OpenStreetMap Data Analysis for San Diego, CA

Why San Diego?

I have traveled to many places but San Diego is one of my favorite cities. This project provides an unprecedented opportunity for me to dig deeper into the city. The following link was used to retrieve the data:

https://mapzen.com/data/metro-extracts/metro/san-diego_california/

Parsing the Data Using the ElementTree XML API

First, let's get an overall idea of the size of our dataset by counting the number of tags in the xml file. This is done using the script *tags.py*. Here's the output:

"defaultdict(<type 'int'>, {'node': 1041623, 'nd': 834206, 'bounds': 1, 'member': 13500, 'tag': 2556910, 'relation': 767, 'way': 94784, 'osm': 1})"

This is a pretty large dataset. We'll do similar analysis using sqlite queries later in the project.

We can also explore the data a bit more to see if there are tags with problematic characters. This is also done using the script *tags.py*. Here's the output:

"{'lower': 648901, 'lower_colon': 1871409, 'other': 36595, 'problemchars': 5}"

We have 5 tags with problematic characters.

Problems Encountered in the Map

1) Inconsistencies in the street name. The script *audit.py* lists all different formats that have been used to name the streets. There are several inconsistencies and abbreviated formats in the street names:

Third Ave.
Dewes Wy
Air Wy
Balboa Ave
Garnet Ave
Magdalena Ave
10th Ave
Adams Ave
Girard Ave
Woodside Ave
Fay Ave
Orange Ave
Lake Murray Blvd.
Shelter Island Dr.
W Mission Bay Dr
Radcliffe Dr

Amaya Dr
Center Dr
Complex Dr
Grossmont Center Dr
North Harbor Dr
1710 W Mission Bay Dr
Starling Dr
Midway Dr
S Greensview Dr
La Jolla Village Dr
Clubhouse Dr
Harbor Island Dr
Judicial Dr
Murray Dr

It looks like we have some cleaning to do. The function “update_name()” in the script cleaning.*py* maps the street names to the appropriate names. The clean data is then written into the csv files. Here’s an example of cleaned data:

Third Ave. => Third Avenue
Dewes Wy => Dewes Way
Air Wy => Air Way
Balboa Ave => Balboa Avenue
Garnet Ave => Garnet Avenue
Magdalena Ave => Magdalena Avenue
10th Ave => 10th Avenue
Adams Ave => Adams Avenue
Girard Ave => Girard Avenue
Woodside Ave => Woodside Avenue
Fay Ave => Fay Avenue
Orange Ave => Orange Avenue
Lake Murray Blvd. => Lake Murray Boulevard
Shelter Island Dr. => Shelter Island Drive
W Mission Bay Dr => W Mission Bay Drive
Radcliffe Dr => Radcliffe Drive
Amaya Dr => Amaya Drive
Center Dr => Center Drive
Complex Dr => Complex Drive
Grossmont Center Dr => Grossmont Center Drive
North Harbor Dr => North Harbor Drive
1710 W Mission Bay Dr => 1710 W Mission Bay Drive
Starling Dr => Starling Drive
Midway Dr => Midway Drive
S Greensview Dr => S Greensview Drive
La Jolla Village Dr => La Jolla Village Drive
Clubhouse Dr => Clubhouse Drive
Harbor Island Dr => Harbor Island Drive
Judicial Dr => Judicial Drive
Murray Dr => Murray Drive

2) Inconsistencies in the phone numbers. A quick look at a small section of the data reveals that several formats have been used to list the phone numbers:

619-291-6443
+1-619-232-3242
+1 619 291 8287
619.264.2072
+16197027160
(619) 582-0136
619-284-8730
619-422-4141
619-232-4737
619-686-8700

(619) 482-7361
+1 858 499 0202
619-422-6414
619-420-6040
18584881191
+1 619 294 7483
+619-692-4910
(619) 475-9880
+1 619 232 3101
+1-858-454-3541

The phone numbers have been updated to 1) keep the digits (hyphens, dots, and other characters were dropped), and 2) drop the country code if it exists. Note that phone numbers can be presented using characters or a combination of characters/numbers to help users memorize them. In that case, decoding of the characters would be necessary. The code currently returns the original value if there are less than 10 digits in the phone value. The clean data were used to write the csv files. The function "update_phone()" within the script *cleaning.py* does this for us. Here's how the output would look like:

619-291-6443 => 6192916443
+1-619-232-3242 => 6192323242
+1 619 291 8287 => 6192918287
619.264.2072 => 6192642072
+16197027160 => 6197027160
(619) 582-0136 => 6195820136
619-284-8730 => 6192848730
619-422-4141 => 6194224141
619-232-4737 => 6192324737
619-686-8700 => 6196868700
(619) 482-7361 => 6194827361
+1 858 499 0202 => 8584990202
619-422-6414 => 6194226414
619-420-6040 => 6194206040
18584881191 => 8584881191
+1 619 294 7483 => 6192947483
+619-692-4910 => 6196924910
(619) 475-9880 => 6194759880
+1 619 232 3101 => 6192323101
+1-858-454-3541 => 8584543541

Importing CSV Files into SQL Databases

Next, we need to import the csv data into a sqlite database to execute some queries. The scripts *create_db.py* creates a sqlite database containing five different tables namely: "nodes_tags", "ways_tags", "nodes", "ways", and "ways_nodes".

Data Overview and New Insights

To obtain an overall overview of the database, I first ran some basic queries. The script *db_query.py* contains commented queries.

1) The total number of nodes: 1041623

SQL query: SELECT COUNT(*) FROM nodes COUNT

2) Total number of ways: 94784

SQL query: SELECT COUNT(*) FROM ways

This data is consistent with what we had before.

3) Total number of unique users: 1077. This is a significant number!

SQL query: SELECT COUNT(DISTINCT(u.uid)) FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) u

4) List of top 10 users: ('n76', 334892), ('Adam Geitgey', 158974), ('Sat', 125254), ('woodpeck_fixbot', 90764), ('TheDutchMan13', 26178), ('Zian Choy', 16856), ('Brian@Brea', 15758), ('TieFaith', 12902), ('stevea', 12506), ('evil saltine', 11942)

SQL query: SELECT u.user, COUNT(*) as num FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) u GROUP BY u.user ORDER BY num DESC LIMIT 10

We see that a single user (the top user) has had more than 300,000 contributions. This user is likely a robot. However, there is no tag in the xml file that distinguishes humans from robots. I think it would be very helpful to include a new tag into the xml files to distinguish human inputs from automated inputs. Such information would motivate many real users to contribute to the OSM project. Perhaps some machine learning algorithms can automatically do this on the current set of data. Of course, this would be a challenging task because not all users would input truthful information here, i.e. they would use robots but imply otherwise (this is a common problem in other similar web services). A potential solution would be implementation of the verifications mechanisms for humans that robots could not pass through. This way, robots we won’t require any verifications for robots as long as they adhere to the protocol. Alternatively, we could let all users provide their inputs. We can then use machine learning algorithms to distinguish real users from robots. We could also notify the users about our decision, i.e. they’ve been categorized as “robot” or “human”. This way we would give them a chance to dispute our decision.

5) List of top 10 amenities: ('place_of_worship', 915), ('fast_food', 538), ('restaurant', 497), ('school', 299), ('bar', 275), ('cafe', 176), ('fuel', 107), ('bank', 83), ('drinking_water', 73), ('bench', 71)

SQL query: SELECT value, COUNT(*) as num FROM nodes_tags WHERE key="amenity" GROUP BY value ORDER BY num DESC LIMIT 10

6) I like “Chipotle Mexican Grill”, so I decided to calculate the total number of Chipotle restaurants in San Diego. There are 26 Chipotle in San Diego. That’s a good news!

SQL query: SELECT COUNT(*) FROM nodes_tags WHERE UPPER(value) LIKE UPPER("%Chipotle%")

- To see if this dataset is actively being updated, I counted the number of inputs that has been made since the beginning of the June, 2017. There has been ~30,000 inputs within the last month. This indicates that the dataset is being updated, that’s also a good news!

SQL query: SELECT COUNT(*) FROM (SELECT timestamp FROM nodes WHERE timestamp >= "2017-06-01" UNION ALL SELECT timestamp FROM ways WHERE timestamp >= "2017-06-01