UDACITY

Data Analyst Nanodegree
Project P3
Wrangle an OpenStreetMap dataset

Submitted by Narendran Santhanam

Introduction

I work in an office located in downtown Manhattan. Owing to the large number of businesses in this area and the high amount of footfall, I believe this area has a lot of useful data that can be analyzed to gain some insights. Here is the link to the area that was used for the project: http://www.openstreetmap.org/export#map=14/40.7195/-73.9858

Problems in the map data

Investigating the downloaded data uncovered the following issues / inconsistencies:

1. Abbreviated street names / types

Investigating the data before loading the database revealed that there were a lot of abbreviated street names and types. The data also revealed that the abbreviated directions were included in the street names like "W. Broadway". I fixed them using a mapping dictionary such as:

```
abbr_names = {
    "st":"Street",
    "ste":"Suite",
    "ave":"Avenue",
    "brg":"Bridge",
    "ct":"Court",
    "rd":"Road",
    "pl":"Place",
    "dr":"Drive",
    "sq":"Square",
    "ln":"Lane",
    "blvd":"Boulevard",
    "hwy":"Highway",
    "wy":"Way",
    "plz":"Plaza",
    "ctr":"Center",
    "n":"North",
    "w":"West",
    "s":"South",
    "e":"East"
    }
```

2. Inconsistent zip codes

Analyzing the zip codes uncovered quite a number of issues. Most of the zip codes were accurate and correctly formatted, but there were a few with the extended 4-digit zip code, there were some which included the state code and there was one with incorrect zip code ('100014'). I restricted all zip codes to just the first five digits for consistency. The most interesting finding was that because of the map area selected, some of the zip codes belonged to New Jersey. I added a flag to the element for such cases to make sure they don't make it to the database.

3. Cities and counties

Looking through the file, I found that cities and counties were found in two types of tags: one, the tiger tags ('tiger:county') and the address tags ('addr:city'). Analysis of these tags revealed that there were:

1) Inconsistencies in how cities were represented – Incorrect capitalization, neighborhoods represented as cities, state name included in city name, etc. Examples: 'York City', 'Tribeca', Manhattan NYC'
2) Cities and counties from New Jersey (Hudson, Jersey City) included in the map data

4. Inconsistently formatted and inaccurate phone numbers

Many of the businesses had phone numbers included in the XML file. Analysis of the phone numbers revealed various different formats in which the numbers have been entered. Here is a set of all the different characters available in the phone number data:

```
[' ', ')', '(', '+', '-', '.', '1', '0', '3', '2', '5', '4', '7', '6', '9', '8'])
```

I reformatted all phone numbers to +19999999999 format before loading them to the database. There were a couple of businesses that had one digit missing in the phone number. I looked them up externally and fixed them.

Overview of the data

1. File sizes

Here are the sizes of the files after loading the database.

```
NYC.OSM          - 70,825 KB
OpenStreetMap.db - 41,538 KB
nodes.csv        - 22,444 KB
nodes_tags.csv   - 2,527 KB
ways.csv         - 2,892 KB
ways_nodes.csv   - 8,906 KB
ways_tags.csv    - 8,126 KB
```

2. No. of nodes, ways and users

Let us write some queries to check the no. of nodes, ways and the no. of unique users in the database.

```
# No. of nodes
query = 'SELECT COUNT(*) FROM NODES'
print "No. of nodes: ", execute_query(c,query)[[0]][0][0]

# No. of ways
query = 'SELECT COUNT(*) FROM WAYS'
print "No. of ways: ", execute_query(c,query)[[0]][0][0]

# No. of distinct users from both nodes and ways tables
query = 'SELECT COUNT(*) FROM (SELECT DISTINCT(UID) FROM NODES UNION SELEC
T DISTINCT(UID) FROM WAYS)'
print "No. of unique users: ", execute_query(c,query)[[0]][0][0]
```

