Recipe Recommender

CSE 158R WI 25 Assignment 2

Abstract

Interest in dietary diversity, health management, and personalized meal planning has been increasing drastically as dietary habits directly impact physical health, mental well-being, and chronic disease prevention. This has led to an increase in individuals seeking tools to guide their food choices. This report presents the design and implementation of a recommender system that recommends recipes to users based on the relationship between their past recipe review ratings and the recipe's nutritional values. The system is developed to assist users in discovering meals that align with their dietary preferences, health goals, tastes, and nutritional needs. By leveraging a dataset of recipes with detailed nutritional information, the system employs a recommendation approach combining content-based filtering, which matches users' ratings of recipes, with collaborative filtering using the nutritional macros of recipes from similar users.

This work highlights the practical potential of combining advanced recommendation algorithms with nutritional science, offering a valuable tool for promoting healthier eating habits and simplifying meal planning.

1. Dataset

We searched for a suitable dataset with enough data to build a comprehensive model that could effectively train and validate our model and also provide the right information relevant to our parameters. This led us to the Food.com Recipes and Interactions dataset. This dataset consists of 180K+ recipes, 25.1k users, and 700K+ recipe reviews covering 18 years of user interactions and uploads on Food.com (formerly GeniusKitchen) and was used in the paper "Generating Personalized Recipes from Historical User Preferences" (2019) by Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley.

From this dataset, the files we used for our model were:

- PP_users.csv: each of the users' id and information pertaining to their respective reviews like the recipes they reviewed and the ratings they gave.
- PP_recipes.csv: each of the recipes with details including recipe id, calorie level, and ingredients.
- RAW_recipes.csv: details about each recipe like minutes to prepare, nutrition information, and the number of steps.
- interactions_test.csv: the data split for the test portion of the model
- interactions_validation.csv: the data split for the validation portion of the model
- interactions_train.csv: the data split for the train portion of the model

Additionally, to better support our model, we used the above files to create the following files to encompass the information that we needed to train our recommender more efficiently.

- recipes_made_train.csv: generated train dataset from interactions_test.csv that contains both negative and positive data of whether a user made a given recipe
- recipes_made_test.csv: generated test dataset from interactions_test.csv that contains both negative and positive data of whether a user made a given recipe

- recipes_made_val.csv: generated validation dataset from interactions_val.csv that contains both negative and positive data of whether a user made a given recipe

It is important to take note that the three data splits are separated by interactions, or the 700k+ recipe reviews, rather than by recipes or by users. There are 694,434 interactions in the training dataset, 6,954 interactions in the validation dataset, and 12,271 interactions in the test dataset.

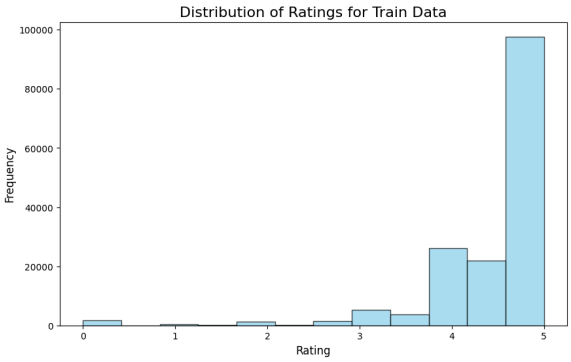

Figure 1: Distribution of Review Ratings in Training Data with no sample minimum

An observation depicted in Figure 1 was that the distribution of ratings observed was heavily skewed towards 5 stars. Intuitively, this makes sense because users would tend to only try recipes they would want to make and would most likely enjoy. Similarly, recipes would usually only be posted if they were to be deemed successful and likely to be enjoyed by others. This creates a survivorship-like bias where the data is mostly comprised of liked recipes among the users. Not only that, but some recipes are lacking in rating data such that there are only one or two ratings on multiple recipes. The lack of data destroys the purpose of the mean since it was influenced by a few users.

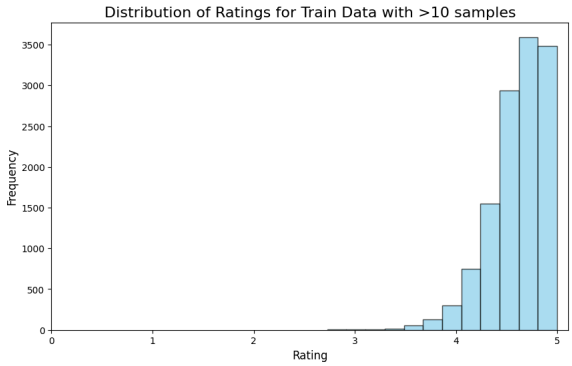

Figure 2: Distribution of Review Ratings in Training Data with more than 10 samples

As seen in Figure 2, the rating data becomes much more nuanced with more ratings. This signifies that there are many users who find low-popularity recipes and review them poorly, possibly regardless of the quality of the recipe. This also depicts that there are recipes with no ratings that were set as 0. It is also important to notice that even with a minimum sample size, the ratings still skew heavily in favor of 4.5 and above meaning that this feature may be hard to predict or hard to use for prediction.

