San Diego Map Analysis

Drew Hontz 1/19/2017

Problems encountered in your map

Postcodes

Postcodes were inconsistent in format, they were either the full length postcode or a range of codes. In order to clean these values, if the code expressed a range (i.e contained a ';') I split the list on the ';' character, formed a list of postcodes between the two numerical values, and stored that list. If the postcode value contained a '-', I removed the '-' and all following digits (or retained the first 5 digits)

Housenumbers

The housenumbers were clean for the most part but there were some issues with 1/2 addresses. Having a 123 1/2 address is completely valid but some addresses had .5 instead of 1/2 in order to be consistent I changed all .5 values to 1/2. Additionally, there were a few ranges in here as well, so similar to the postcode cleaning task, I split values with ';' characters and created a list using python's range function

Street Names

Some of our street names used abbreviations so I created a dictionary to map the abbreviations to their proper name and used that value instead.

Phone Numbers

Our phone numbers were in several different formats so I decided to remove all non digit characters, remove leading 1's, then add a '-' in the third and sixth index in the string to achieve the XXX-XXX-XXXX format.

Cuisine

Cuisine had a lot of redundant entries like 'donut', 'doughnut', and 'doughnuts'. I created a set of all unique values under cuisine and scanned through them to write rules to help consolidate all of these unnecessary categories.

Fast Food Restaurant Names

This was by far the category that needed the most attention. There were several different spellings for most of our franchises; Carl's Jr. had 7 different spellings, while In-N-Out Burger, and Jack in the Box, had multiple entries as well. I wrote a function that would flag entries that had the first 4 characters similar to another entry to help cut down on the time I would spend looking for each entry that needed consolidation. Ultimately I ended up writing a regex to find and replace erroneous entries (like Carls Jr) with their correct value (Carl's Jr.)

Overview of the Data

File Size

Here are the file sizes of the files I worked with during this project. The sample is what I used in my auditing and writing rules for the cleaning functions phase, the san-diego_california.osm was used as the full data set, and sd.json is the jsonify output of the shaped data from san-diego_california.osm

