Final Capstone Project: The Battle of the Cities

Filipe S. M. Guimarães

Date: September 5, 2019

Abstract

In this project, we will compare different cities in a given list. The cities will be clustered using the k-means algorithm into different clusters. The program is set to be general, so it may be used to aid in any decision making related to going to a new city. The application created by this project helps any person going through this common procedure. Even though this project only gets information from the Foursquare API to categorize the different cities, it can easily be generalized to add new features in the future.

1 Introduction

Many times someone needs to go to a city — either for a temporary visit or a permanent move —, a choice between different places is possible. However, this decision is not easy as it involves many factors: size of the city, available restaurants and recreational places, prices and many others not always related to the place itself. These situations usually require a reasonable amount of research and analysis, and there are many available tools that can help in different phases of this process. Wikipedia, Google Flights, Booking.com, Google Maps and Foursquare are examples of instruments that are used to get familiar with a city, obtain forms and prices on how to get there, and the places available for staying and entertainment.

In all of these tools, the users need to analyze each option by themselves and then, after withdrawing weaker candidates or strengthening another one with the obtained information, make a verdict. Often, one still feels insecure about the choice made. In this sense, a tool that automatically groups cities in a given list, returning useful information about each group certainly assist the procedure. Such application can be used with unknown cities in an iterative way to reduce the possible alternatives, or also to include known cities as a reference.

With this comparison, we may answer questions such as:

- Is a big city with many restaurants more equal to a small city with many restaurants, or a big city with more parks?
- Which city is more like Dublin: Cologne or Amsterdam?
- Is Shanghai, Tokyo or Johannesburg more different than São Paulo?

In Chapter 2, we discuss the data that will be used. Chapter 3 is devoted to explain the methods used in this work. The analysis and build up of the functions is done in Chapter 4. These functions are then used in Chapter 5 to obtain results for different groups of cities. Finally, in Chapter 6 we summarize our findings and discuss possible ways to expand this work.

2 Data

To compare the different cities, we will obtain an extensive list of venues using the Foursquare API. The categories of the venues — restaurant, park, stadium, etc. — will be used to cluster the cities in different groups, which will be displayed on a map and the final *character* of each city will be printed out. The number of venues inside a given radius fetched from the API will also be used, and can be seen as a measure of the city density. Other types of data may be added later, to make the comparison more complete.

3 Methodology

Our goal is to compare different cities. With this intent, we are going to use K-Means clustering algorithm, which labels the cities in a chosen number of labels. Since the final result may depend on the initial choice of the

centroids, we chose to initialize the method with 10 different initial configurations. The number of clusters can be chosen depending on the number of cities in the list. In our case, we are going to use 3 different clusters. This method provides an initial idea of which cities are more similar or distinct. To quantify the distinction between the cities, we calculate the distance matrix using the Euclidean distance from the `scipy` package. This provides a numerical quantity that is used to distinguish either common or uncommon places. We also create a `Folium` map to display the resulting clustering.

The features to be used on the methods above are the categories of the venues that will be fetched from the Foursquare API. To avoid cities with names in different languages, we use the 10 first items (closest to the given location by latitude and longitude) to get possible alternative names for the cities. On top of that, we will use the property 'distance' of the venues provided by Foursquare. However, one must be very careful when trying to define quantities from the distances of the venues. In this project, we initially defined features such as: the total distance of venues (summing up all the distances of the venues), the maximum distance (obtained from the fetched venue that is located further from the location but still inside the city). Those quantities were intented to give an idea of the size of the city. Nevertheless, since the fecthing process does not acquire all the venues equally, the resulting values did not reflect reality: small cities like Jülich had larger values than Amsterdan or New York. To avoid outlier values of distances, the median was also tried - but without success.

Finally, we have then defined the quantity *density*, which is obtained from the number of venues inside a given radius (in this case, 200km). Even then, the *density* value must carefully considered for two reasons: some cities does not have all the venues registered in the database and the geography of the place may hinder large density of venues even in large cities (as happens, for example, in Rio de Janeiro or New Delhi). For this reason, an optional input was added to use the density or not when clustering the cities (`use_dentity=True` or `False`).

