An approach for cities clustering

Phuc Tran Duy

April 15, 2019

1. Introduction

1.1. Background

One of the most concern problems of international companies when entering new market is understanding market's culture. However, they can apply their success case studies of similar market in order to reduce analytic and research effort for new market. To achieve this, they must have a method to measure how similar one city with others. This paper propose an approach to represent city by a vector base on nearby venues and then calculate similarity between them.

1.2. Problem

Using venue data of cities in a specific area which might include category tags for each venue to transform to a vector which represents a city. This project aim to tell how similarity between two cities, then apply to clustering a larger set of cities.

2. Data acquisition and cleaning

2.1. Data sources

In order to get geography location of each city, I use Google Map API. Fortunately, there are Python client library which easily integrate with code in notebook.

Base on geography location of each city, I can get nearby venues by using Foursquare API and Google Map API. All we need is venue category field in response from those API.

Example response from Foursquare API (we care about **categories** field):

```
{
    "meta": {…
    },
    "response": {
      "venues": [
        {
          "id": "5642aef9498e51025cf4a7a5",
          "name": "Mr. Purple",
          "location": {…
          },
          "categories": [
            {
              "id": "4bf58dd8d48988d1d5941735",
              "name": "Hotel Bar",
              "pluralName": "Hotel Bars",
              "shortName": "Hotel Bar",
              "icon": {…
              },
              "primary": true
            }
          ],
          "venuePage": {…
          }
        }
      ]
    }
  }
```


Figure 1: Foursquare API response

Example response from Google Map API (we care about **types** field):

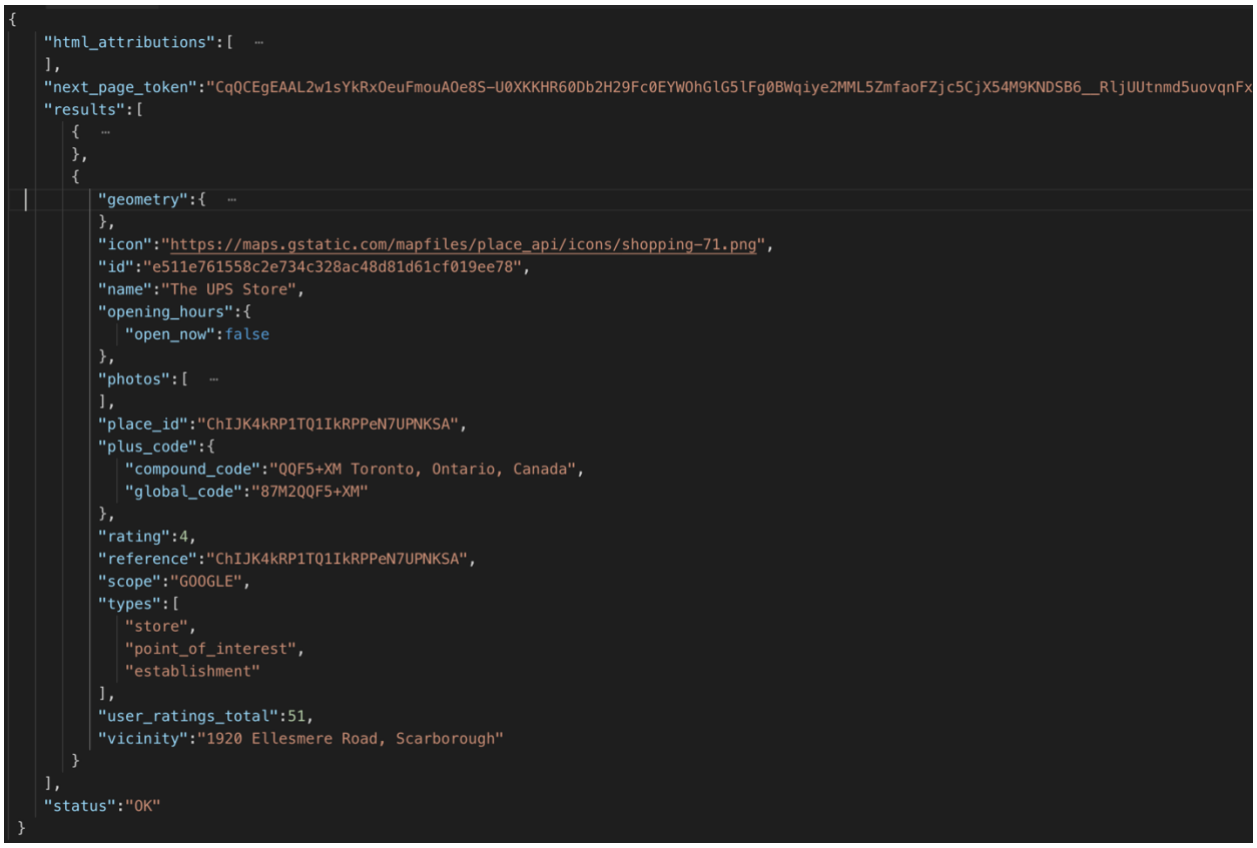

Figure 2: Google map API response

Since venue classify convention is different between two API, we need a normalization stage.

