Applied Data Science Capstone: Most Similar Neighborhood in Another City

E. M. de Morais

February 17, 2021

1 Introduction

1.1 Background

São Paulo and Rio de Janeiro are the two largest cities in Brazil. Although relatively close (given the continental dimensions of the country), they have very different characteristics. One of the factors that contribute to the existence of these discrepancies is the obvious geographical distinctions that naturally shape social behavior and, indirectly, the urban structural configuration. The beaches of Rio de Janeiro, for example, give residents a natural option of entertainment and the nearest commercial places have characteristics aimed at this purpose. On the other hand, in São Paulo, entertainment options are directly dependent on experiences that can be provided as a business model.

From a social point of view, Rio de Janeiro inherits aristocratic characteristics, for having long been the capital of the empire of Brazil. In contrast, the history behind São Paulo's growth is much more associated with commercial reasons. For this reason, the lifestyle in São Paulo tends to be much more pragmatic than in Rio de Janeiro. This characteristic also influences the business model, as there are many more establishments in São Paulo that provide what people do not have time to do.

1.2 Problem

We have two of the largest cities in the world with very distinct characteristics: São Paulo and Rio de Janeiro. Locally, each of these places acts like professional centers attracting professionals and companies. Despite the differences mentioned, it is expected that several specific neighborhoods of these cities have characteristics in common. One way to look for this similarity would be to consider the types of venues in each neighborhood. In this way, it would be possible to detect whether a neighborhood has characteristics more focused on commerce, leisure, sports, culture or housing. So, what we need here is a procedure capable of relating neighborhoods, not only from the same city, but also from different cities.

	city	neighbourhood
0	Rio de Janeiro	Abolição
66	Rio de Janeiro	Gávea
91	Rio de Janeiro	Lagoa
141	Rio de Janeiro	Rocha
439	São Paulo	Jardim Ana Maria
1585	São Paulo	Vila Espanhola
663	São Paulo	Jardim Galli
190	São Paulo	Alto do Pari

Figure 1: Example of neighbourhoods in São Paulo and Rio de Janeiro.

1.3 Interest

It is very common for people to have to move to another city. The reasons for this are quite diverse and can involve professional opportunities, personal reasons, among others. This process can be difficult for some people, as it involves adapting to a new and possibly different environment. Although two cities may be completely different, they may have neighborhoods with similar characteristics in common. Knowledge of this information can be crucial when choosing a new place to live in a new city. Thus, a person can choose to move to a neighborhood in a new city with similar characteristics to the neighborhood in the city where he lives in and is already acclimated.

Sometimes, a company in one of these cities is interested in a professional that lives in the other city. Certainly, if the professional finds a similar neighborhood, he will be more likely to accept an offer if it is professionally interesting.

2 Data

The name of all neighborhoods of São Paulo and Rio de Janeiro are needed to solve the problem. The data given at the site `www.guiamais.com.br` is used. The respective URLs are:

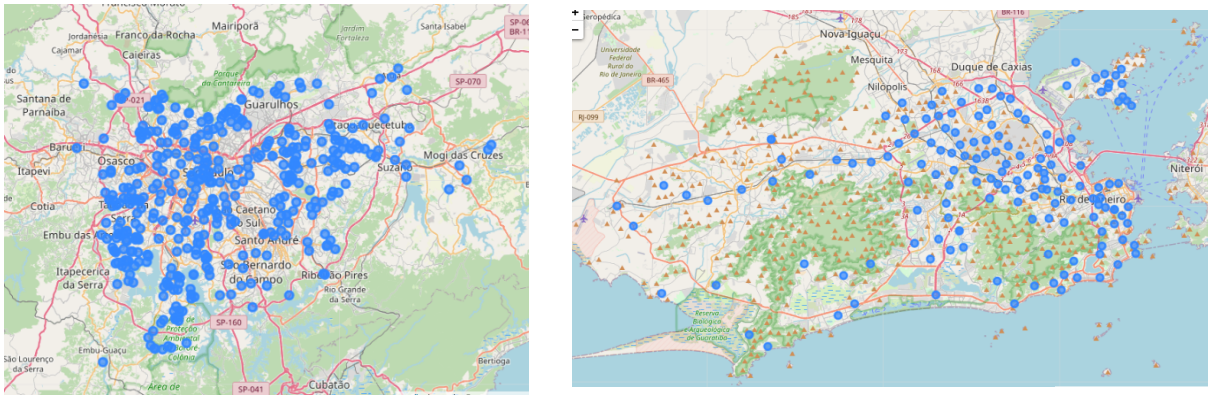
- (São Paulo): https://www.guiamais.com.br/bairros/sao-paulo-sp
- (Rio de Janeiro): https://www.guiamais.com.br/bairros/rio-de-janeiro-rj

We can see an example of data scraped from the pages in Fig. 1.