No. of nodes: 249935
No. of ways: 44323
No. of unique users: 1095

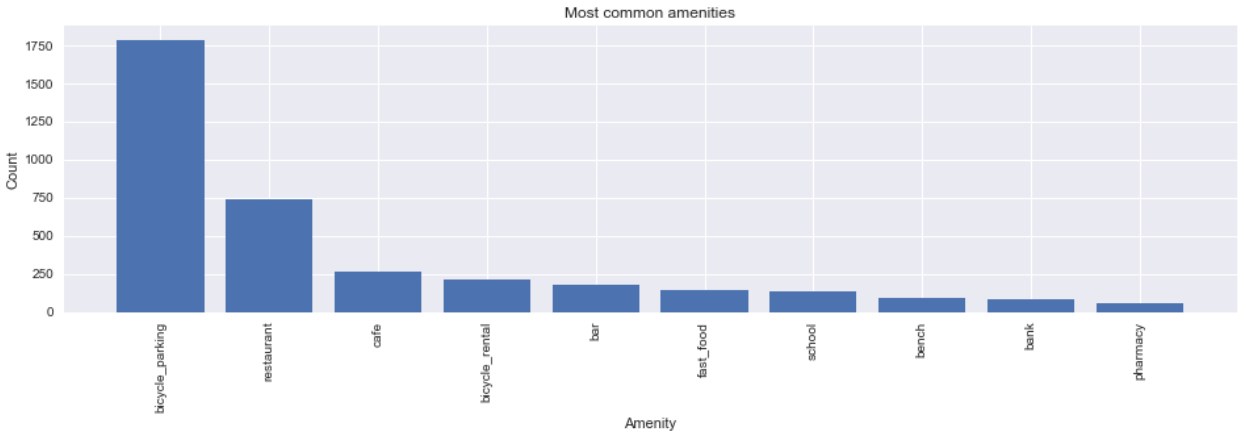
3. Additional analysis

a. Top amenities by count

Let us do some additional analysis of the data in the database. There are various types of amenities in the NYC area. Let's find out which ones are the most common.

```
query = 'SELECT VALUE, COUNT(*) FROM NODES_TAGS WHERE KEY="amenity" GROUP
BY VALUE ORDER BY COUNT(*) DESC LIMIT 10'
df=pd.DataFrame(execute_query(c,query))

groupByPlot(df,'Most common amenities','Amenity')
```



b. Bicycle parking capacity

Bicycle parking is the top amenity. What is the total number of bicycle parking spaces available?

```
query = 'SELECT VALUE FROM NODES_TAGS WHERE KEY="capacity" AND ID IN (SELE
CT ID FROM NODES_TAGS WHERE KEY="amenity" AND VALUE="bicycle_parking")'
df=execute_query(c,query)

print sum(pd.to_numeric(df[0]))
```

```
6460
```

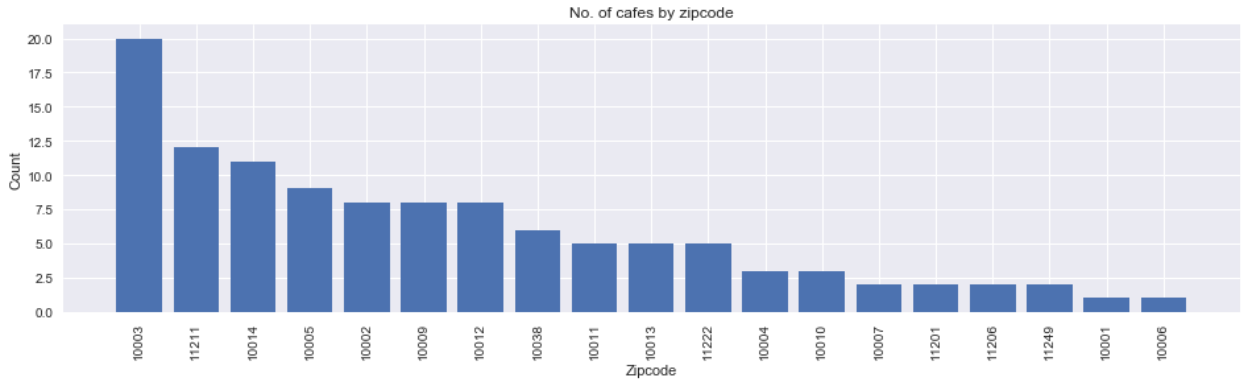
c. Restaurants, cafes and bars by zipcode