Since the numbers obtained for the features built upon the average of venues categories are small compared to the density one, we need to use feature scaling. This not only avoids overestimating the importante of the density feature, but is also useful for future extensions that may be included.

The clustering, distance matrix and map is returned by a single function that takes as an input a list of addresses, the number of clusters (default value is 3) and the option `use_density`. If the later is chosen to be True, the marker size is calculated with the value of the density. Separate functions for each stage are also provided.

4 Analysis

The analytical part is mostly done in the Jupyter Notebook. Our goal was to create generic functions that may be used for different inputs (list of cities) and may also be extended with more features in the future. This tool may then be used to answer different types of questions regarding city comparisons.

In this section of the notebook, we define:

- Function to get latitude and longitude from an address as an input;
- Function to build a DataFrame with the address and it's geographical coordinates (Latitude and Longitude);
- Function that fetches Venues of different cities using the Foursquare API;
- Function to transform the Categories into columns with binary values;
- We also build a function to use when grouping the values for each city. This function also generates new features that may represent the city (such as the density, in our case);
- A wrapping function that takes a list of cities and builds the final DataFrame to be used with different Machine Learning methods. This function performs the following tasks:
  - Gets the Latitude and Longitude of the cities;
  - Obtain the list of venues, their categories and their distances;
  - Transforms the categories into features (columns) with binary values;
  - Groups the different cities by taking the average for each category, and optionally the number of venues inside a radius ('Density').

To recognize the characteristics of each city after performing some machine learning algorithm, we also build a function to create a Dataframe with the top venues categories in a given city.

4.1 Feature scaling

Even though all our features have values between 0 and 1, the numbers corresponding to the average categories can be small compared to the density (when this is included). Besides, if further extensions are made to the feature list, normalizing the values will avoid overestimating or underestimating the importance of a certain feature. Therefore, we scale the features using the `StandardScaler` from the `sklearn` preprocessing library.

4.2 Distance calculation

To quantify the difference between the cities, we calculate the distance matrix using the *Euclidean distance*.

4.3 Clustering cities

After building a Dataframe with all the features (venues categories and, optionally, density), and apply the K-Means Clustering algorithm to group the cities. The function `build_top_venues_df` is then used to create the final Dataframe to be analysed, after adding the cluster labels and, optionally, the density.

4.4 Visualizing the results

Lastly, to visualize the cities and their cluster, we use a Folium map.

4.5 Generic function

Using all the definitions above, we create a complete function that creates the dataframe, performs, feature scaling, cluster the cities, calculates the distances, and returns the distance matrix, an interactive map with their labels printed out, as well as a dataframe with the top 10 venues characteristics.

5 Results and Discussion

With the generic functions defined in the previous chapter, we can now use them to compare different cities and try to answer a few questions.

5.1 Big and small cities

We start by comparing big cities (Dublin, London, Cologne, Amsterdam, New York and São Paulo), as well as a small one (Jülich, Germany). 259 uniques categories were fetched from the Foursquare API, which were transformed into features. The returned Dataframe is displayed in Figure 1 and lists the most common categories of venues, as well as the cluster calculated by the K-Means algorithm. We can note from this results that a small

	Address	Latitude	Longitude	Cluster Labels	Density	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Dublin, Ireland	53.349764	-6.260273	0	0.590	Mobile Phone Shop	Coffee Shop	Salon / Barbershop	Bus Line	Bus Stop	Women's Store	Café	Fast Food Restaurant	Ice Cream Shop	Convenience Store
1	London, England	51.507322	-0.127647	0	0.510	Office	Outdoor Sculpture	Taxi	Government Building	Italian Restaurant	Coworking Space	Embassy / Consulate	Tech Startup	Coffee Shop	Business Service
2	Cologne, Germany	50.938361	6.959974	2	0.610	Hotel	Office	Bar	Italian Restaurant	German Restaurant	Plaza	Gay Bar	Doctor's Office	Residential Building (Apartment / Condo)	Historic Site
3	Juelich, Germany	50.922093	6.361102	0	0.315	Miscellaneous Shop	Pizza Place	Salon / Barbershop	Optical Shop	Café	Bank	Pharmacy	Clothing Store	Shoe Store	Bakery
4	Amsterdam, Netherlands	52.374540	4.897976	2	0.690	Bar	Coffee Shop	Burger Joint	Office	Adult Boutique	Marijuana Dispensary	Bridge	Café	Hostel	Hotel
5	New York, USA	40.712728	-74.006015	2	0.615	Office	Taxi	Food Truck	Government Building	College Classroom	Building	Lawyer	Bar	Bus Station	General College & University
6	Sao Paulo, Brazil	-23.550651	-46.633382	1	0.725	Office	Pharmacy	College Academic Building	Historic Site	Cosmetics Shop	Building	Courthouse	Church	Music Venue	Outdoor Sculpture

