The battle of neighborhoods Project

A Travel Advisor

1. Background and business problem definition:

Traveling has become a routine part of our daily lives, be it visits to different neighborhoods, traveling for a vacation or permanently moving to a new location. We choose our destination typically based on our previous experience, recommendation or based on how well the place relates to our personal preferences. Nowadays, the availability of the data that contain information about neighborhoods makes it possible to design a travel advisor that could assist in choosing the right place. In this paper, we define a simple travel assistant that provides a user with a list of most suitable neighborhoods given user's preferences. A practical solution might be deployment of this model into an application, where a user would choose one or multiple categories representing their preferences and based on their choice a list of most suitable neighborhoods will be provided.

2. Data description:

For the purposes of a simple proof of concept, we have decided to continue with the dataset of Toronto's neighborhoods. The postal codes of Toronto are available on the Wikipedia page. The geospatial information of latitude and longitude for these neighborhoods is provided within the project repository. Based on this data, we use Foursquare API to obtain venues located in the neighborhood. For this analysis, we are only interested in a category of the venue, however further expansions might also consider customer reviews. Thus, we associate each postal code with the array of venues' categories which we conveniently lowercase (Figure 1: Data example).

```
Postal Code
M1B                 [fast food restaurant, home service]
M1C    [golf course, construction & landscaping, home...
M1E    [bank, electronics store, spa, restaurant, mex...
M1G    [coffee shop, coffee shop, korean bbq restaura...
M1H    [caribbean restaurant, hakka restaurant, thai ...
Name: Venue Category, dtype: object
```

Figure 1: Data example

3. Feature extraction:

We cannot use a list of categories as features therefore we will continue in data transformation to obtain a more suitable format. Our goal is to create a sparse matrix where each category will be represented as one feature, e.g. one column. We could use bag of words approach but instead we decided to use a method called tf-idf (term frequency – inverse document frequency). This

method is frequently used in text analysis but will serve our purposes well. The score is a product of two computations – term frequency and inverse document frequency.

Term frequency of category c is a simple raw count of the category in each neighborhood. Inverse document frequency of category c can be calculated in three steps. First, we calculate the proportion of neighborhoods that have a venue of category c. Secondly, we take the inverse of this count and lastly, we take its logarithm. In mathematical notation

$$idf_c = \log \frac{N}{|\{d \in D : c \in d\}|}$$

Tf-idf score helps to determine which categories are rare and which are quite frequent. If we were to use only simple bag of words approach, we will favor frequently occurring categories. However, it is believed that rare categories might be more important in determining the correct neighborhood for person's preferences.

Exploring the dataset further, we can notice that the most frequent categories present in almost half of the neighborhoods are coffee shops, pizza place, park.