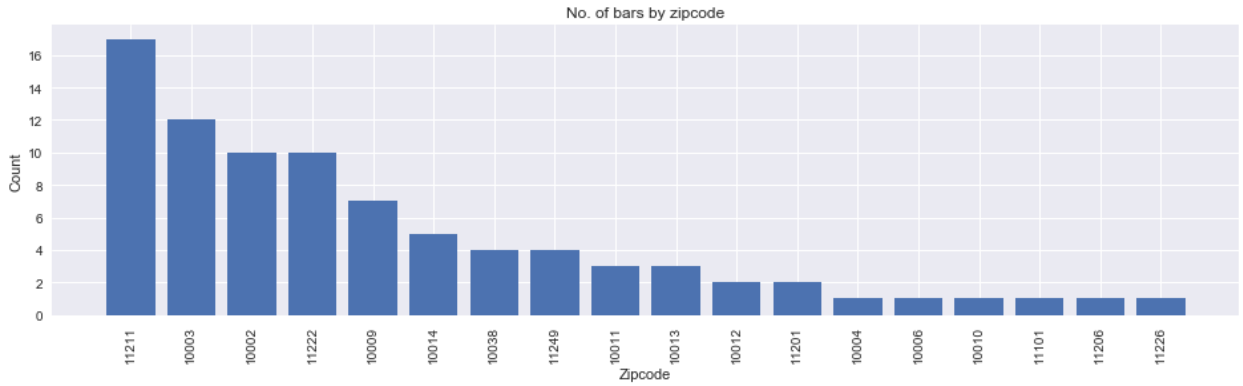
Which localities have the highest no. of restaurants, bars and cafes? Let us try grouping the amenities by zipcode.

```
def amenities_by_zipcode(c,amenity):
    query='SELECT VALUE, COUNT(*) FROM NODES_TAGS WHERE KEY="postcode" AND
ID IN (SELECT ID FROM NODES_TAGS WHERE KEY="amenity" AND VALUE="'+amenity+
'") GROUP BY VALUE ORDER BY COUNT(*) DESC'
    return execute_query(c,query)

amenities=['restaurant','cafe','bar']

for i in range(len(amenities)):
    plt.figure(i)
    df=amenities_by_zipcode(c,amenities[i])
    groupByPlot(df,'No. of '+amenities[i]+'s by zipcode','Zipcode')
```





We could see that downtown Manhattan, East Village and Williamsburg in the Brooklyn borough are very popular. While there are a lot of restaurants and cafes in downtown Manhattan, there are a lot of bars in the Williamsburg and East Village areas.

Improvements to the dataset and their benefits

1. Zip code data

One of the major improvements that I can think of is adding a zip code to each of the node tags. This will enable categorization of data based on zip code and it will be possible to mine better insights based on localities and neighborhoods. For example, even though I made efforts to filter out New Jersey data from the file, a few nodes and ways still made it to the database because the zip code data was not available. This was evident only when I plotted the data on a geo map.

US zip code geographical boundaries are publicly available as shapefiles in the URL specified below: (https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html). We should be able to integrate this data into OpenStreetMap since we already have the geographical coordinates available for each node.

2. Enhancing business information

I found that a majority of the businesses that are available in Google Maps, were actually missing from the OSM file. Since the user base of Google Maps is very high, businesses are inclined to update their data there, but not on OpenStreetMaps (I don't think a lot of people know about it, to be frank). Moreover, there is no "reward" system within OpenStreetMaps for contributors, which gives them little incentive to update information. Whereas in Google Maps, contributors get a "Local Guide" badge and different levels (and rewards) to go with it. This kind of gamification can definitely be a motivating factor for OSM contributors.

As seen in my queries above, there were some businesses with incorrect information in the XML file. I could fix it manually by looking up information from external websites. I am sure that this can be automated using a script that can scour webpages for information and update this in OpenStreetMaps. This would definitely make the data on OSM much richer than it is already, enabling users to derive more value out of it.

The business information coupled with zip code information will give a very good view of the most "happening" neighborhoods, and this view is something that any marketing department would love to have.

3. Satellite imagery

Satellite imagery is definitely something that can be leveraged along with machine learning algorithms to identify apartments, parking lots, sports amenities such tennis courts, baseball and football grounds, etc. This can then be validated against existing data where it is available, or added to nodes where information is not available. With improvements in resolution of the images, it may even be possible to figure out business names.

Drawbacks in implementing improvements

- The major drawback of gamifying the system is that there would be attempts to "automate" contributing to OSM, thus resulting in inaccurate data getting into OSM, just for the sake of getting a better rank on the leaderboard. It would be tough to design a validation mechanism, unless that too, is crowdscourced.
- The limiting factors in analyzing satellite imagery would be performance and costs. Analysis of images and running machine learning algorithms would definitely need immense processing power. An open source project such as OSM may not be able to afford such extreme technical infrastructure.