diego area, in Carlsbad.

Number Of Documents With A `cuisine` Field And Labeled As `restaurant`

This query returned the number of documents with a `cuisine` field and labeled as `restaurant`. This number was calculated to assist in determining the most frequently occurring restaurant cuisine types in the dataset (see below).

```
>db.sd.aggregate([{"$match": {"cuisine": {"$exists": "True"},
                              "amenity": "restaurant"}},
                  {"$group": {"_id": "Restaurants",
                              "count": {"$sum": 1}}}])

{ "_id" : "Restaurants", "count" : 75 }
```

There were a significant number of food amenities missing a cuisine type in this dataset. Only 49.3% of the restaurants in this dataset had a cuisine listed.

Most Popular Restaurant Cuisine:

This query found all of the documents with a `cuisine` field and labeled as `restaurant`, counted how many times each cuisine type appeared in those documents, and returned the top 10 most frequently occurring restaurant cuisine types and counts in descending order.

```
>db.sd.aggregate([{"$match": {"cuisine": {"$exists": "True"},
                              "amenity": "restaurant"}},
                  {"$group": {"_id": "$cuisine",
                              "count": {"$sum": 1}}},
                  {"$sort": {"count": -1}},
                  {"$limit": 10}])

{ "_id" : "mexican", "count" : 15 }
{ "_id" : "american", "count" : 8 }
{ "_id" : "pizza", "count" : 5 }
{ "_id" : "sushi", "count" : 5 }
{ "_id" : "italian", "count" : 4 }
{ "_id" : "thai", "count" : 4 }
{ "_id" : "burger", "count" : 3 }
{ "_id" : "steak_house", "count" : 3 }
{ "_id" : "deli", "count" : 2 }
{ "_id" : "sandwich", "count" : 2 }
```

Mexican (20.2%) was the most popular restaurant cuisine in the dataset, likely due to San Diego's close proximity to Mexico. Not surprisingly, `american` cuisine (which typically includes hamburgers; 10.6%), `pizza` (6.7%), and `sushi` (likely due to the proximity to the ocean; 6.7%) followed in popularity.

Number Of Documents With A `religion` Field And Labeled As `place_of_worship`

This query returned the number of documents with a `name` field and labeled as `place_of_worship`. This number was calculated to assist in determining the most frequently occurring religion type in the dataset (see below).

```
>db.sd.aggregate([{"$match": {"religion": {"$exists": "True"},
                              "amenity": "place_of_worship"}},
                  {"$group": {"_id": "Religions",
                              "count": {"$sum": 1}}}])

{ "_id" : "Religions", "count" : 339 }
```

There were 339 places of worship with a religion type listed, compared to 364 places of worship in the dataset. Adding religion types for the 25 places of worship missing them is another opportunity for improving the dataset.

Most Popular Type Of Religion:

This query found all of the documents with a `religion` field and labeled as `place_of_worship`, counted how many times each religion appeared in those documents, and returned the top 10 most frequently occurring religions and counts in descending order.

```
>db.sd.aggregate([{"$match": {"religion": {"$exists": "True"},
                              "amenity": "place_of_worship"}},
                 {"$group": {"_id": "$religion",
                             "count": {"$sum": 1}}},
                 {"$sort": {"count": -1}},
                 {"$limit": 10}])

{ "_id" : "christian", "count" : 324 }
{ "_id" : "buddhist", "count" : 4 }
{ "_id" : "muslim", "count" : 3 }
{ "_id" : "hindu", "count" : 2 }
{ "_id" : "bahai", "count" : 1 }
{ "_id" : "scientologist", "count" : 1 }
{ "_id" : "taoist", "count" : 1 }
{ "_id" : "ascended_master_teachings", "count" : 1 }
{ "_id" : "jewish", "count" : 1 }
{ "_id" : "unitarian", "count" : 1 }
```

`Christian` places of worship represented an overwhelming majority (95.6%) of the dataset. Given Christianity's position as the dominant religion in the US (78.3%) and Mexico (95.1%), San Diego's close neighbor, this is unsurprising. `Buddhist` (1.1%), `Muslim` (0.9%), and `Hindu` (0.6%) were the next three most popular religions in the dataset.

Other Ideas About The Dataset

Given the ubiquity of websites, services, and phone applications which use map data, it is easy to see the potential for a dataset like the one examined here. This information could be used for things like finding the closest hamburger fast food joint, making a reservation at the closest Italian restaurant, finding directions to your friend's house, or looking up new churches to try in the local area. The biggest difficulty in improving a dataset like OSM is that it relies heavily on users to contribute and improve it, while a majority of users of map data likely use more mainstream services like Google Maps, Apple Maps, or MapQuest and are probably not interested in constantly contributing. With the normal cycle of things like businesses opening and closing, road construction, and other changes, an OSM map could quickly become out-of-date unless there are active users contributing to the map area. While users have created scripts to import massive amounts of data from other sources, sometimes this can introduce additional headaches such as incorrect values or superfluous information.

As previously identified throughout this analysis, there are many areas for improvement in this dataset. First, an analysis could be done to determine how many nodes with an address do not have both a city and zip code listed and fill in the missing information. Similarly, missing fast food and restaurant cuisine types, and religious types and denominations could be filled in. For the mismatched city names and zip codes, a detailed coordinate analysis could be done to determine the correct city name and zip code pair. For the addresses with formatting issues or missing street types, the coordinates could be used to determine the most probable correct value. Finally, a detailed investigation could be done to identify which users created the features with incorrect or mismatched values in order to potentially identify additional documents for closer scrutiny or to predict what types of users are more prone to producing errors.

With a dataset a large as this one and with as many different contributors, many data wrangling cycles would be necessary to fully clean and standardize the data. Fortunately, there are many cross-referencing resources available for users with the both the time and motivation to accomplish these improvements, which are beyond the scope of this work. This dataset shows the need for a

standardized data schema, as well as the difficulty of having many different data sources and having information which needs to be constantly updated.