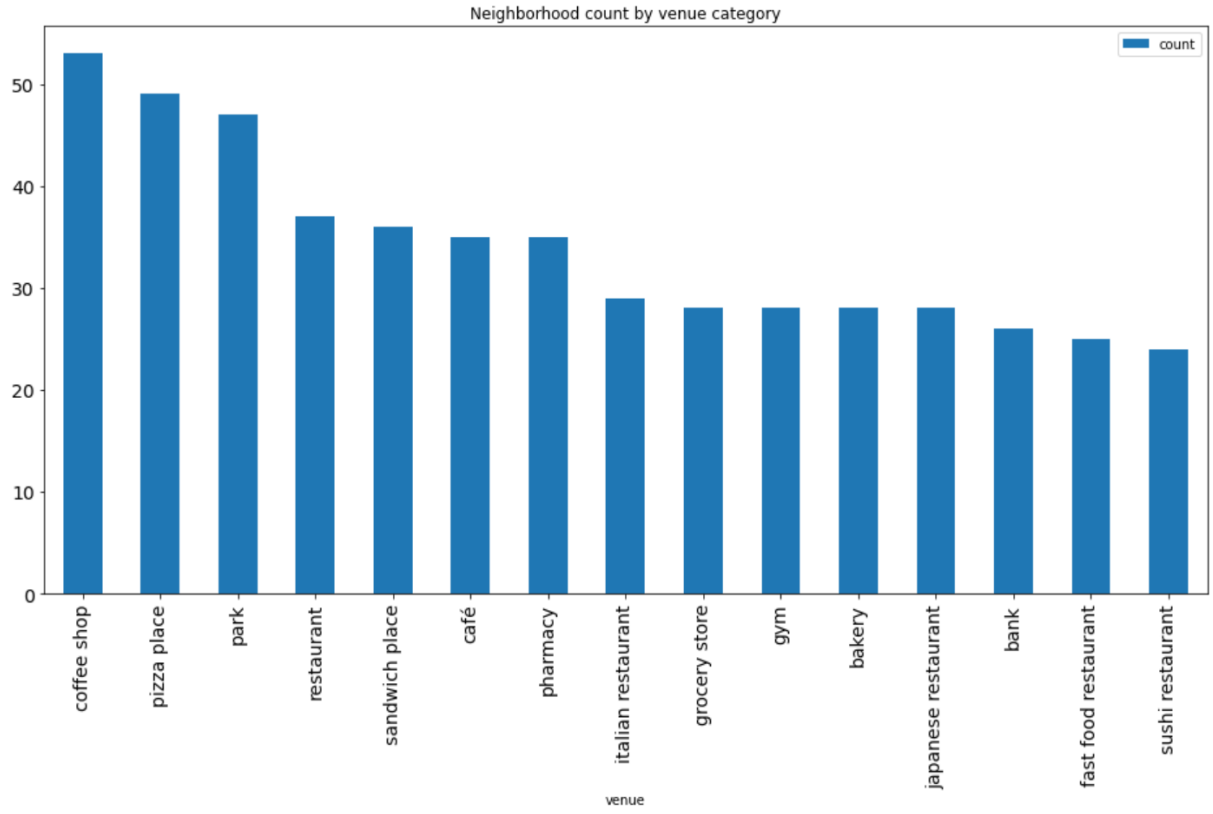

Figure 2: Neighborhood count by venue category

4. Model building:

Now we have everything ready for our model. Latent semantic indexing (LSI) was chosen. The outcome of this method is a similarity matrix that can help us determine how close a new query

is to the corpus, e.g. the neighborhoods. Moreover, it also helps us project our data in a smaller dimensional space. Although the outcome features are hard to interpret, they are more efficient for further use.

The LSI method is based on data transformation leveraging linear algebra. That said, any positively defined matrix can be decomposed into three matrices

$$X = USV^T$$

S is a diagonal matrix of singular values; U is the matrix of left eigenvectors and V is the matrix of right eigenvectors. The singular values are determining the amount of information and typically are sorted from largest to smallest. The small values hold very little information and thus are redundant.

A variable part of this method is to choose the correct number of components, e.g. latent features. We used a scree plot to determine this. The scree plot has a sharp decrease and with as many as 20 components already captures over 77% of variance. Still, we will go a little bit further and choose 50 components to capture 94.6% of explained variance.

