Recipe Recommender Executive Summary

Nadir Hajouji, Felix Almendra Hernandez, Nathan Schley, Ali Arslanhan, Katherine Martin

Goal Build a recommendation engine that suggests recipes to users.
Data We used recipe and review data that was scraped from food.com and made available on Kaggle.

- We had a recipe dataset containing instructions, ingredients, descriptions, etc. for 500,000+ recipes.
- We also had a review dataset containing 2 million ratings and reviews for the recipes in the dataset.

Stakeholders

- Our final product can be used as a standalone product that people use to obtain recipe suggestions (e.g. à la New York Times Cooking).
- The recommendation engine can also be used to assist with targeted advertising - apps like Instacart are already offering (unsolicited) recipe suggestions.

Data cleaning and Processing

- The information in the recipe dataset was inconsistent from recipe to recipe, and some of the information was not relevant for our purposes. We distilled the content in the recipe dataset into a set of keywords and used multi-hot encoding to obtain a "clean" dataframe.
- We ultimately decided not to use the numerical ratings in the review dataset; our modeling only considers which recipes a user has reviewed.

The data obtained from the review dataset was distilled into an affinity matrix, where each row of the matrix corresponds to a user, each column a recipe, and where the (i, j) entry is 1 if user i reviewed recipe j, and 0 otherwise.

Modeling Problem

This is the precise question we wished to answer:
If we know a subset of the recipes reviewed by a user, can we predict other recipes that the user is likely to interact with?

- We used the training data to construct a function f : Users × Recipes $\rightarrow \mathbb{R}$. We interpret f as follows: If f(user, recipe1) $>$ f(user, recipe2), then our model expects that the user is more likely to interact with recipe1 than recipe2. *In theory, the values of f can be arbitrary. We will scale f so that it assigns a mean score of 1 to user-recipe pairs in the training set, and then compare the scores assigned to user-recipe pairs in the holdout set to the global average value of f.*
- To obtain such a model, we embed the users and recipes into a common vector space and define f(user, recipe) to be the dot product of the associated vectors.

Results

- We used singular value decomposition to obtain matrix factorizations of the affinity matrix, interpreting the matrix factorization as the desired set of embeddings of recipes and users into $\mathbb{R}^2$.
- The average (normalized) score given to recipes in the holdout set was 0.803, compared to an average score around 0. Thus, our model does a good job predicting which pairs were left out of the training data.

Final Product

- We created a web app that allows users to enter a freeform text query and receive a list of recipe recommendations. The recommendation is based on a weighted score that combines the user's likelihood of interacting with the recipe (computed using the model we trained) and a score that quantifies how similar the recipe description is to the user's query. To obtain the latter score, we used a pretrained sentence transformer to convert the query and recipe descriptions to vectors.