To reduce scope for this report, I just select cities in America. Data of cities in USA was downloaded from https://simplemaps.com/data/us-cities

2.2. Data cleaning

I decided to choose top 3 cites which have highest population in each state. There are a lot of records which miss population field from **us-cities** datasets. So the first thing to do is remove all missing population field records.

Since cities is indicated by their name, I will check if there any duplicate city name records and remove it.

Data from Foursquare API and Google map API are already well-formatted, so I just check if there any missing value in API responses.

2.3. Feature selection

Using cities data from **us-cities**,
For each city, I used Google API to get longitude and latitude. Then base on that geo location, I will get around 100 venues in radius 50000 meter using Foursquare API.

As I mentioned in section 2.1, I just need **venue's categories** field. So for each venue's response, I extracted categories part and assign city name which that venue belong to. For example:

Category	City
Coffee Shop	Dallas
Bar	San Diego
Park	Los Angeles

This dataframe will be transformed to our destination vectors.

3. Methodology

3.1. Explore data analysis

There are **420** unique categories were found in our dataset.

From venue categories dataframe, count appearance of each categories and choose top 10 most popular venues.

Category	Count
Coffee Shop	677
American Restaurant	655
Pizza Place	601
Park	509
Brewery	498
Mexican Restaurant	420
Bar	358
Grocery Store	355
Italian Restaurant	332
Ice Cream Shop	317

3.2. Clustering Model

3.2.1. Feature vector

Transform each venue to one-hot encoding vector.

Coffee Shop	Restaurant	Park	…	Grocery Store
1	0	0	…	0
0	1	0	…	0
0	0	0	…	1

Group these vectors by city name, then calculate average of appearance of each venue.

City	Coffee Shop	Restaurant	Park	…	Bar
Miami	0.2	0.1	0.05	…	0.1
Dallas	0.15	0.08	0.03	…	0.07

This appearance frequency vectors will be our feature vectors.

3.2.2. Cosine similarity

Consider following picture:

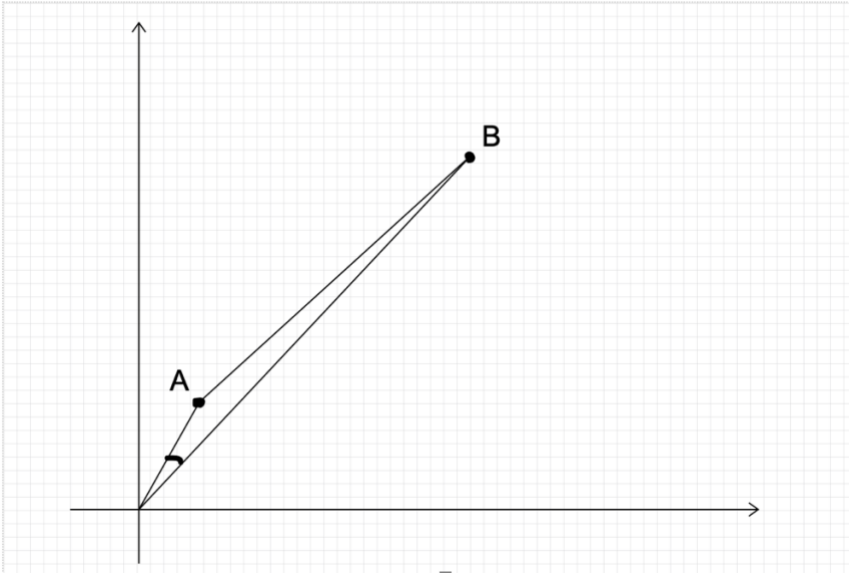

Figure 3: Cosine similarity vs Euclidean distance

You can see that the Euclidean distance between two point A and B can be long. However, the angle which created between them are small. In our case, suppose A and B present cities, we consider two cities a similar if they have similar venues distribution. Euclidean distance doesn't work well with our frequency vectors, but cosine similarity can solve this issue because it only depends on direction of vector, not the magnitude. Therefore, I choose cosine similarity for clustering model.

Cosine similarity formular:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 4: Cosine similarity formula (https://en.wikipedia.org/wiki/Cosine_similarity)

3.2.3. Spectral clustering

Clustering is a widely used unsupervised learning method. There are 2 broad approaches for clustering:

1. Compactness — Points that lie close to each other fall in the same cluster and are compact around the cluster center. The closeness can be measured by the distance between the observations. E.g.: **K-Means Clustering**

2. Connectivity — Points that are connected or immediately next to each other are put in the same cluster. Even if the distance between 2 points is less, if they are not connected, they are not clustered together. **Spectral clustering** is a technique that follows this approach.

