Closest Venues in the City

Enrique Gámez Flores

Coursera Capstone Report

2020-03-04

Introduction

A simple generic user program to make recommendations about geographical clusters of certain venues in a selected city, or location, is presented.

Imagine that you are planning a trip to a certain city to explore the jazz scene , but you don't want to move so much around and want to explore the possibility to find as much as jazz clubs you can, and a hotel, or hostel, to stay there and all nearby (example 1 in this report). Or imagine that you are willing to open a yoga center, or anything else, but you want to make sure that either there is no yoga center around (example 2 in this report) or there few of them, but in any case all the candidate places are surrounded by certain kinds of stores, so, you want to know how many places, with such conditions, are in the city. In all the aforementioned cases you are willing to get some recommendations about those locations.

All this can be accomplished with the same method, with the use of a particular machine learning algorithm, runned over data provided by the *Foursquare Places* service, in a simpler way.

This solution can be used by travel agencies, to offer personalized experiences, entrepreneurs who like to evaluate new locations, or anyone who is planning a trip.

Data

To generate such recommendations we will make use of the *Foursquare Places* service, for venue discovery, and the final user requirements.

User Requirements

From the final user we require:

- the location to search for, either by its geographical coordinates or by a geolocation string,

- the categories (according to the Foursquare Places Categories already defined), and
- a set of flags about a certain kind of venue must be, or must not be, in the final recommendation

The flags mentioned are to create a simple rule and it is only used in the final part of the full procedure, and it is used for discrimination pourposes. This rule is passed as a list of tuples, and each tuple consists of a venue category name (according to the Foursquare categories) and an integer, and it is precisely this integer the one used to characterize how much a venue type is wanted, and their possible values are:

- ❏ 2, if a venue type must be in the final recommendation,
- ❏ 1, if a venue type may appear,
- ❏ 0, if a venue type could possibly appear also,
- ❏ -1 if a venue must appear in the final recommendation when a venue with value -2 is excluded
- ❏ -2 if a venue needs to be excluded in the final answer but venues with a value of -1 must be in

This information is integrated into the `pandas.core.DataFrame` mentioned in the next section. No data cleaning is required for this type of data.

Venue Discovery and Data Cleaning

We make use of the Foursquare Places service which already provides us a categorized set of venues along with their geographical coordinates, which are the main ingredients to deliver the recommendations.

Once the location of interest is defined, a call to the API endpoint `explore` is made to get all the venues, in the location selected and for the categories wanted. The response is then parsed and a `pandas.core.DataFrame` is created, with only a few pieces of information like the one shown in figure 1.

```
[9]: venues.head()
```

[9]:

	Venue	Latitude	Longitude	Category	FormattedAddress	Wanted
0	sayuri	19.044346	-98.150481	Jazz Club	México	2
1	Festival Co-mu-co	19.067719	-98.279270	Jazz Club	Container City, San Andres De Cholula, Puebla,...	2
2	La Purificadora	19.044183	-98.190885	Hotel	Callejón 10 Norte 802 (Blvd. Héroes del 5 de M...	1
3	Hotel Descansería Business And Pleasure	19.040498	-98.194037	Hotel	3 Oriente 627 (Esq. con Blvd. Héroes del 5 de ...	1
4	Hotel Quinta Real	19.042294	-98.201048	Hotel	7 Poniente 105 (Entre 16 de Septiembre y 3 Sur...	1

Figure 1. Data collected example.

Only the geographical coordinates, *Latitude* and *Longitude* are used to group data, the *Category* and *Wanted* will be used later to produce a recommendation.

Methodology

To deliver recommendations two stages need to be completed:

A. Venue clustering, and
B. Clustering selection based on user's requirements

and two examples were worked out:

1. Puebla, México
2. Geneva, Switzerland

Venue clustering

For venue clustering we use the *KMeans* algorithm from the *Scikit-Learn* package. We run this algorithm with the geographical coordinates as features without any other information, because in this stage we are looking for closest venues only.

An important parameter for the `sklearn.cluster.KMeans` algorithm is the number of clusters to be found in given set. To select such a number a loop is run for different numbers of clusters (from 2 to 18 clusters), all over the same dataset, the *Silhouette Score* and the *Inertia* for each number of clusters is obtained, as shown in the image