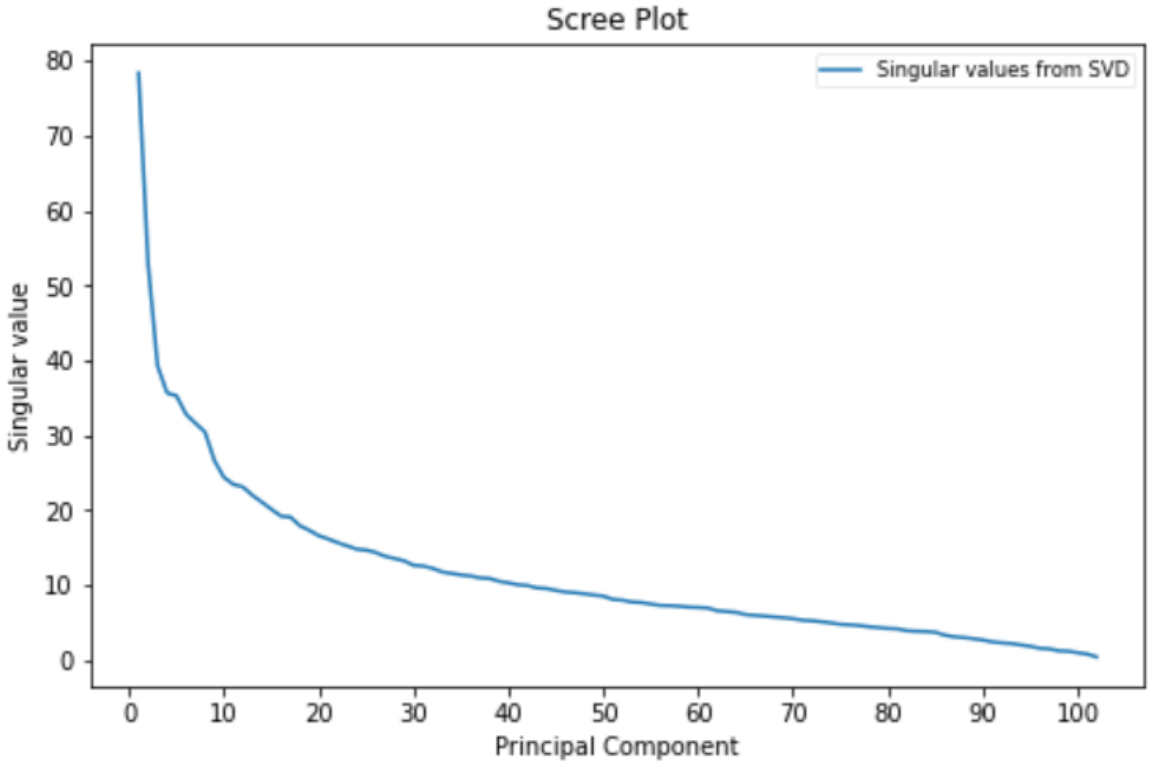

Figure 3: Scree plot

We can use the LSI method to also analyze how similar our neighborhoods are. For that, we will use a TSNE method to translate our matrix into a two dimensional space for better visualization.

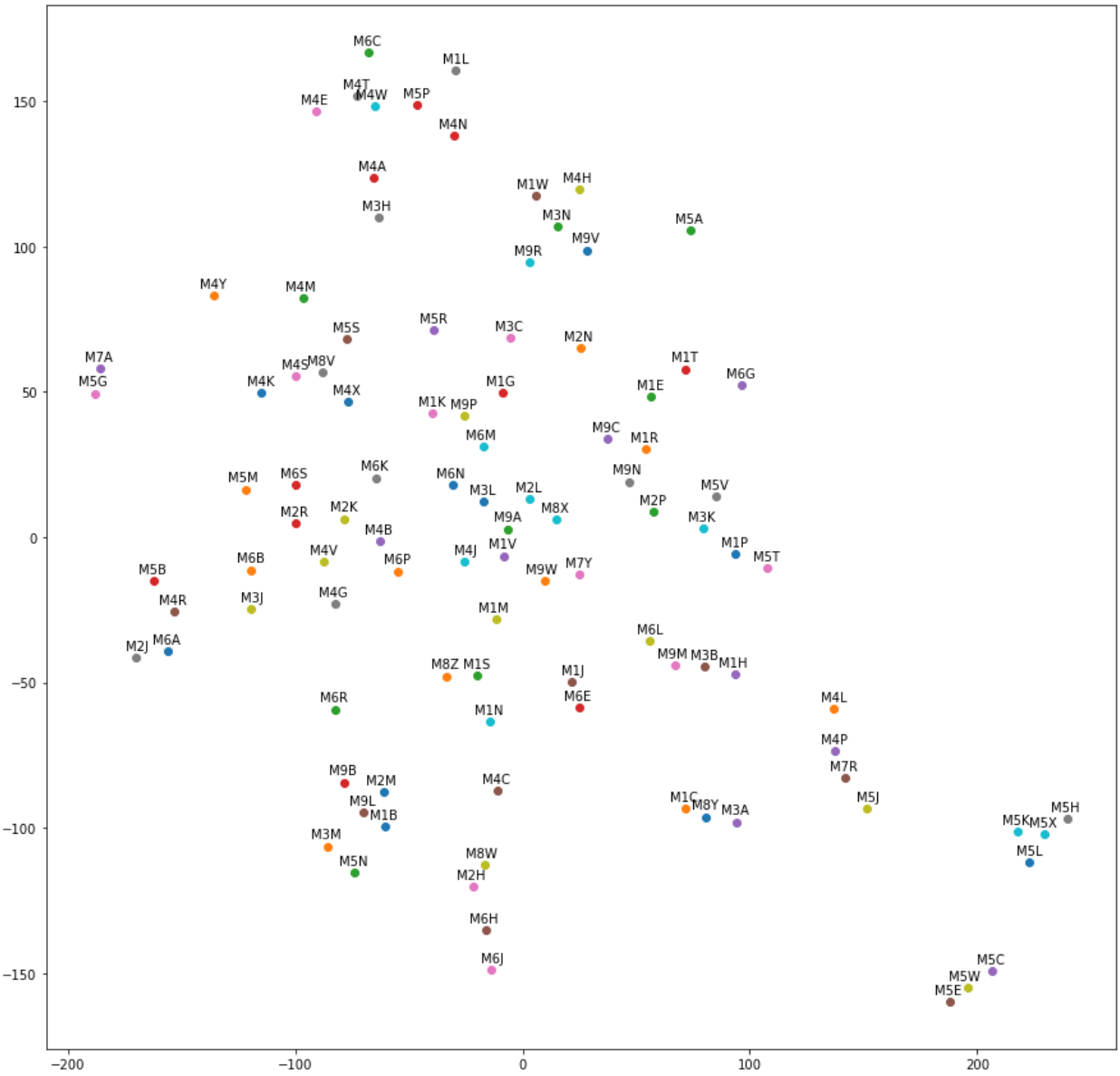

Figure 4: Neighborhoods

Last step in our analysis is to inspect how our model interacts with a user query. For our simplified advisor, a search query will be a list of categories that represent user's preference. First, we transform the query into a vector with tf-idf features. Then, we use a cosine similarity to calculate a distance vector between our query vector and our LSI matrix. The result is a vector of distance measures where each entry represents a similarity to a neighborhood. We can sort the results and retrieve the first 5-10 records.

5. Results and discussion

The first query that we test is with keywords *bakery, japanese restaurant, park, bar*. We inspect the first 5 results, and we compare them to a simple keyword search. The advantage of our method is that it provides a ranked order of the best matches. For example, both methods found neighborhood with postal code M6J, however the LSI method ranked it as the best whereas

simple search retrieved it according to its position in the original dataset. The M6J neighborhood seems to be a good match because it has in total 14 bars. Bars are located only in 22 neighborhoods, so it is a rather high value category. Interestingly, M6J neighborhood also has venues like wine bars, cocktail bars, breweries. Simply, it also has values of similar meaning to the user.

Another difference is neighborhood M5X which was retrieved by LSI method but was not retrieved by the simple keyword search. This neighborhood does not have park but has 4 Japanese restaurants. Japanese restaurant is more difficult to find because it is only located in 28 neighborhoods whereas parks are in 46 neighborhoods. Our search favors the neighborhood with more rare categories. This finding leads us to believe that our search will be stable also in case the user's preferences will not be satisfied by any neighborhood. In other words, there can be 0 neighborhoods that match the user's search query, but our model will still retrieve meaningful results.

We test this hypothesis with our second query where we search for *skating rink, dog run* and *park*. Simple search does not retrieve any results because there is no neighborhood with all three categories. The best neighborhood by LSI model is neighborhood M4C because it has two skating rinks. Only 6 neighborhoods have skating rinks, so it is a rather rare place. Also, the best neighborhood has similar venues like curling ice. Similarly, other retrieved neighborhoods have venues that are sport related and thus fit better the general interest of a person.

6. Conclusion:

Our goal was to build a travel assistant for wide population. In this paper we were able to establish a proof of concept by using tf-idf category coding and training LSI model on neighborhoods in Toronto, Canada. Foursquare API helped us find venues in each neighborhood. We have determined that the LSI model is quite stable and gives ranked results that favor rare categories and neighborhoods with similar venues that are not even specified in the user's query.

The model can be expanded and improved to also take into consideration user's rankings of venues or user's comments about the venues. The LSI method was developed to support text analysis so it will be well suited to integrate unstructured text.

Lastly, our method is efficient because it needs to store only the similarity matrix which is already reduced in dimension. Therefore, it will be well suited to serve online application where a user could search the neighborhoods in real time.