- sample.osm 30.1 MB
- san-diego_california.osm 296.7 MB
- sd.json 300.7 MB

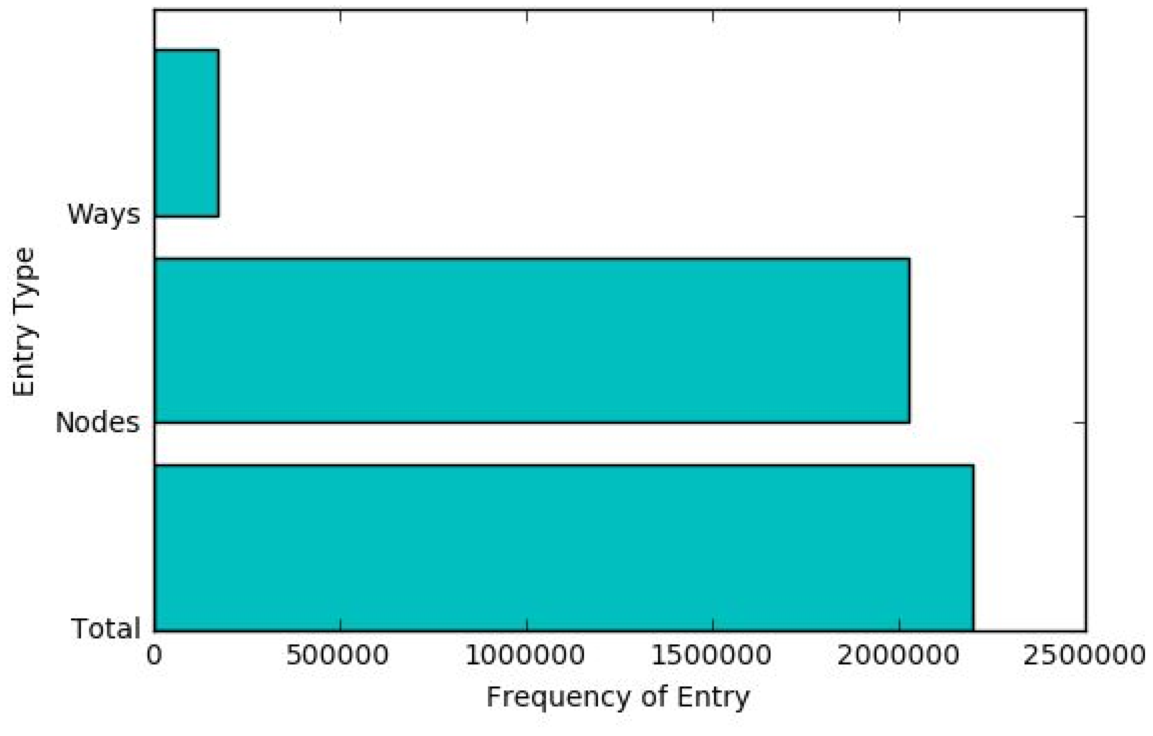
Number of documents, nodes, and ways

In [1]:
```
from IPython.display import Image
```

As you can see below, our nodes vastly outweigh the number of ways but that is to be expected as a way is comprised of many nodes.

In [2]:
```
Image(filename=r"C:\Users\Drew\Desktop\Code\DAND\Wrangle\Images\doc_vis.jpg")
```

Out[2]:



```
Number of entries:  2199884
Number of nodes:    2024896
Number of ways:     174710
```

Querying for the data above

In order to get an idea of how many nodes, ways, and total documents existed I ran the following three queries (in the order of total, node, way):

```
col.find().count()

col.find({"type": "node"}).count()

col.find({"type": "way"}).count()
```

Number of users

Here is the query* I ran to retrieve the number of unique users:

```
len(col.distinct('created.user'))
```

Which returned 953 Distinct users

*col is the collection name

Top ten contributors

I wrote a convenience function for returning a sorted list of values and frequencies called get_field_counts

I used this along with a limit parameter to return the ten users with the most contributions to the San Diego Map.

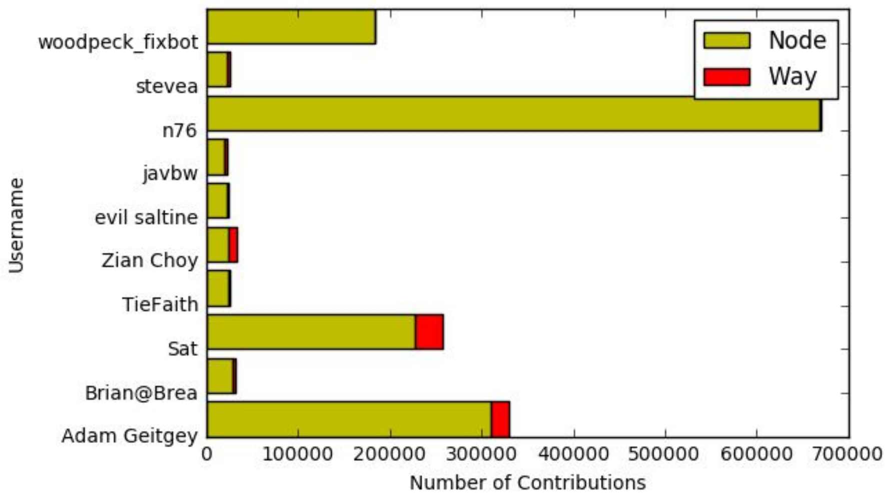
Here are the results with user, contributions, and proportion of the map.

User	Contributions	Proportion of Map Contributed
n76	670346	0.304718794264
Adam Geitgey	329612	0.149831536572
Sat	257028	0.116837069591
woodpeck_fixbot	183974	0.0836289549813
Zian Choy	32644	0.0148389642363
Brian@Brea	31862	0.0144834909477
TieFaith	26000	0.0118188049915
stevea	25314	0.0115069703675
evil saltine	24300	0.0110460369729
javbw	22064	0.0100296197436

In order to get more insight into these contributions, lets look at what proportion of these top ten users' contributions were ways and nodes.