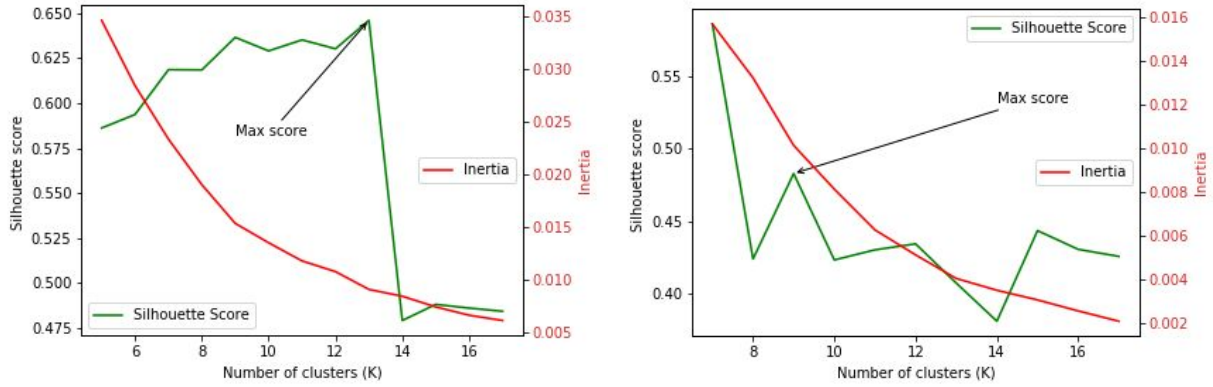

Figure 2. Silhouette score vs. Number of clusters. The image on the left is for example 1, the image on the right for example 2.

Once an adequate number of clusters is chosen, the data is clustered. Figure 3 shows how all the venues for example 1 were clustered.

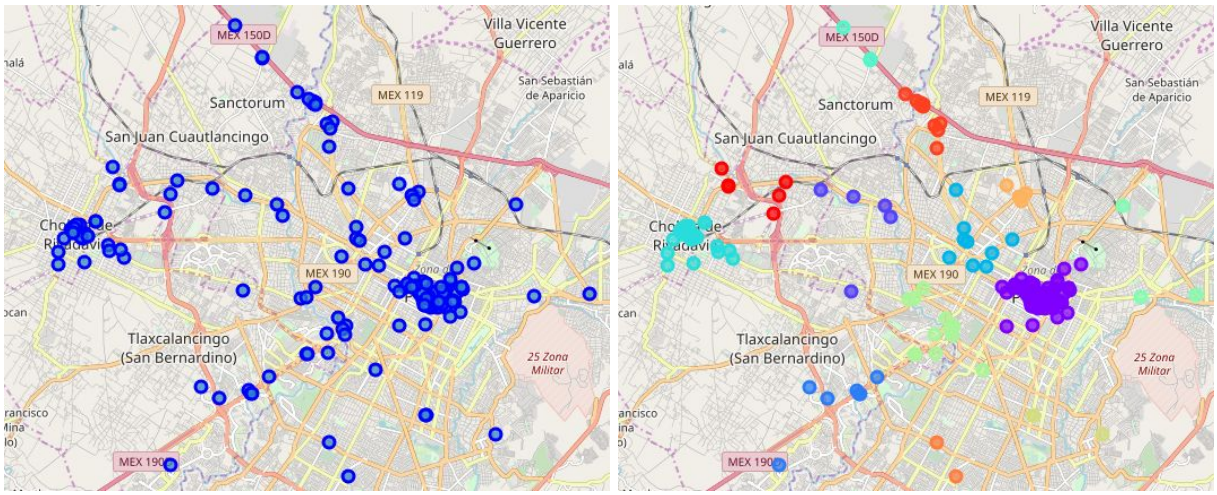

Figure 3. Venues visualization on a map for example 1, before clustering (left) and after clustering (right), different colours for different clusters.

Clustering selection

Once all the venues were clustered, the (clustering) selection can be performed in a straightforward way applying a kind of custom metric function over each cluster (just using the DataFrame method `apply()`), and for both cases presented, a custom score is defined depending on user requirements.

For the examples the score defined ranges from 100 to -100, the greater the score (among all the clusters), the better the recommendation.

It is at this stage where the main decisions are made by adding certain conditions to tune the recommendations and deliver more appropriate answers. A function to calculate the geographical distance between the most wanted venue on each cluster was also included as an example how to include more elaborate functions to characterize each cluster and deliver more accurate recommendations.

Results

Two example applications have shown. In both examples we end up with recommendations about venues geographically closer, and in both cases the KMeans algorithm was chosen with geographical coordinates as features for venue clustering. In one of the cases two recommendations were proposed, as shown in figure 4. For the second example the resulting recommendation was also two places, two locations surrounded by either a spa or an ice cream ship and no yoga center around, as shown in figure 5.

Aside from the recommendations made for the two cases described in which we end up with a result based on the user requirements, we have shown a generic application of how to use the Foursquare Places service and some machine learning algorithms to solve efficiently problems that years ago could be solved only by human inspection, or using a highly sophisticated and particular algorithm.

