# **THE BEST PLACE FOR TWO**

Leonid A. Trusov

March 27, 2019

# **Introduction/Business Problem**

There was described a problem in the beginning of the course: a person needs to relocate and wants to find the same district at the new place.

But it is common when two or more persons participate in the process (a family) and everybody has his/her favorite place. E.g. he is from Toronto, she is from New York, and they want to live together in Paris.

It is not so easy task because it is not just to find the place close by features to the average place of two.

The goal is to provide a tool to choose the place that is the best for all the persons involved knowing their favorite places.

Who can be interested in such a service:

* alone person who wants to relocate
* a man and a woman (or whoever) who want to live together
* friends (e.g. students) looking for an apartment in the city to live together

So the service will help people to see the start points, areas from which they should start to look for what they want.

# **Data**

We will use Foursquare data to retrieve place information, mainly results of explore function because it's free of use.

1. Features of initial place 1
2. Features of initial place 2
3. Features of target places 3 distributed by grid

We will search the best place in the grid (not neighborhoods).

Main information we will extract for every point: quantity of different sections and categories. So there will be a vector-like description of the points.

Plus we will try to use different functions to merge interests of the persons.

1. Min
2. Average
3. Geometric mean

Having calculated all this data, we will get a color map with recommendations.

# **Methodology**

We use some ideas to solve the problem of looking for the best place for 2 or more people knowing their favorite places.

1. The first one is that some valuable features of the places are the venues categories around them that can be extracted using Foursquare API.
2. The second one is that we can adequately normalize raw data using statistics of categories of the whole city.
3. The third one is that similarity can be estimated with dot product of unit vectors.

**Step 1. Defining initial data**

Our initial data is information about favorite places (usually two) and the search place (e.g. the new city). We actually need geo coordinates of the points.

So we can define geo coordinates or names that can be transformed to coordinates with Foursquare API.

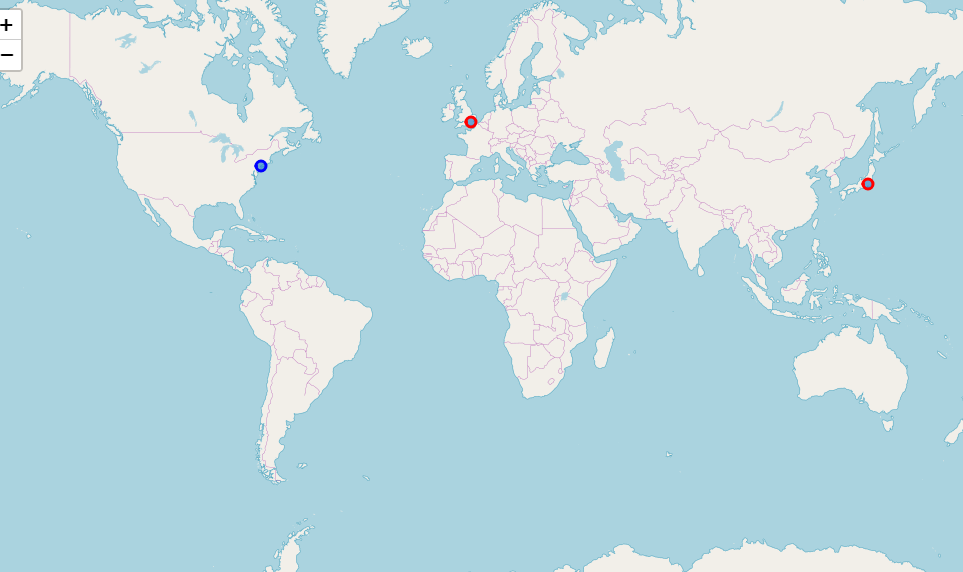
The hint here is that ‘Observe’ function can use not only coordinates but the name, search the appropriate geographical place, and return its coordinates (among with venues that are actually not needed here).