As the next step, we need to get the latitude and longitude of all neighborhoods. It can be done using the Nominatim search API, from OpenStreetMaps. An example of data with coordinates can be see on Figure 2. Its possible to see the neighborhood positions in the maps of Figure 3.

	index	city	neighbourhood	latitude	longitude
85	85	Rio de Janeiro	Jardim América	-22.810005	-43.323212
123	123	Rio de Janeiro	Penha	-22.838113	-43.276445
38	38	Rio de Janeiro	Coelho Neto	-22.832245	-43.350191
110	110	Rio de Janeiro	Olaria	-22.304596	-42.540627
1053	1053	São Paulo	Jardim São Rafael	-20.421091	-49.991631
877	877	São Paulo	Jardim Nova Tereza	-23.346378	-46.746549
1641	1641	São Paulo	Vila Império	-23.643194	-46.583425
402	402	São Paulo	Ipiranga	-23.589273	-46.606162

Figure 2: Example of neighbourhoods in São Paulo and Rio de Janeiro with latitude and longitude imported from Nominatim.



(a) São Paulo

(b) Rio de Janeiro

Figure 3: Example of venues on neighbourhood with index `nb_index` = 0.

	id	nb_index	name	categories	distance
0	4d5dad65f7d7224b54652081	0	Crepe Lieto	Deli|Bodega	93
1	4ba55e8df964a5200a0039e3	0	Rei do Bacalhau	Portuguese|Restaurant	420
2	523e424f11d2b11ae99aa553	0	Esquina do Espeto	Restaurant	350
3	4c6f2ae134443704c360205f	0	Tramelinha	Bar	508
4	5225259811d2d985551c1ba2	0	Sabor do Árabe	Middle|Eastern|Restaurant	1824
5	4b899309f964a520f34232e3	0	Outback Steakhouse	Steakhouse	1910
6	4f13006ce4b019e992bd19e2	0	Kopenhagen	Chocolate|Shop	1719
7	4bb247ae42959c74a49e202c	0	Saraiva MegaStore	Bookstore	1812

Figure 4: Example of venues on neighborhood with index `nb_index` = 0.

The coordinated data are then used for the foursquare API in order to get the venue data of all neighborhoods. For this, I used the explore query:

```
https://api.foursquare.com/v2/venues/explore?&client_id={}
&client_secret={}&v={}&ll={},{}&radius=5000&limit=1000
```

where the `client_id`, `client_secret` and `v` are credentials information, `ll` is the latitude and longitude of neighbors. I limited the search to up to 1000 venues within a 5 km radius of each neighborhood. We will need the name, categories and distance from neighborhood coordinates of each venue. An example of this data is given in Figure 4. This table is linked with the neighborhood table by the index `nb_index`. The radius of 5km may be to large for some neighbors. However, it is possible to select the best value of this parameter based on the quality of the clustering. This optimization procedure will be covered later.

3 Methodology

3.1 Pre-Processing

In this point, we have two main tables, which is that of Figures 2 and 4. We can merge this datasets by the `nb_index`. In fact, this is what will be done. However, it is convenient to allow the user to choose only the venues within a maximum radius determined by a parameter that we will call `max_distance`. So, to make this process easier, lets define the following function

```
def limit_venue_distance(X,max_distance):
  return X[X['distance']<=max_distance]
```

After this, we have to aggregate the venue table by `nb_index`. To do this, lets consider the function:

```
def agg_venues_by_nb(X):
```

neighbourhood_id	city	neighbourhood	latitude	longitude	categories
486	São Paulo	Jardim Bom Pastor	-23.678629	-46.551279	Bar|Coffee|Shop|Warehouse|Store|Park|Motorcycl...
847	São Paulo	Jardim Morro Verde	-22.388723	-46.932356	Ice|Cream|Shop|Pizza|Place|Hotel|Asian|Restaur...
1014	São Paulo	Jardim Santa Maria	-23.583182	-46.811105	Café|Gym|Fitness|Center|Shopping|Mall|Warehous...

