Skip The Effort - Restaurant Recommendation System

Team Member: Nitish Pradhan, Hassan Anwar, Cole Pawliw, Zheng Chen, Mustafa Muhammad

Academic Advisor: Mostafa Muhammad

Sponsoring Company: Skip the Dishes

SCHULICH School of Engineering

Motivation

More people prefer ordering food online and getting it delivered due to health risks associated with visiting restaurants during the pandemic. This project aims to increase ordering frequency and SERP for Skip by providing personalized restaurant recommendations to customers for maximum satisfaction. Current recommendations are based on limited parameters such as location, but our project will enhance the personalization component by using customer data to recommend restaurants and increase economic revenue.

Solution

Our machine looks at two primary data sources:

What Skip gave us:

10 million customer order history and the items offered in restaurants in 2200 uniquely named restaurants meaning only one branch of mcdonalds in Calgary.

How we transformed this:

We create two primary vectors Restaurant and Customer Profiles:

Customer Profile:

We utilized our proprietary filter to generate a personalized unit vector for every individual customer based on their order history. This unique vector comprises a set of values that reflect the customer's preference toward different cuisines. To determine these values, we employed our "Proprietary_filter" algorithm to map each item in the customer's order history to specific cuisines. The algorithm assigns a weight to each cuisine based on the frequency and recency of the customer's orders. Recent orders carry greater weight than older ones.

	customer_id	African	Alcohol	Bakery	\
0	0000000d-9a20-4580-85aa-f4ca9062388c	0.0	0.000000	0.000000	
1	0000267e-c83e-4a48-9776-8163eab97b6a	0.0	0.000000	0.010832	
2	00007e2c-cd9c-43d5-9148-60457215dfd8	0.0	0.000000	0.085024	
3	00008ec4-df06-4b68-a809-b043a65faf7e	0.0	0.000000	0.000000	
4	0000a630-1c0a-45db-8516-d07bda7f22dd	0.0	0.000000	0.000000	

An example of Customer profile

In this case we can see that customer 1 has a value of 0.010823 for bakery cuisine. In other words, about 1 percent of their orders were bakery items. From this we know that this customer does not order bakery goods that often and hence, the recommendation system will less be likely to recommend bakery based restaurants.

Restaurant Profile:

Restaurant profile represents the cuisine offered and also the cuisine that is popular based on frequency of customer orders.

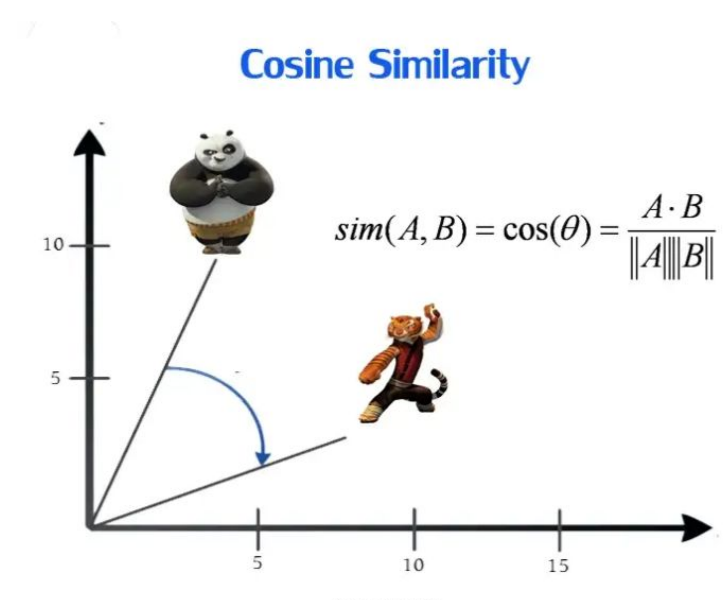
To create our restaurant profile we combined two different intermediate vectors. The first vector is created using our filter applied onto the menu items the restaurant offers. This gives us a general idea of what the restaurant focuses on. We call this vector the **Restaurant Profile.** The second vector is created using the 10 million orders skip provided us. We use these orders to infer what the most popular items from a restaurant are. We call this vector the **Order Profile.**

Finally we use a **weighted combination** of both of these intermediate profiles to create a final generalizable profile which showcases what cuisines the restaurant is known for.

