Photo by Phoenix Parks Foundation.
http://phxparksfoundation.org/

Phoenix OSM Project

“THE VALLEY OF THE SUN”

Abdulrahman Alemadi | Udacity DAND | October 20, 2017

Intro

This is the 4th project of Udacity's Data Analysts Nanodegree. The goal of the project is to choose a city from the Open Street Map database, to wrangle and shape it to a SQL database, so we can use the data for analysis. This project is important because data analysts spend most of their time wrangling data.

I chose the city of Phoenix, AZ as the subject of this project, for several reasons, I have had lived there for 5 years, therefore I have a good knowledge of the area. Initially, I wanted to do on my hometown, Doha, but after I downloaded the raw data it looked that the data has not been updated in a while, also I have noticed that I will have a hard time with the Arabic letters in the data, so I resorted to do the project on Phoenix.

This is the most challenging project I have ever done. I felt like quitting numerous times throughout the project, but I am glad that I pulled through. As a result. I felt like I learned tons of new stuff like:

- XML
- HTML
- JSON
- Using APIs
- Data Wrangling
- SQL

I am going to talk about:

- Raw data statistics
- Problematics in the data
- The data wrangling process.
- Suggestions for how to improve the data.
- Statistical Queries about the city of Phoenix.

Raw Data Statistics:

The count of all different XML tags are the following:

```
{'node': 318952, 'member': 27051, 'nd': 407790, 'tag': 279359,
'bounds': 1, 'note': 1, 'meta': 1, 'relation': 837, 'way': 51849, 'osm': 1
```

The top 20 contributing users with their number of contributions are the following:

```
[(u'Dr Kludge', 120221), (u'TheDutchMan13', 43895),
(u'\u042e\u043a\u0430\u0442\u0430\u043d', 30552), (u'lukas64', 23688),
(u'jfuredy', 15633), (u'woodpeck_fixbot', 9543), (u'Adam Martin', 7948),
(u'dwh1985', 7399), (u'adenium', 6793), (u'kookiemonster', 6727), (u'AJ
Riley', 5017), (u'Bike Mapper', 4810), (u'horndude77', 4419), (u'pflier',
4390), (u'Rub21', 3416), (u'maponpoint33', 3048), (u'Sundance', 3002),
(u'danielmoberly', 2729), (u'samely', 2382), (u'CorranHorn', 2237)]
```

Below I will use Regex to find how the data is formatted. 3 patters are provided:

1- "lower", for tags that contain only lowercase letters and are valid.

2- "lower_colon", for otherwise valid tags with a colon in their names.

3- "problemchars", for tags with problematic characters

```
{'lower': 187453, 'lower_colon': 88074, 'other': 3831, 'problemchars': 1}
```

Other Problematics:

I noticed that the phone numbers are not unified, these are some examples of the formats in the data:

- (480) 897-8080

- +1 480 412 5437
- 4808200333

Another problem was the street suffixes were not unified such as:

- ‘St’
- ‘St.’
- 'Ave'
- 'Blvd
- 'Pkwy’
- 'Rd.'
- 'Rd'
- 'Dr'
- 'Rd,'

Also, there was a considerable amount of data was entered in the wrong section. Such as house apartment numbers in the street address section. This made my regex function to pick up data that were not street suffixes.

Data Wrangling Process:

This part I used this function template that were supplied by Udacity then modified it to suite my needs. The function takes in the names of the streets and the mapping. The function fixes 2 problems by:

1- Removing APT and house numbers from the street address. So, **S Alma School Rd #112** becomes **S Alma School Rd.**

2- Using a regex function to search for problematic names, then modifies it according to the given mapping. So, **S Alma School Rd** becomes **S Alma School Road**.

```
    def update_name(name, mapping):
        if '#' in name:
            name = name.split('#',-1)[0].strip()
        if ',' in name:
            name = name.split(',', 1)[0].strip()
        m = street_type_re.search(name)
        if m:
            street_type = m.group()
            if street_type not in expected:
                if street_type in mapping:
                    name= re.sub(street_type_re, mapping[street_type] , 
name)
        return name
```

Since the phone numbers' format were not unified, I made this function to solve the problem:

```python
def std_phone(num):
    '''
    Standardizes phone number
    '''


    # Remove anything that is not a number
    num = re.sub('[^0-9]+', '', num)
    # add dashes to 10 digit nums
    if re.match(r'^\d{10}$', num) is not None:
        num = num[:3] + '-' + num[3:6]+ '-' + num[6:]
        fixed_num.append(num)
    # add dashes to 11 digit nums
    elif re.match(r'^\d{11}$', num) is not None:
        num = num[0] + '-' + num[1:4] + '-' + num[4:7] + '-' +num[7:]
        #fixed_num.append(num)




        return num
```

Suggestions for how to Improve the Data:

One cool thing that could be implemented is to assign a heatmap score to the locations. For example, the more the place has more people the higher the score of the location. I think this Google API could be used for that purpose. The benefit of this is we can rank amenities and restaurants based on location 'hotness'. One anticipated problem can occur is the complexity of implementing an objective 'score system'. Traffic jams can be misinterpreted as a 'hot' area.