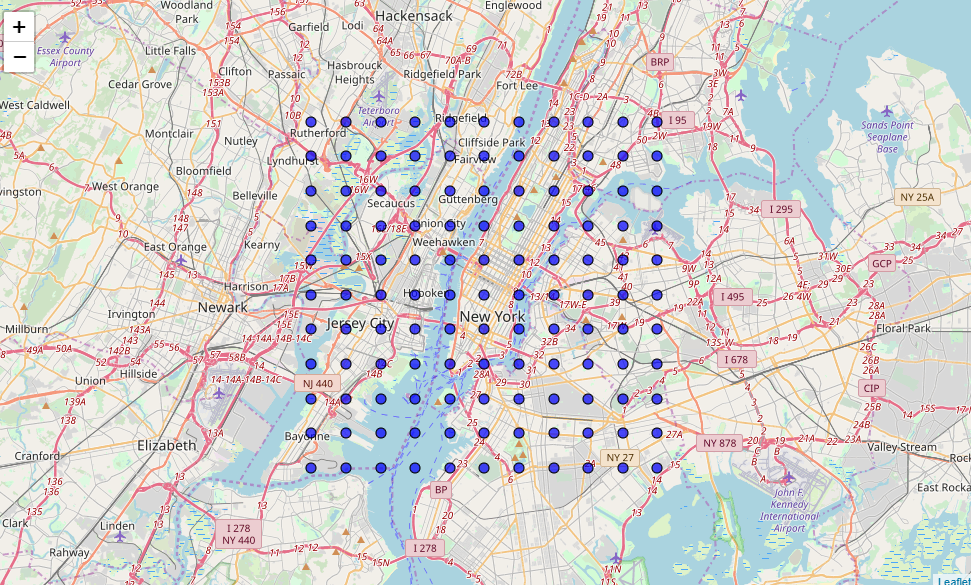
After that we creating the grid of point around the search place with equal step (distance).

We use a geometrical shape of a sphere to calculate steps in units of degrees of latitude and longitude.

Then we show two maps (using folium):

* the map of the favorite places and the search place
* the map of the search grid





**Step 2. Оbtaining information**

The only source of information is Foursquare API.

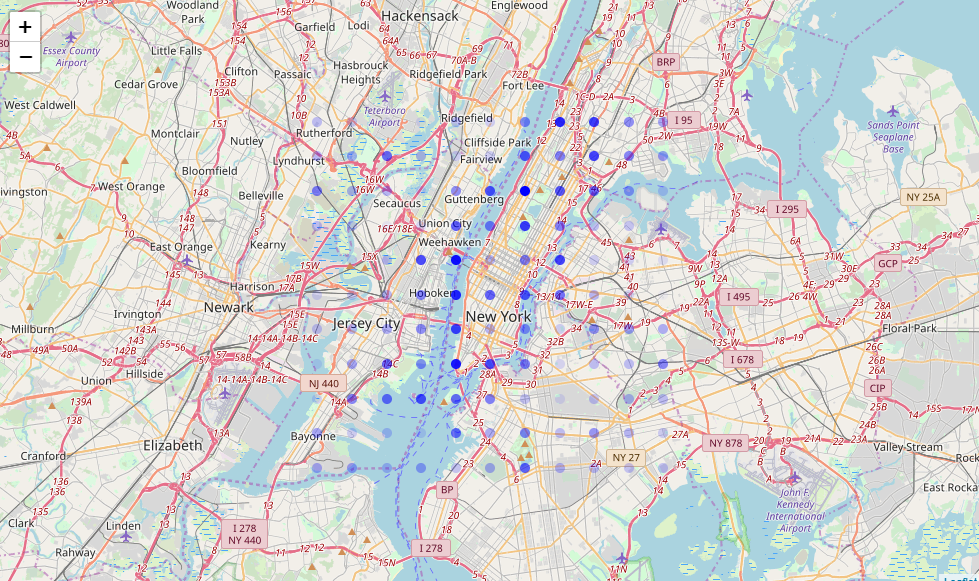
It is very restricted in the free version that is why we use very limited number of search points with big step.

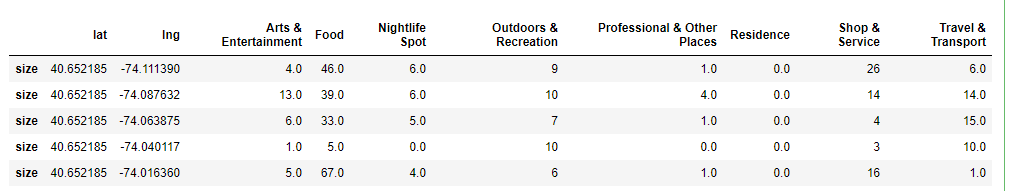
The observe function returns venues and their categories but there are so many categories that only few of them appears for every point. BUt the same function gives a hint that there are hierarchy of categories. So in this paper we use root categories only that we obtain with the function ‘categories’.

We store data in pandas dataframes.

Here some figures:

* 'Outdoors & Recreation' category distribution across New York
* dataframe head

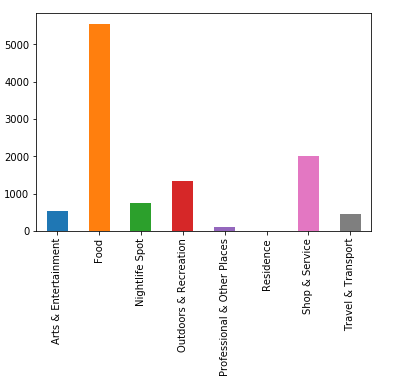




**Step 3. Cleaning and preparation of data**

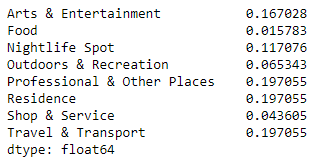
We have noticed that 'College & University' and 'Event' categories are always 0 that is why we drop them from dataframes.