An example of Restaurant_Profile:

	short_name	African	Alcohol	Bakery	Barbecue	\
0	1600 world bier haus	0.0	0.005814	0.017442	0.023256	
1	17th ave liquor boutique	0.0	0.720437	0.008202	0.000684	
2	1886 buffalo	0.0	0.000000	0.106742	0.067416	
3	1886 buffalo cafe	0.0	0.000000	0.011765	0.047059	
4	19 kitchen cooking nw	0.0	0.000000	0.000000	0.291667	

In this we can see that restaurant 4 has a Bakery value of 0.29. This value takes into account two factors: the popularity of Bakery cuisine among customers based on their order history, and the focus index of Bakery cuisine in the restaurant's menu. In other words, the Bakery value reflects both how much customers like Bakery cuisine and how much the restaurant emphasizes it in their offerings.



What do we do with the profiles?

Using the customer profile and restaurant profile, we have the following stages to produce a set of final recommendations, where each stage applies unique filters and modifications:

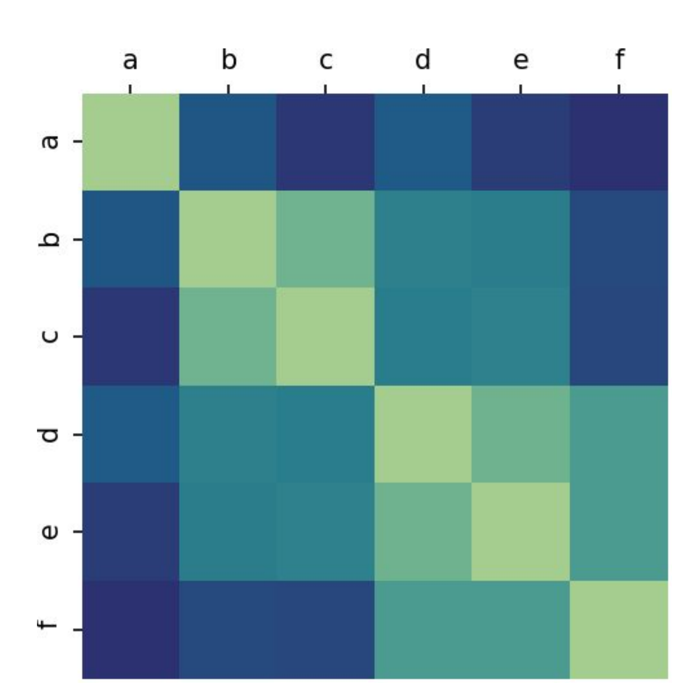
Base recommendations:

For each customer profile we find the most similar restaurants by applying cosine similarity calculation in each restaurant profile. In other words, Let's say for Customer A profile, we run cosine similarity with all the restaurant profiles and then pick the top 5 similar restaurants based on the cosine similarity score. The following picture demonstrates the cosine similarity calculation process:

The top 5 restaurants are based on customer's preference because it's recommended based on their order history

Filter using Cosine similarity matrix:

We want our recommendations to have variations and recommend new restaurants that they might like as well. To achieve this, we have created a similarity matrix of cuisines, that have values between 0 - 1. The similarity matrix indicates how similar the cuisines are to each other based on the order history of customers We use the similarity matrix to replace the top 3 cuisines of customer profile with the most similar cuisines. To be more specific, for each of the top 3 cuisines, top 5 similar cuisines are calculated. And the cuisine to replace from the top 5 list is randomized, so every time a customer signs in to the app, they get new recommendations.. After the top 3 cuisine gets replaced, we run the cosine similarity calculation again on the "Updated_Customer_Profile", and pick the top 5 restaurants based on the cosines scores. The recommendations now won't be based on the cuisine they like, but it will be based on cuisines that they might like.



Filter using Skip score:

Skip the dishes has Skip score ratings for all of their restaurants and our last filter incorporates that in our recommendations. The idea is to give more weighting to restaurants with higher skip scores. The default weighting of skip score is 0.1, which means that skip score will account for 10 percent of the final score. The skip score addition is applied to both base recommendation and recommendation using the similarity matrix.

After the addition of skip score, the new values then will be used to pick top 5 restaurants from the base recommendation and top 5 recommendations from the recommendation using the similarity matrix. So, in total of 10 restaurants will be displayed to the user. This number of restaurants displayed is customizable and can be easily changed in the backend python code.

Results

We had some people participate in a restaurant recommendation system test through our front-end website. They simulated their orders several times by selecting food items and dates, and then the front-end finally gave five restaurant recommendations based on taste and five based on a similarity matrix. These four diagrams represent the level of satisfaction with the results they received