Figure 1: Final Dataframe containing the information for six big cities and a small one. The cluster labels were obtained using the K-Means algorithm. It also contains the 10 most common venues in the cities, which gives an idea for the character of the city.

city (Jülich) may be clustered together with the big cities in spite of other big cities such as São Paulo being in a separate cluster.

The distance matrix, showed in Figure 2, provides a quantification for those differences. Note that the distance from São Paulo to all the other cities tend to be larger - even though other cities are also disparate (as Dublin from Jülich, or New York from Cologne). In this sense, the city that is more similar to Dublin is London, followed closely by Amsterdam and then Cologne.

The clusters can be plotted on a map, which can provide further insights to the results. It is illustrated in Figure 3. It seems that geographical distance have a large influence on the similarity or difference between the cities. For example, Jülich and Cologne, nearby cities, have the closest Euclidian distance. Another correlation can originate from historical reasons: New York and London are relatively similar to each other.

	Dublin, Ireland	London, England	Cologne, Germany	Juelich, Germany	Amsterdam, Netherlands	New York, USA	Sao Paulo, Brazil
Dublin, Ireland	0.000000	24.246767	24.564217	25.478818	24.393808	24.775783	25.935727
London, England	24.246767	0.000000	24.303802	24.877851	24.405267	24.845097	25.162867
Cologne, Germany	24.564217	24.303802	0.000000	22.601027	22.627780	25.166785	25.695264
Juelich, Germany	25.478818	24.877851	22.601027	0.000000	23.770524	24.673967	26.207041
Amsterdam, Netherlands	24.393808	24.405267	22.627780	23.770524	0.000000	22.914864	24.833543
New York, USA	24.775783	24.845097	25.166785	24.673967	22.914864	0.000000	25.365967
Sao Paulo, Brazil	25.935727	25.162867	25.695264	26.207041	24.833543	25.365967	0.000000

Figure 2: Distance matrix for the cities in the list. The values were obtained by calculating the Euclidian distance from the normalized set of features.