The next observation is that the categories are absolutely unbalanced in number.



So we summarized all the categories and created the vector of weights.

Values for rare categories were so huge that we set them to the next biggest value in the vector.



**Step 4. Calculation**

After that we switch dataframe framework pandas to mathematical one numpy.

We do some additional mathematical preparations:

* multiplying weights
* adding extra feature mean (that should carry information about place popularity vs venue pooraty)
* creating unit vectors (final normalization)

After that we estimate places similarity by calculating dot product.

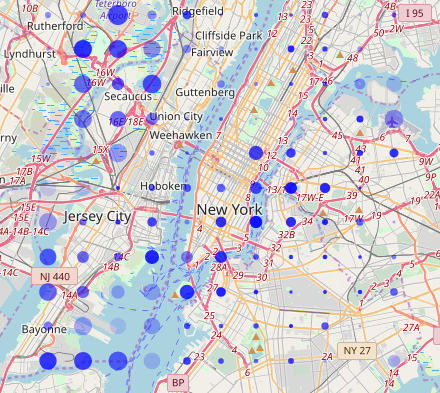
As a result we have two (or more) numbers of similarity (‘goodness’) for every point in the city (through the grid).

Then we use different functions to merge these results for different people: min and mean.

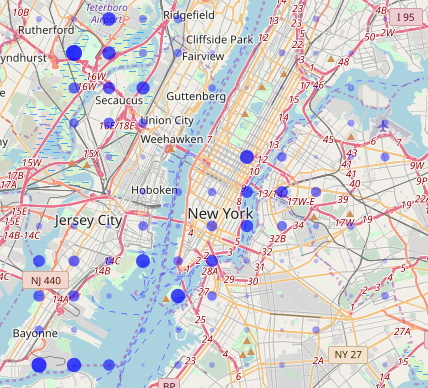
# **Results**

The results for two persons (two favorite places) can be shown with two parameters: dots size and transparency.

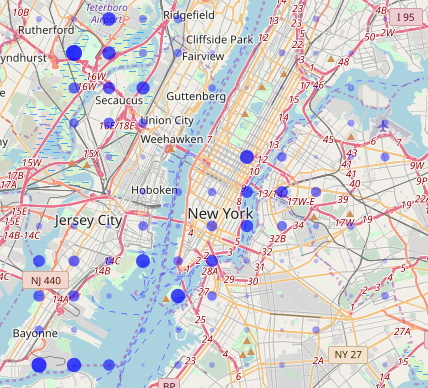
For example, one likes Hyde Park in London, another likes Tokio. What about New York?

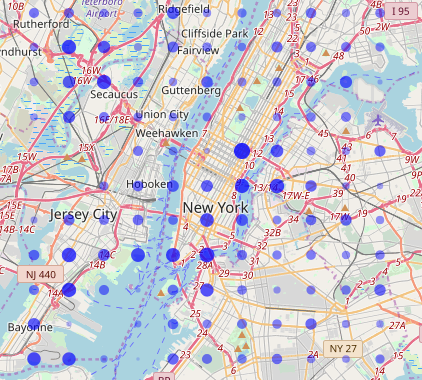


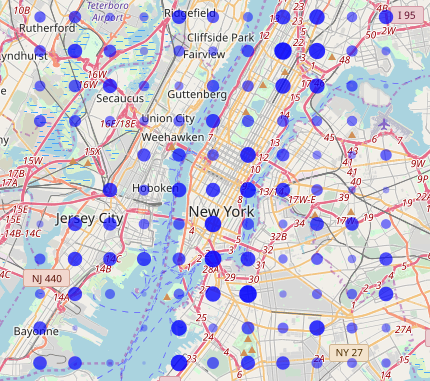
The same figure with using min function:



Then let’s see some maps for different initial favorite places.

Hyde Park (London) vs center Tokyo 

Hyde Park (London) vs center Moscow 

Center Moscow vs center Tokyo 

# **Discussion**

While getting different pairs of initial favorite places and changing search cities, there have been observed some interesting things.

If initial favorite places are very different, the final picture is rare and sparse. For example, Hyde Park in London and some business district in a capital. In this case our map really help to find something that can be liked by both people.

Business districts in favorite places lead to business districts in search cities, parks leads to suburban places and other parks. This obvious result proves our model is consistent.

# **Conclusion**

This paper shows that some geographical data, e.g. Foursquare API, can be used to help people to find places that are similar to the places these people like.

Additional data what precisely people like about the places can make the model better.