OpenStreetMap Data Case Study

Yilin Du

Map Area

San Jose, CA, United States

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

It's where I live now. So I would like to discover more on this area

Problems encountered during the data wrangling

1. Redundant key for street names: similar to the quiz, the tag "key" value contains ":" for address, key = "addr: street: name" .

2. Phone numbers are in different format: the following different formats of valid phone numbers are found
    a. (1)(408) 766-7000',
    b. '(408) 262-0401',
    c. +1 (408) 245-8434',
    d. '+14089829040',
    e. '1408-251-6900',
    f. '+1-408-245-5330',
    g. '+14089829040',
    h. '1408-251-6900',
    i. '+1.408.245.5432'

3. Invalid or incomplete phone numbers:
    a. ['+1',
    b. '2924779',
    c. u'\u200e+1 408 279-4225',
    d. u'\u200e+1 408 292-4300'])

4. For the tags with "key" value as "building", there are duplicate/similar record names. Like "garages" and "garage", "train" and "train_station", "truck" and "food_truck".

Programmatically Data Cleaning

1. Clean up key = “addr: street: name” to key = “street: name” and type = “addr” when import the raw data into csv. (see data.py)

```
key = child.attrib['k'].split(':', 1)
if len(key) > 1:
    dict['type'] = key[0]
    dict['key'] = key[1]
else:
    dict['type'] = 'regular'
    dict['key'] = child.attrib['k']
```

2. Clean up the phone number to format: 1-888-888-8888

```
# format "18005555555" as '1-800-555-5555'
def phone_format(number):
    return format(int(number[:-1]), ",").replace(",", "-") + number[-1]


# Auditing Validity: check is a string is a valid phone number, and return
# a bool value and the string representation of that phone number
def phone_is_valid(phone_number):
    digit_string = filter(str.isdigit, phone_number)
    if len(digit_string) == 10:
        digit_string = "1" + digit_string # add country code
    if len(digit_string) == 11: # valid phone number
        format_num = phone_format(digit_string)
        return True, format_num
    return False, phone_number
```

3. Create a map to update similar building names

```
def update_building(name):
    map = {
        'garages': 'garage',
        'train': 'train_station',
        'truck': 'food_truck'
    }
    if name in ('garages', 'train', 'truck'):
        return map[name]
    else:
        return name
```

Overview of the Data

Size of the osm file: 277 mb

Number of nodes: 1313063

```
SELECT COUNT(*) FROM nodes
```

Number of ways: 174403

```
SELECT COUNT(*) FROM ways
```

Number of distinct users: 1312

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e
```

Number and Kind of "Leisure" Place

```
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key = 'leisure'
GROUP BY value
ORDER BY num DESC
LIMIT 10
```

[(u'swimming_pool', 147),
(u'playground', 121),
(u'park', 65),
(u'picnic_table', 38),
(u'sports_centre', 30),
(u'pitch', 28),
(u'fitness_centre', 16),
(u'slipway', 8),
(u'dance', 6),
(u'dog_park', 3)]

Which "way" has the most nodes in it

```
SELECT id, COUNT(*) AS num
FROM ways_nodes
GROUP BY id
ORDER BY num DESC
LIMIT 10"
```

[(50372540, 1913),
(185551839, 1899),
(140642538, 1824),
(38790356, 1798),
(185551777, 1546),
(33106235, 1380),
(185551739, 1372),
(51960836, 1283),
(185551700, 1223),

(33106359, 1071)]

more than 1000 nodes in one way , that's a lot!

Idea for Additional Improvements

Let's query for the top 10 building types in this area:

```
SELECT value, COUNT(*) AS num
FROM ways_tags WHERE key = 'building'
GROUP BY value
ORDER BY num DESC
LIMIT 10
```

[(u'yes', 83969),
(u'house', 7041),
(u'residential', 4000),
(u'apartments', 506),
(u'school', 421),
(u'roof', 409),
(u'commercial', 223),
(u'office', 181),
(u'garage', 177),
(u'terrace', 175)]

And query for the top 10 most seen maximum speed here:

```
SELECT value, COUNT(*) AS num
FROM ways_tags WHERE key = 'maxspeed'
GROUP BY value
ORDER BY num DESC
LIMIT 10
```

[(u'sign', 2327),
(u'35 mph', 1539),
(u'40 mph', 1385),
(u'25 mph', 1000),
(u'45 mph', 624),
(u'30 mph', 537),
(u'65 mph', 444),
(u'survey', 388),

From the query results, we can see that the top 1 building types is "yes" and the "maximum speed" is "sign". These are not what we expected. Because "yes" specified the "tag" IS a "building" but there is no additional information on what kind of building it is. The same for maximum speed. We want to know the number of each speed sign instead of the truth that the "maxspeed" is a ''sign". So I suggest to have an additional tag in the dataset to specified the type. So we'll see <tag k = 'marker' v = 'building'> or <tag k = 'marker' v = 'sign'> in these records. It helps to specify the type of the nodes

and avoid the 'marker' information exists in the field where we actually expect a value. When implement the additional 'marker' tags, I think the main challenge is to have a standard to generate the value for all the markers.

UPDATED: To think out of the box, we can use the other popular website service to improve the dataset. Say, through the Yelp API, we can connect to the business owners or restaurant in Yelp.

Benefits:

1. By connecting to Yelp, we can update some newly open restaurants.
2. From the analysis in the previous section, I found some of the telephone numbers are invalid and it is also possible some of the telephone numbers are not accurate. We can cross check the basic information for the business like telephone number, address, fax number, website with the information published in Yelp. I believe the information in Yelp should be more accurate as they are provided and updated by the business owners.

Anticipated Problems:

1. Anyone can create a business in Yelp and the information for those unclaimed business may not be accurate.
2. The information in Yelp is also limited. It may help with some business and place of interest. But it's hard for us to find any changes of the road or constructions. To improve this aspect. We may need to use Google Maps API as well

Conclusion

In this project, I'm trying to explore the dataset for san jose. I cleaned up a little bit of the street name, phone number and some leisure place names. There are more to explore here. Similar to phone number, I guess the postal codes are also quite "dirty", we can explore different formats of the postal codes and clean up some invalid values. There are should be duplicate/similar naming everywhere: upper/lower cases, different naming conventions. It'll be great to contribute the toe OpenStreetMap.org in the future and be one of the user with top contribution percentage.