In [3]:
```
Image(filename=r"C:/Users/Drew/Desktop/Code/DAND/Wrangle/Images/node_way_vis.jpg")
```

Out[3]:



As you can see above, our top contributor, 'n76' doubled the contributions of our second place contributor but was not the highest contributor of 'ways'!

It was clear that the top 4 users contributed significantly more than any of the other users so I thought I would use a pie chart to help illustrate just how much more. I settled on 3 groups, the top 10 contributors, the next 100 contributors, and finally the remaining contributors (user rank 112 - 953, where rank is determined by the number of contributions made to the map).

Querying for Node/Way Contributions by User

This was a tricky one, I wanted to retrieve the node and way contributions seperately for each user in the top ten contributor list so I ran the following 2 queries

- Node Contributions

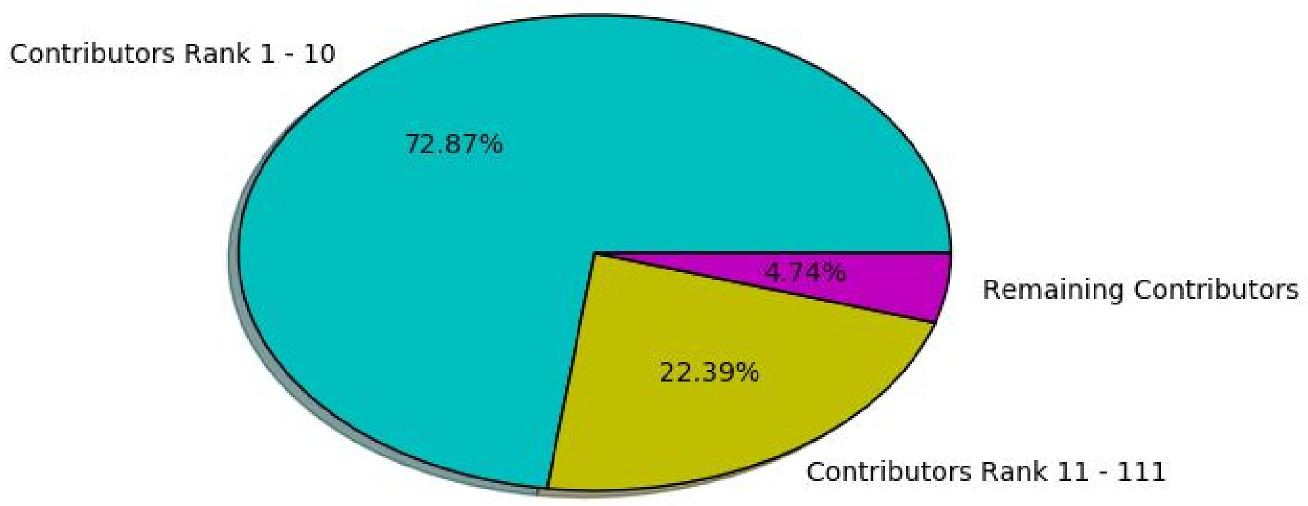
```
col.aggregate([{"$match": {"type": "node", "created.user":{"$in": top_ten_usernames}}},
{"$group": {"_id": "$created.user", "count": {"$sum": 1}}}])
```

- Way Contributions

```
col.aggregate([{"$match": {"type": "way", "created.user":{"$in": top_ten_usernames}}},
{"$group": {"_id": "$created.user", "count": {"$sum": 1}}}])
```

In [4]:
```
Image(filename=r"C:/Users/Drew/Desktop/Code/DAND/Wrangle/Images/contributor_proportion.jpg")
```

Out[4]:



As you can see above, 110 users (or 11.54% of users) account for over 95% of the map's contributions.

Querying for the data

This was the query I used to retrieve all the users and the number of contributions they made. Since the results were sorted, I was then able to break these up into the 3 groups in the visualize above using the python slice operator.

```
col.aggregate([{"$match": {"{}".format("created.user"): {"$exists" : True}}},
{"$group": {"_id": "${}".format("created.user"), "count":{"$sum": 1}}},
{"$sort": {"count": -1}}])
```

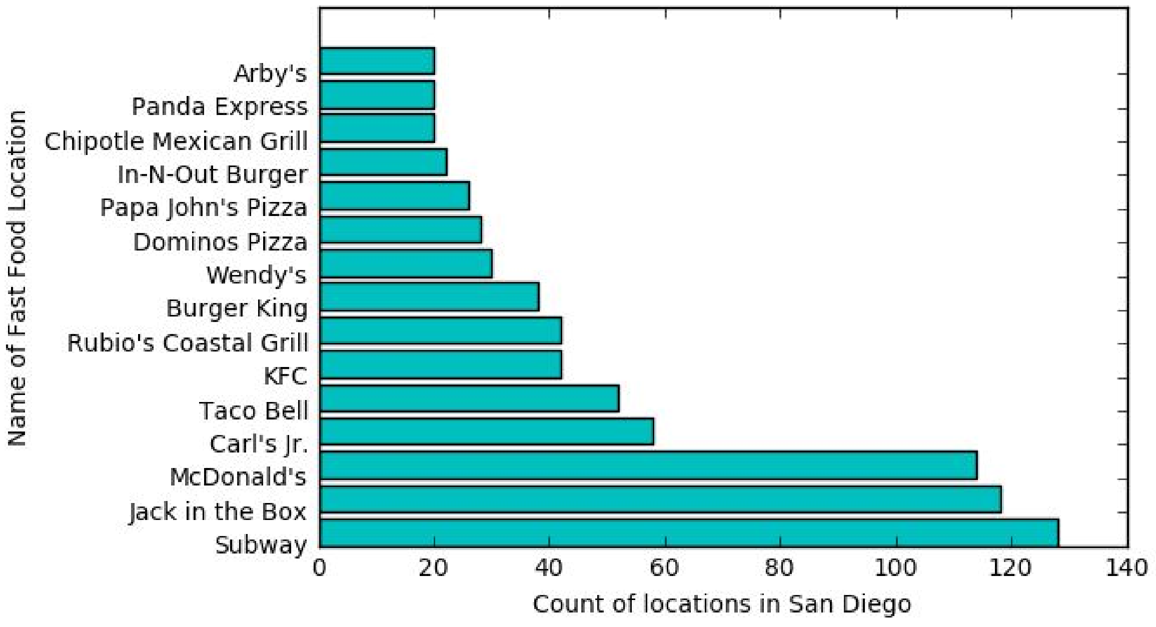
Overview of Fast Food

I thought it would be a fun idea to look at the fast food in San Diego and answer the following questions:

- What is the most common franchise?
- What is the most common cuisine (i.e. burger, pizza, mexican, etc)
- In each of these cuisine types, which franchises are most common?

In [5]:
```
Image(filename=r"C:/Users/Drew/Desktop/Code/DAND/Wrangle/Images/ff_freq.jpg")
```

Out[5]:



Looks like **Subway**, the world's most common fast food franchise, also has the most locations in San Diego! Next is San Diego based company, Jack in the Box, I suppose that is not too surprising.

Querying for the data

In order to get the data to form the visualization above, I had to gather all fast food documents with names, group them on the name field, and count them. Then since there were too many results to display neatly in one plot, I sorted the results (descending) and limited the number of results to 15.

