Personalized Recipe Recommendation Using Heterogeneous Graphs

Nicholas DeGroot
University of California, San Diego
La Jolla, CA 92122
ndegroot@ucsd.edu

Abstract

[TODO]

1 Introduction

We've all been there. It's been a long day of work, but you're finally home and ready to cook. The only problem: you have no idea what to make.

- You could fall back on some classics, but it feels like you've been eating the same thing for weeks.
- You could go out to eat, but that's expensive and you're trying to save money.
- You could order takeout, but that's unhealthy and you're trying to eat better.

What's needed is a way to find new recipes that you'll actually enjoy, personalized to the things you already have on hand. This project accomplishes exactly this. The result is a completely automated recipe recommendation system that will take into account the ingredients you have on hand, your dietary restrictions, and your personal preferences to recommend weekly meal plans that you don't need to think about. Users should be able to open up the app, see what's been scheduled for the day, and start cooking. Should users not like what's been scheduled, it'll be easy to swap out recipes for something else and incorporate that feedback back into the model for future meal plans.

To my knowledge, nothing like this is available on the market. Existing services generally fall within one of two categories.

- **Recipe Aggregators:** These services provide a collection of recipes that users can browse and search. Users are expected to find the recipes they want to cook themselves. Some services such as Yummly have integrated personalized search recommendations to make finding recipes easier, yet require users to manually build out their meal plans.
- **Meal Kits:** These services provide pre-portioned ingredients and recipes for users to cook. Users select from pre-determined plans (such as Meat & Veggies) and are sent a box of pre-portioned ingredients with their associated recipe each week (Hello Fresh). Minor customization is possible, but users are locked into a limited number of recipes. These services can become quite expensive and often require users to commit to a subscription.

Each service has its own strengths and weaknesses. Recipe aggregators are free and allow users to cook whatever they want, but require users to do all the work of finding recipes and building out meal plans. Meal kits are convenient and allow users to cook without having to think about it, but are expensive and require users to commit to a subscription. This project combines the best of both worlds: providing a cheap, personalized, and low-effort meal planning service.

We accomplished this using a variety of graph-based models, which differ from the more traditional approaches used by other recommendation services. Graph-based models are able to capture more complex relationships between users and recipes, and are more easily able to combine additional information such as ingredients.

2 Methods

To build out our recipe recommendation system, we used an existing dataset published by Majumder u. a. (2019) based of food.com reviews. After transforming the data and loading it into TigerGraph, we were left with 231,636 recipes across 1,132,366 interactions. Each interaction was associated with a particular user and their rating from a scale of 1-5. Furthermore, we were able to extract additional information about each recipe, including the ingredients and any tags associated with it. We used this data to train a variety of graph-based models, including collaborative filtering, node embedding, and deep learning models. We then evaluated the performance of each model using a variety of metrics.

2.1 Collaborative Filtering Models

We started with a classic collaborative filtering model. The model works in three steps.

For any given user u:

1. Find all recipes r that u has interacted with.
2. Find all users O that have interacted with the same recipes as u (r).
3. Calculate the average rating for each user in O ($\bar{R}_o$)
4. Calculate the similarity score for each user in O using pearson correlation, where I_u is the set of recipes that u has interacted with:

$$Sim(u,o) = \frac{\sum_{i \in I_u \cup I_o}(R_{u,i} - \bar{R}_u)(R_{o,i} - \bar{R}_o)}{\sqrt{\sum_{i \in I_u \cup I_o}(R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_u \cup I_o}(R_{o,i} - \bar{R}_o)^2}}$$

5. Calculate the rank of each recipe r using the following formula, where U_r is the set of users that have interacted with recipe r:

$$Rank(r) = \sum_{o \in U_r} Sim(u,o) \cdot (R_{o,r} - \bar{R}_o + 1)$$

While such a model isn't exclusive to graph-based networks, using one does allow the model to perform more efficiently in finding the relevant users/items.

It's important to note that while most recommendation systems filter out recipes that users have already interacted with, we decided to keep them in the model. This is because we wanted to be able to recommend recipes that users have already interacted with, but may have not tried in awhile. This is particularly important to note when considering the results of our user research, which suggested that users generally prefer to cook recipes they've already cooked before.

2.2 Node Embedding Models

While the collaborative filtering model was able to capture some of the relationships between users and recipes, it was unable to capture more complex relationships. For example, if a user has interacted with a recipe that contains a particular ingredient, we would like to be able to recommend other recipes that contain that same ingredient. To accomplish this, we next implemented a node embedding model.

For any given user u:

1. Use the FastRP algorithm (Chen u. a., 2019) to generate a node embedding for each user.
   We used an embedding dimension of 5 over 3 iterations.
   We allowed the algorithm to walk across all User, Recipe, and Ingredient nodes.

2. Calculate the similarity of u with every other user using euclidian distance over their FastRP embeddings. Calculate the rank of each recipe r using the following formula, where U_r is the set of users that have interacted with recipe r:

$$Rank(r) = \sum_{o \in U_r} Sim(u, o) \cdot (R_{o,r} - \bar{R}_o + 1)$$

This has a number of advantages over the collaborative filtering model. Mainly, it allows us to capture more complex information between users and recipes. For example, if a user has interacted with a recipe that contains a particular ingredient, we would like to be able to intergrade that information into the recomendation. Our model captures this through the FastRP embeddings, which calculates latent information about each ingredient and forwards it through the model to each user.

2.3 Deep Learning Models

[TODO]

3 Results

[TODO]

4 Discussion

[TODO]

5 Bibliography

[Chen u. a. 2019] CHEN, Haochen ; SULTAN, Syed F. ; TIAN, Yingtao ; CHEN, Muhao ; SKIENA, Steven: *Fast and Accurate Network Embeddings via Very Sparse Random Projection*. 2019. – URL `https://arxiv.org/abs/1908.11512`

[Hello Fresh] HELLO FRESH: *Fresh Food & Meal Kit Delivery Service*. – URL `https://www.hellofresh.com/`

[Majumder u. a. 2019] MAJUMDER, Bodhisattwa P. ; LI, Shuyang ; NI, Jianmo ; MCAULEY, Julian: Generating Personalized Recipes from Historical User Preferences. In: *EMNLP 2019*, URL `https://arxiv.org/abs/1909.00105`, 2019

[Yummly] YUMMLY: *Yummly: Personalized Recipe Recommendations and Search*. – URL `https://www.yummly.com/`

A Appendix

Optional.