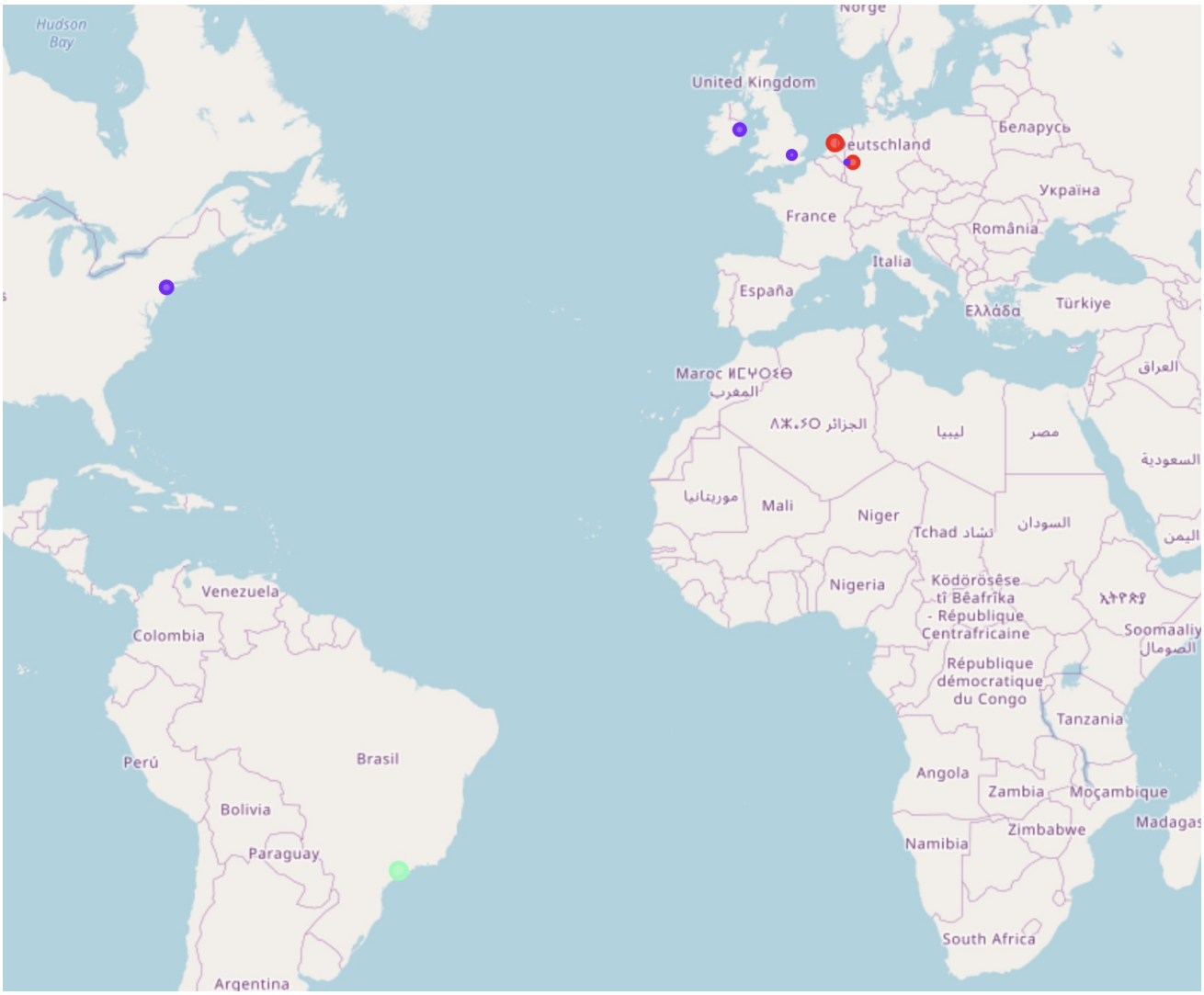

Figure 3: Map displaying the cities in the list. Different colors on the markers represent different clusters. The marker size is obtained from the 'Density' column of Figure 1.

5.2 Large differences

We now try to find the difference between very disparate big cities — a question that a traveller or an explorer may have. We compare the cities of Shanghai, Tokyo, Johannesburg and São Paulo. As a mean of comparison, we also include Cologne and Jülich, the cities with shortest distance from the previous example. In this case, the density will not be used, since Foursquare provide very different number of venues in each of the cities. 240 uniques categories were obtained, which were transformed into features.

We start by looking at the clusters and the most common venues for each city. The Dataframe is shown

in Figure 4. In this case, the city of São Paulo stands in a separate cluster, indicating that it is still the most

	Address	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Shanghai, China	31.232274	121.469175	1	Chinese Restaurant	Coffee Shop	Office	Theater	Art Museum	Japanese Restaurant	Shopping Mall	Tea Room	Convenience Store	Italian Restaurant
1	Tokyo, Japan	35.682839	139.759455	1	Historic Site	Japanese Restaurant	Office	Convenience Store	Italian Restaurant	Coworking Space	Chinese Restaurant	Hotel Bar	Police Station	Lounge
2	Johannesburg, South Africa	-26.205000	28.049722	1	Office	Building	Café	Automotive Shop	Clothing Store	Portuguese Restaurant	Fast Food Restaurant	Bank	Coworking Space	Breakfast Spot
3	Sao Paulo, Brazil	-23.550651	-46.633382	2	Office	Pharmacy	College Academic Building	Historic Site	Building	Cosmetics Shop	Courthouse	Church	Monument / Landmark	Spa
4	Cologne, Germany	50.938361	6.959974	1	Hotel	Office	Bar	Plaza	German Restaurant	Italian Restaurant	Gay Bar	Residential Building (Apartment / Condo)	Road	Brewery
5	Juelich, Germany	50.922093	6.361102	0	Miscellaneous Shop	Pizza Place	Salon / Barbershop	Café	Optical Shop	Pharmacy	Bank	Clothing Store	Doner Restaurant	Bakery