Figure 5: Example of categories in neighborhoods.

```
  concatenate_categories = lambda series:  reduce(
    lambda x, y:  x +'|'+y, series)
  return X.groupby('nb_index').agg({'categories':  concatenate_categories})
```

The last step is the merge:

```
def merge_categories(df, df_venues, max_distance=1000):
  categories = agg_venues_by_nb(limit_venue_distance(
    df_venues,max_distance=max_distance))
  return pd.merge(
    df,
    categories,
    how = 'inner',
    left_on = 'index',
    right_index = True,
  ).rename(columns={'index':'neighbourhood_id'}).set_index(
    'neighbourhood_id')
```

At this point, we have a table with a column containing all the categories in a given neighborhood. These categories can be repeated within a row since the aggregation function concatenated the categories of all the neighborhood venues. An example of this table can be seen in the Fig. 5. The next step is count how many times each category appeared in total. To do this, I defined the following function:

```
def n_most_common_words(categories,n=None,n_min = 1,n_max = 9999999):
  words_list = '|'.join(categories)
  for char in ['/','&']:
    words_list = words_list.replace(char,'')
  result = Counter(words_list.split('|'))
  if n:
    result = dict(result.most_common(n))
  else:
    result = dict(result.most_common())
  return dict((filter(lambda x:  (x[1]>=n_min) and (x[1]<=n_max),
      result.items())))
```

The most common words are Shop appearing 8224 times, Gym with 6710, Place with 5573, Store with 5267 and Bar with 4352. I will consider only the words that appears more than

neighbourhood_id	city	neighbourhood	latitude	longitude	Shop	Gym	Place	Store	Bar	Bakery	...	Farm	Pie	Mountain	Hall	Cocktail
504	São Paulo	Jardim Cambará	-23.422117	-46.555927	13	10	7	6	5	10	...	2	0	0	0	0
433	São Paulo	Jardim Amália	-23.412180	-47.406458	0	3	2	2	0	2	...	0	0	0	0	0
58	Rio de Janeiro	Flamengo	-22.933984	-43.174574	10	4	0	2	9	0	...	2	0	2	1	2
819	São Paulo	Jardim Marilu	-20.811324	-49.502651	4	2	5	1	6	1	...	0	0	0	0	0

Figure 6: Example of count of categories in neighborhoods.

a specified number of times. This number will be a new model parameter, called `n_min`, that I will optimize.

Selecting the most significant words, let's count how many times each of these categories appears in each row. This value will be allocated in a new column with the category name. After this, the dataframe will look like the Fig. 6. After this, I will try two different types of interpretation. The first one is to scale the columns using the Min-Max approach. The second one is to consider every number greater then `n_cats` as 1 and 0 otherwise. The parameter `n_cats` is free and is something we have to find. We need the optimization procedure to determine which is the best. To proceed this pre processing step, I define a function called `categories_processing` that can be accessed in the GitHub Repository related to this project [1].

3.2 Model

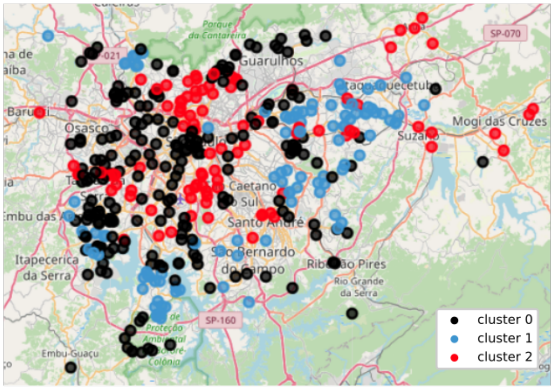
The clustering procedure we choose is the KMeans [2]. We are not going to determine the number of clusters `k` and the `algorithm` now. These parameters will be kept free and optimized together with others in the same step.