Since we use cosine similarity, **Spectral clustering** is better choice than **K-Means Clustering**. So how does Spectral Clustering work?

In spectral clustering, the data points are treated as nodes of a graph. Thus, clustering is treated as a graph partitioning problem. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. An important point to note is that no assumption is made about the shape/form of the clusters.

These are steps for Spectral clustering:

1. Compute a similarity graph
2. Project the data onto a low-dimensional space
3. Create clusters

Step 1:

Construct similar matrix for our city, for example:

	Dallas	San Diego	Phoenix
Dallas	1	c_s(Dallas, San Diego)	c_s(Dallas, Phoenix)
San Diego	c_s(San Diego, Dallas)	1	c_s(San Diego, Phoenix)
Phoenix	c_s(Phoenix, Dallas)	c_s(Phoenix, San Diego)	1

c_s(x, y): cosine similarity function

Step 2 & Step 3:

Using Spectral Clustering in sklearn library, take similar matrix from Step 1 to fit model with 6 cluster.

For the background of Spectral Clustering please find the paper in the reference section.

4. Result

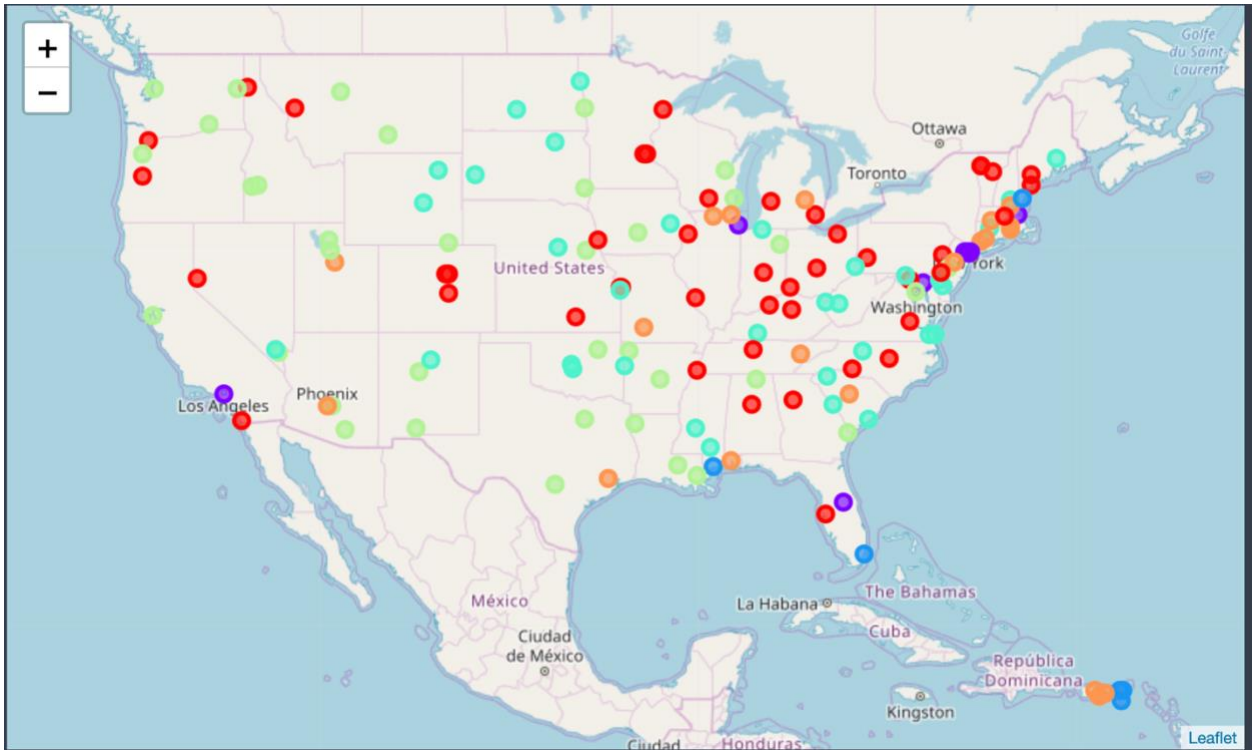

Figure 5: Clustered cities on USA map

You can see there are 6 color represent for 6 cluster. Dots have the same color mean they belong same cluster and likely have strong similarity.

5. Discussion

You can base on result map to choose similarity cities. Suppose you have a success Seafood Restaurant in Miami, you can consider open another one in blue dots cities.

6. Conclusion

In this study, I proposed an approach for clustering cities in USA base on their venues. By present frequency of venues of each city to a vector, I can calculate similarity between them. I also point out why cosine similarity is better suite than Euclidean distance for this problem and apply spectral clustering to fit with cosine similarity. The result can be helpful in supporting decision when entering new potential markets, or find a place similar to your home town.

7. Reference

- https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7
- http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf