Travel Recommendation System

Yong Joon THOO

March 23, 2020

Note: The code and notebook (TravelRecommendation_notebook.ipynb) described in this report can be found at: `https://github.com/yjthoo/Travel-Recommendation-System`

1 Introduction

Planning a holiday or a weekend trip to another country or city can be a tedious task involving multiple hours of research via travel guides, websites and blogs. Furthermore, such guides are written from the point of view of other people who may not share the same concept of travel or fun as you do. Having a system that could take in the user's preferences based on what they liked during a previous trip or even based on areas they love in their current hometown and suggest areas to visit during their next trip could provide additional information to the user to help plan their trip.

This project presents a proof of concept for a travel recommendation system that takes in a user's preference in terms of location and suggests similar locations in their destination city.

2 Methodology

This section presents the methodology used within this project including datasets, Python libraries and the approach taken to recommend locations.

2.1 Data Collection

The approach taken to collect the required data is to first obtain the names of the neighborhoods or boroughs via webscraping of the relevant Wikipedia pages of the desired cities, in this case London[1] and San Francisco[2]. The coordinates of these boroughs/neighborhoods were then collected either through datasets such as in the case for London[3] or via the Nominatim library[4] which determines the coordinates of a provided address. The datapoints whose coordinates were either NA or 0.0, 0.0 (in the Atlantic Ocean) were then removed from the data, enabling us to display the coordinates of each location on a map by means of the Folium[5] library such as in figure .

[1]`https://en.wikipedia.org/wiki/List_of_areas_of_London`

[2]`https://en.wikipedia.org/wiki/List_of_neighborhoods_in_San_Francisco`

[3]`https://www.freemaptools.com/download/outcode-postcodes/postcode-outcodes.csv`

[4]`https://nominatim.org/`

[5]`https://python-visualization.github.io/folium/`

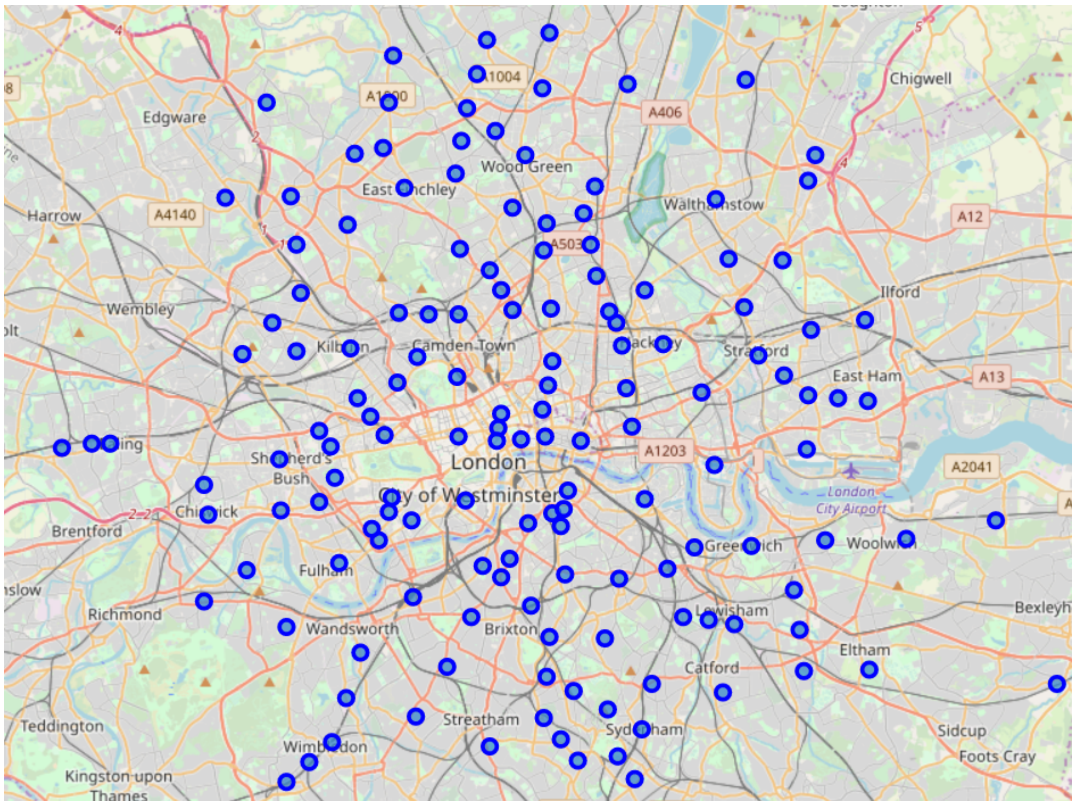


Figure 1: Illustration of the locations found within London

Once this was done, the venues within a given radius of each location were determined via the Foursquare API which provides data for venues, countries, etc. This enabled not only the determination of venues within a given radius of provided coordinates but also the categories of these venues. A score based on the number of venues for each category was then assigned to each location for the following stages.

2.2 Recommending similar locations

Once the processing stages are done, location recommendation can be done either by recommending similar locations within the same city or recommending locations in one city similar to a location in another city.

To recommend locations in the first case, a K-Means algorithm was performed to cluster locations with similar locations together. These were then displayed on a map with the points belonging to the same cluster displayed in the same colour. This would enable travellers to visit locations similar to those that they like in the same city. Additionally, another functionality that was implemented was to display locations that are most relevant to a given venue type/category on the map. This was done by determining the locations with the N highest scores determined previously for that venue type/category.

In the second case, we first need to determine the venue categories which are present in the data of both cities such that they are comparable. Assuming the first city is the one a person has already visited and the second one is the one the person wants to visit, the user needs to provide the name of the location within the first city that they liked and would like to see similarities to in the second city. The feature vector of the common categories for that location in the first city is then extracted as well as the feature vectors of all of the locations in the second city. We can then assign a score to all of the feature vectors in the second city by means of the Euclidean distance:

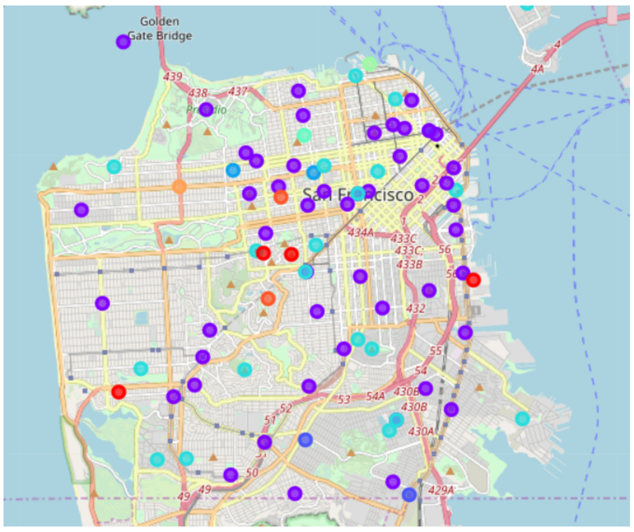
$$dist_{a_ib} = \parallel \overrightarrow{a_ib} \parallel = \sqrt{a_i^2 - b^2} \tag{1}$$

where a_i is the ith datapoint in the second city and b is the datapoint for the desired location in the first city.

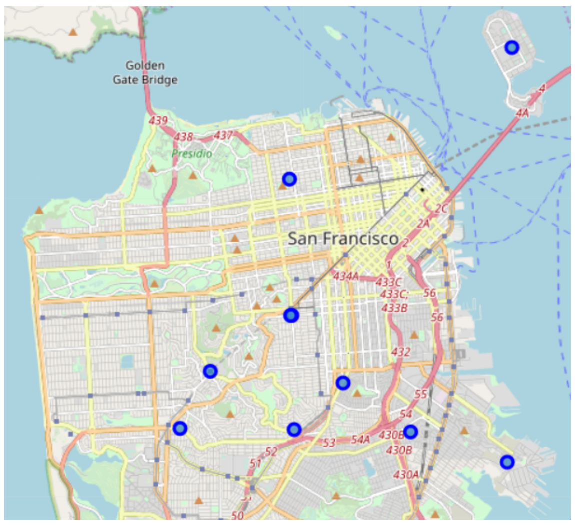
The datapoints in the second city for which the score is the lowest are then those that are the most similar to the location in the first city.

3 Results

The first case can be seen in figure 2a where K-Means has been applied to illustrate locations with similar venue types in the same color. The second case is illustrated in figure 2b where the top N locations that are similar to a location in another city are displayed.



(a) K-Means clustering illustrating locations with similar locations in San Francisco



(b) Display of locations in san Francisco with similar venues to a location in another city

Figure 2: Illustration of 2 of the recommendation types

An additional use case that is presented in the code is the ability to recommend a location to a user based simply on the venue type such as 'American Restaurant', 'Arcade', 'Boxing Gym', 'Museum', etc. This produces a map similar to the one displayed in figure 2b.

4 Discussion

One of the main weaknesses of the whole program comes from the radius within which to look for venues from a given location. For instance, if the radius is too small, we may only receive a small subset of the locations within that area that are not representative of the location. However if the radius is too large, it may become less specific to that location and even overlap the radius surrounding other locations such that they essentially represent the same venues.

Additionally, the feature vector of each datapoint/location is based on the average number of counts of each venue type. Although this does have the effect of suggesting the locations with the greatest presence of a given venue type, it is unable to reflect the overall feeling/atmosphere of a given area. For such feedback, it would be interesting to look at customer feedback of the venues and/or tourist locations within each area to better reflect what the area is really like.

Finally, the venue categories can be quite subjective and in some cases may not at all be representative of the venue such that the recommendation can be biased towards certain venue types and not others.

5 Conclusion

The code written for this project enables the recommendation and display of the coordinates of areas that may be of interest to a user. A first use case is to enable a user to discover similar locations within a city based on the venues within a given radius of each location and a second use case is to recommend locations based on their similarity to a location in another city.

Although the results illustrate the potential of such a travel recommendation tool, it should be noted that the data collected does not necessarily represent the overall feeling and atmosphere of each location such that the tool may be more likely to recommend locations with similar amounts of certain venue types. Future work would therefore involve the collection of data from reviews of tourist attractions, restaurants and other kinds of venues for a more stable recommendation system.