3.3 Optimization

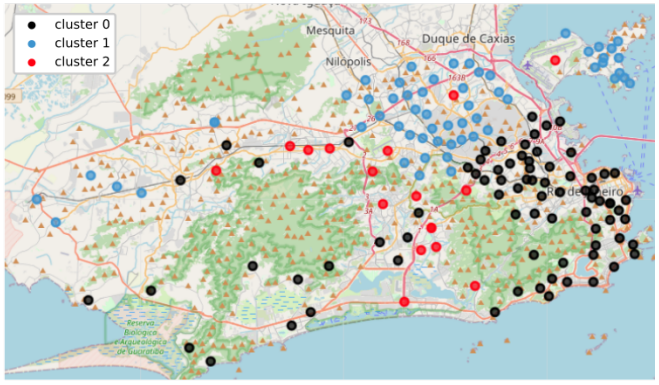
We have to determine the parameters `max_distance`, `n_min`, `n_cats`, `k` and `algorithm` before proceeding to the results. This will be done using a Markov-Chain-Monte-Carlo approach as a stochastic optimization procedure[3]. The metric to be used here will be the Calinski Harabasz score [4]. I will run a sample with 1000 iterations and get the optimal solution.

4 Results and Discussion

As commented before, the approach depends on parameters left free until now. To determine these parameters, I ran a MCMC sample with 1000 iterations and got the following optimized solution:



(a) São Paulo



(b) Rio de Janeiro

Figure 7: Clustering approach in neighborhoods.

Parameter	Best Value
`max_distance`	4892
`n_min`	69
`n_cats`	-
`k`	3
`algorithm`	`full`

This parameters configuration means that we are considering that a venue is part of a neighborhood if it is within a radius of 4892 meters from its coordinates. The categories will be chosen if it appears at least 69 times through the venues. The optimization procedure select that the MinMax scaling approach is better than to consider every number greater then `n_cats` as 1 and 0 otherwise. So, `n_cats` is meaningless in this context. The optimal configuration for KMeans clustering is with 3 clusters, using the algorithm `full`. We can see the distribution of neighborhood with its respective labels in Fig. 7.

The most common words in each cluster can be seen in Fig. 8. We can see that a lot of words are appearing in all clusters. The reason for this is that both cities have a very strong urban profile. So, it is natural that venues that meet basic needs are recurrent. We can see that in cluster 0, we have categories as Beach, Park, Hotel Cultural Center and etc, showing us that this cluster is more associated with places with more leisure options. The cluster 1, we can see categories as Fast_Food_Restaurant, Burguer_Joint, Food_Truck and etc. This means that in these places people tend to look for places for fast food. This may be an indication that these are places aimed at professional or corporate environments.

5 Conclusion

It was possible to cluster the neighborhoods of both cities and determine three types of clusters. Maps showing the geographical distribution of these clusters have been generated, and indicate a relationship between the central regions of these cities, leisure regions and suburban regions.

	0	1	2
category_1	Restaurant	Food	Restaurant
category_2	Ice_Cream_Shop	Restaurant	Burger_Joint
category_3	Park	Fast_Food_Restaurant	Japanese_Restaurant
category_4	Dessert_Shop	Ice_Cream_Shop	Ice_Cream_Shop
category_5	Steakhouse	Japanese_Restaurant	Dessert_Shop
category_6	Italian_Restaurant	Burger_Joint	Pet_Store
category_7	Hotel	Market	Park
category_8	Coffee_Shop	Dessert_Shop	Café
category_9	Pet_Store	Snack_Place	Italian_Restaurant
category_10	Market	Food_Truck	Steakhouse
category_11	Beach	Park	Coffee_Shop
category_12	Café	Coffee_Shop	Market
category_13	Theater	Supermarket	BBQ_Joint
category_14	Supermarket	Seafood_Restaurant	Supermarket
category_15	Museum	BBQ_Joint	Hotel
category_16	Bookstore	Grocery_Store	Middle_Eastern_Restaurant
category_17	Pharmacy	Steakhouse	Grocery_Store
category_18	Plaza	Plaza	Seafood_Restaurant
category_19	Cultural_Center	Beach	Snack_Place
category_20	Grocery_Store	Café	Food

Figure 8: Most common words in each cluster.

References

[1] E. M. de Morais. GitHub Repository. Applied Data Science Capstone. Accessible in https://github.com/emdemor/Coursera_Capstone.

[2] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297,1967).

[3] Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). *An introduction to MCMC for machine learning. Machine learning*, 50(1), 5-43.

[4] Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis, *Communications in Statistics-theory and Methods* 3(1): 1–27.