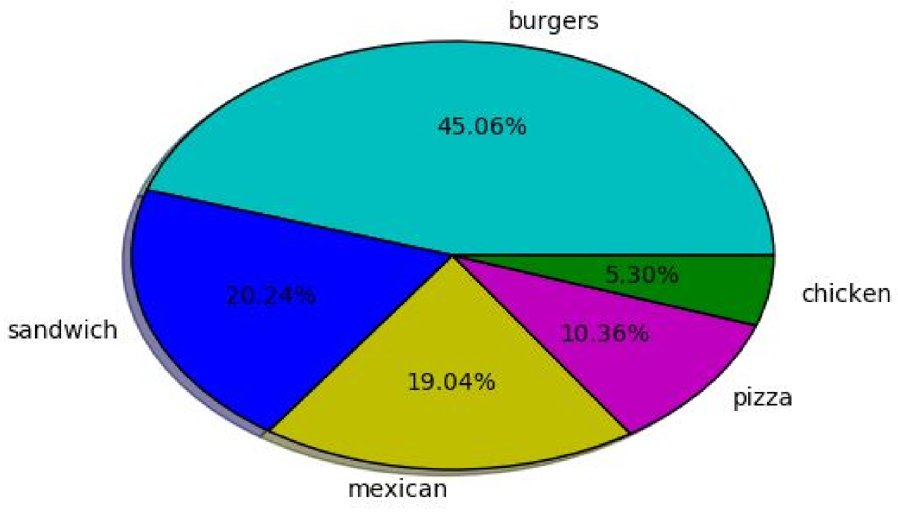
Here is that query:

```
col.aggregate([{"$match": {"amenity": 'fast_food', "name":{"$exists": True}}},
{"$group": {"_id": "$name", "count": {"$sum":1}}},
{"$sort":{"count":-1}},
{"$limit": 15}])
```

In [6]:

```
Image(filename=r"C:/Users/Drew/Desktop/Code/DAND/Wrangle/Images/ff_proportions.jpg")
```

Out[6]:



Burger joint's account for 45% of San Diego fast food locations! While, 20% are sandwich shops. I am willing to guess that Subway makes up a large portion of that 20% itself while the burger distribution is likely shared amongst Jack, McDonalds, and Carl's Jr.

Querying for the data

```
col.aggregate( [{"$match": {"amenity": "fast_food", "cuisine": {"$exists": True}}},
{"$unwind": "$cuisine"},
{"$group": {"_id": "$cuisine", "count": {"$sum": 1}}},
{"$sort": {"count": -1}},{"$limit": limit}])
```

You will notice I had to use unwind because some of the fast food entries fit under multiple cuisine's (like Carl's Jr. / Green Burrito)

We can see above that Jack in the Box and McDonald's dominate the fast food burger market in San Diego.

Other ideas about the dataset

Adding GeoCache Locations to OSM

Being able to analyze where people tend to hide/find geocache's around San Diego would be interesting data to use to create a heat map. Perhaps local businesses would sponsor geocaching (or simply start placing geocaches near their store fronts) to help increase foot traffic in their area.

Potential Problems A potential problem implementing this is scraping all the data necessary to add these nodes. There are several different websites and app that geocachers use to communicate where they hide and find geocaches and so it would be very easy to have an incomplete and out of date entries.

Adding SurfSpot Locations to OSM

I noticed when I grabbed a list of unique tags in the initial audit that there were fields for beach parking but I never saw anything relating to surf spots. Often times (especially in San Diego) some of the best breaks can be a long walk from the car. Adding more accurate paddle out points could be very useful for surfers moving to or visiting San Diego.

Potential Problems Surf spots often have multiple breaks that can be spread out enough to warrant multiple nodes. I don't think it would be accurate to denote a surf spot as a way, so how would we determine where to actually place the location? Is the parking lot not our best option (as most people can track down where to walk from there)? What if the break does not have a parking lot (as is the case with some spots within state park boundaries)? These would likely be left up to the user uploading these node's discretion.

Conclusion

I had a great time with this exercise, it was fun to do a real life sample, audit, cleaning, and analyze cycle. It was clear that the OpenStreetMaps probably did not have all the Fast Food locations (as compared with something like Google Maps) but it was still fun to play around and see what analysis could be done.