Another suggestion is to add a field called 'distance of closest alternative' to emergency services locations. We can implement that by using Euclidean distance formula, which taking the difference between location A's and B's longitude and latitude. One benefit of this could be for in case of a natural disaster, we could find out the nearest alternative hospital if the other one is full or has been damaged. Another use is for city planners to see if they have sufficient emergency services (fire, police and healthcare) in each area, and it will help them to decide where are these services are needed most. The problem of implementing this, is there will always be a error margin with the Euclidean distance method. Because, it calculated the shortest direct distance without, accounting for obstacles. For instance, a police station could be in a mountain, and say its closest alternative is 5 miles away, but with the nature of terrain it will take 15 miles driving. The Euclidian distance does not account for the 'actual' drive, but only for the latitude and longitude of a given location.

Lastly, OSM could possibly implement a suggestion box that pops up when entering addresses such as 'did you mean Street, instead of st.'. That could save a lot of time by streamlining the data. Or even better, they could implement 'rules of data input' where they have specific rules for to unify all data entry. One problem with this idea is it will be hard to convince people to stick to a certain convention. Also, if we account for other languages, it will only increase the complexity of this problem by multiple folds.

Statistical Queries about the city of Phoenix:

Highway Stats:

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags) i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='highway'
GROUP BY nodes_tags.value
ORDER BY num DESC;

"crossing"      "3879"
"turning_circle""3744"
"traffic_signals"       "1293"
"street_lamp"   "945"
"stop"  "277"
"bus_stop"      "255"
"motorway_junction"     "141"
"turning_loop"  "69"
"give_way"      "9"
"elevator"      "8"
"speed_camera"  "1"
```

Amenities Stats:

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags) i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='amenity'
GROUP BY nodes_tags.value
ORDER BY num DESC;

"restaurant"    "183"
"fast_food"     "155"
"fuel"  "147"
"place_of_worship"      "108"
"school""82"
"cafe"  "37"
"bank"  "35"
"pharmacy"      "29"
"bench" "28"
"atm"   "25"
"trailer_park"  "25"
"drinking_water""24"
"toilets"       "24"
"bar"   "22"
"car_wash"      "16"
"waste_disposal""16"
"pub"   "14"
"vending_machine"       "13"
"fire_station"  "12"
"post_office"   "12"
"fountain"      "11"
"waste_basket"  "11"
"parking"       "9"
"bicycle_parking"       "8"
"post_box"      "8"
"swimming_pool" "7"
"clinic""6"
"dentist"       "6"
"police""6"
"bicycle_repair_station""5"
"doctors"       "5"
"library"       "5"
"telephone"     "5"
"bbq"   "4"
"grave_yard"    "4"
"office""4"
"car_rental"    "3"
"charging_station"      "3"
"cinema""3"
```

```
"community_centre"	"3"
"courthouse"	"3"
"hospital"	"3"
"public_building"	"3"
"shelter"	"3"
"theatre"	"3"
"university"	"3"
"arts_centre"	"2"
"college"	"2"
"nightclub"	"2"
"parking_entrance"	"2"
"social_facility"	"2"
"veterinary"	"2"
"bicycle_rental""1"
"casino""1"
"coworking_space"	"1"
"dojo"	"1"
"gym"	"1"
"kindergarten"	"1"
"marketplace"	"1"
"prison""1"
"recycling"	"1"
"studio""1"
```

Leisure Stats:

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags ) i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='leisure'
GROUP BY nodes_tags.value
ORDER BY num DESC;

"swimming_pool"	"534"
"park"	"42"
"playground"	"15"
"sports_centre"	"7"
"picnic_table"	"6"
"fitness_centre""5"
"pitch"	"4"
"garden""3"
"park_bench"	"2"
"slipway"	"2"
"dog_park"	"1"
"hackerspace"	"1"
"ice_rink"	"1"
"track"	"1"
```

Conclusion:

This project requires all basics of data analysis from data exploration, to data munging into csv files and then finally to into a SQL Database, so we could analyze it. These skills are essential for anyone who wants to work with data. This project has truly pushed my limits, a month ago I was overwhelmed by all the new information I had to absorb. I learned tons thanks to the Udacity mentors, online sources such as Datacamp, Dataquest and stack overflow.