Figure 4: Final Dataframe containing the cluster labels and the 10 most common venues in the cities, which gives an idea for the character of the city.

different from the other ones. This is confirmed by the distance matrix displayed in Figure 5. Note that, even

	Shanghai, China	Tokyo, Japan	Johannesburg, South Africa	Sao Paulo, Brazil	Cologne, Germany	Juelich, Germany
Shanghai, China	0.000000	23.110470	24.129704	24.264237	23.550396	23.752523
Tokyo, Japan	23.110470	0.000000	23.546244	24.225114	23.327369	23.288283
Johannesburg, South Africa	24.129704	23.546244	0.000000	25.396546	24.318102	23.828016
Sao Paulo, Brazil	24.264237	24.225114	25.396546	0.000000	24.630309	25.496425
Cologne, Germany	23.550396	23.327369	24.318102	24.630309	0.000000	22.966630
Juelich, Germany	23.752523	23.288283	23.828016	25.496425	22.966630	0.000000

Figure 5: Distance matrix for the cities in the list. The values were obtained by calculating the Euclidian distance from the normalized set of features.

though the distance between Cologne and Jülich is still the smallest one, K-Means clustering placed them in different clusters.

We can extend now the clustering results between the big cities above by decreasing the list size. By removing Cologne and Jülich, we obtain the Dataframe and distance matrix given in Figures 5 and 6, respectively. We see

	Address	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Shanghai, China	31.232274	121.469175	2	Chinese Restaurant	Coffee Shop	Office	Convenience Store	Noodle House	Italian Restaurant	Shopping Mall	Japanese Restaurant	Theater	Art Museum
1	Tokyo, Japan	35.682839	139.759455	1	Historic Site	Japanese Restaurant	Convenience Store	Office	Police Station	Italian Restaurant	Lounge	Coworking Space	Hotel Bar	Chinese Restaurant
2	Johannesburg, South Africa	-26.205000	28.049722	1	Office	Building	Café	Automotive Shop	Clothing Store	Bank	Fast Food Restaurant	Portuguese Restaurant	Coworking Space	Doctor's Office
3	Sao Paulo, Brazil	-23.550651	-46.633382	0	Office	Pharmacy	College Academic Building	Cosmetics Shop	Building	Historic Site	Church	Courthouse	Music Venue	Road

Figure 6: Dataframe containing the cluster labels and the 10 most common venues in the cities, which gives an idea for the character of the city.

now that São Paulo seems closer to Shanghai and Johannesburg, and Tokyo is the most different one. Although this seems to be an unintuitive change compared to the previous result, this may be caused by the feature scaling that now does not include Cologne and Jülich.

6 Conclusions

In this work, we have compared cities using K-Means clustering and Euclidian distances. This was done using the categories of the venues in a city. We have also defined the number of venues inside a given radius as a measure of the density of a city. This allowed us to answer the question: can a large city be more like a small city that has similar venues than another big city with different venues? Our results showed that it can, and geographical and historical correlations may play a bigger role than the size of the city.

We have also compared cities in different corners of the world, to see how so contrasting cities would be compared by our algorithm. The distances we obtained were, in fact, comparable to the ones for cities that are alike. These results, however, are certainly not exact. Our list of features only took into account the list of venues in a city. A realistic result should also include the cost of living (such as average rental and price of food),

language, density of streets, number of cars, and so on. Not only that, but our results may also be affected by the amount of venues retrieved from locations where Foursquare is not common. Even though the comparisons made by our algorithm can already provide important insights regarding the cities, the extension of the features and improvement of the data will certainly make them more realistic. Another possible extension for this project is to implement other functionalities. The Decision Tree method, for example, can refine the results by displaying where and how one city diverge from the other.

Finally, apart from the numerical results obtained in this work, the functions defined here may also be used as examples and templates for future projects.