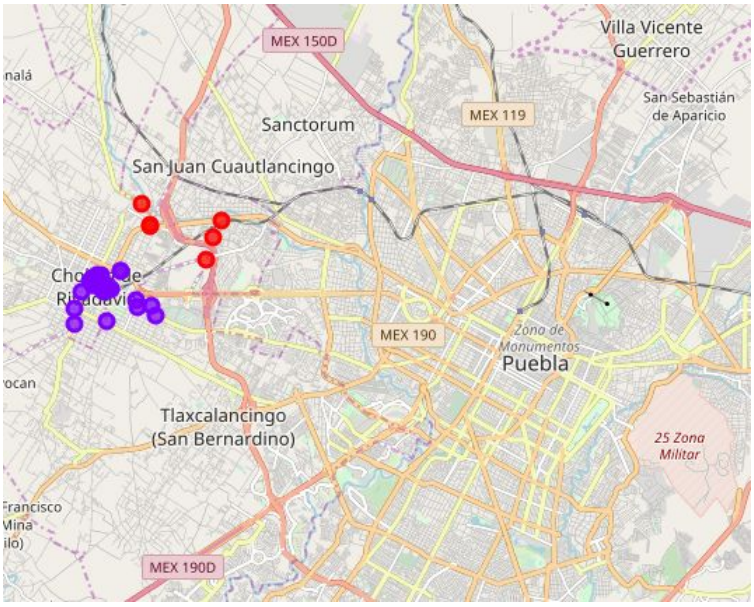

Figure 4. Two clusters (purple and red), both include a jazz club and a few hotels nearby.

Discussion

Querying Foursquare Places, for all the venues we seek, around a specific location can be done in a single call, but finding certain relations can be a little more complicated, relations like spa near a business center, or finding all the possible locations of a yoga studio and an ice cream shop in the city.

We have shown a method to perform such searching tasks with the use of the Python programming language, the Pandas package to manipulate and organize the information, and machine learning algorithms for heavy computational manipulations (and, at the same time, reducing possible errors), in particular the KMeans algorithm. All this can be imagined like a giant infrastructure with some specific knobs to tune calculations and deliver the desired answer, the knobs we need to move are the pieces that make sense the data with the final answer.

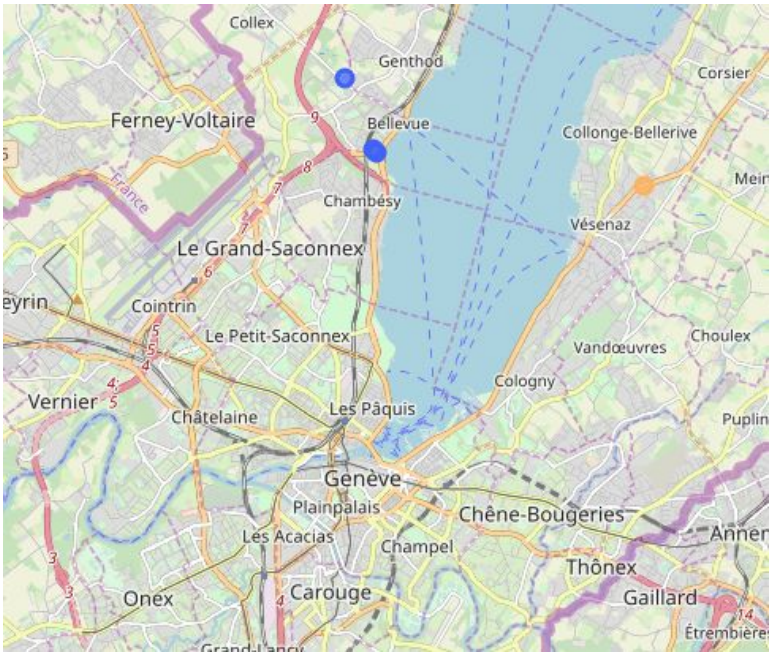

Figure 5. Two clusters were found (blue and yellow) with an Ice Cream Store and no Yoga Club around.

Many data sources were not included, like information about the rates could enrich the recommendations, and can be simply added as a feature into the KMeans algorithm. Also some other features could be included, like real user ratings, social media comments, etc.

About the decision to not include the *Inertia (closest distance)* as a parameter to select a more proper number of clusters, the decision was to group as many venues as we can.

There is another important missing feature, an analytical validation procedure, i.e. no all example recommendations were verified,

Conclusion

The use of machine learning algorithms, to solve problems that years ago could only be solved by human inspection or long, long lines of code, along with robust packages to manipulate the data collected, like the `pandas` library in this case, integrated on an environment written in the same programming language, let us focus only in the essentials of the problem and in how to deliver more robust answers, answers that can include a lot more data sources which adds more features to it. All this could allow us to make more accurate decisions based on more facts.

Applications like the one described it's an example of this, a single (robust) data source, with proper analysis methods put in to the table new insights, all this augment the probability to make more accurate decisions.