This is important to keep in mind when building the model around predicting recipes that the user would make and highly rate.

2. Task

We want to build a recommender system model that can predict whether or not a user would make a recipe based on the nutritional information of the recipe, their past recipe review ratings, and the ratings of similar users. The effectiveness of this model can be measured using the accuracy of its predictions on the test dataset. The higher the accuracy, the better fit the model is to predict recipes and this result would be significant if it gives an output with greater accuracy than our baseline model. We

will use a popularity-by-rating model as the baseline and compare if using the nutritional value as a parameter will lead to more accurate predictions compared to popularity-by-made, logistic regression, and collaborative filtering.

As described in the previous section, Section 1, the pre-processing we did was to reformat the raw data into datasets that better fit our model. The features we chose to include were how long it took to make the recipe, the nutritional details for the recipe, the number of steps it took, and the number of ingredients used. These were all columns in the initial, more generalized files provided by the original dataset so we reformatted the ones relevant to new files to use to train our model.

3. Models

3.1 Popularity-By-Rating (Baseline)

With interaction data, the intuitive and most straightforward solution would be to use some popularity model. As a baseline, we used a popularity-by-rating model. This model tracks the highest-rated recipes by mean and keeps the top 50% as the “most popular”.

Due to the nature of our data, a few issues with using this model arose. As shown by Figure 1, the distribution of the average ratings skewed heavily towards 5/5. As discussed in Section 1, many recipes lack reviews, so the average rating is 5.0. To combat this, a minimum number of ratings of 6 was chosen to weed out any recipes lacking data. This number was chosen due to the balance between the amount of data it removed and the amount it kept. This will be the standard going forward.

We tested our initial model on the testing set yielding an accuracy of 59.61%. This indicates that this simple model is doing some accurate predictions, but is still quite random.

3.2 Popularity-By-Made

This model works similarly to the Popularity-By-Rating model but has a different motivation. As previously stated, a large problem in our dataset is that our recipe for user interaction is sparse. Focusing on the most popular recipes by their uses rather than their rating helps reduce bias that is caused by a lack of reviews. Running this model on the test set yields an accuracy of 68.97%.

3.3 Logistic Regression

Logistic regression is great at finding linear divisions between two classes. The big step with this model is the introduction of supportive data rather than just interactive data. RAW_recipes.csv contains a plethora of information for a given recipe_id. Categories such as time to make, tags, name, calorie level, macros, ingredients, etc. are all present. The motivation behind this model is to see if there is a relationship between what users choose to make and the statistics of the recipes.

The columns used as data were *minutes, nutrition n_steps, and n_ingredients*. Minutes, n_steps, and n_ingredients were all preprocessed by removing outliers and standardizing. Nutrition had to be parsed into a list and then further split into *calories, total fat, sugar, sodium, protein, saturated fat, and carbohydrates*.

This model had an accuracy of 51.1% with an AUC-ROC score of 0.518. The AUC-ROC score signals how well a model is predicting the data, regardless of correctness. An AUC-ROC score closer to 0.5 means the model is predicting at pure randomness. With a score of 0.518, our model had minimal predictive power. This signifies that the recipe statistics had little correlation with which recipes a user chose to make.

3.4 Content-Based Filtering

Content-based filtering works to provide a user with highly specified recommendations based on a user's preferences and an item's characteristics. A similar style of preprocessing recipes from Section 3.2 was used to get the item feature matrix. The similarity between items was calculated using cosine similarity.

$$Cosine\ Similarity(u,p) = \frac{\sum_i u_i p_i}{\sqrt{\sum_i u_i^2}\sqrt{\sum_i p_i^2}}$$

Figure 3: Cosine Similarity Formula

Cosine similarity was used over other distance formulas due to the high dimensionality of our recipes. Thus, this formula will best describe the similarities between recipes.

This model was tested on the test set and got an accuracy of 50.2%. Although the poor results are surprising, the reasons for why this happened will be discussed in Section 5.

Another model that is in progress is a hybrid model between content-based filtering and collaborative filtering. The idea is to take the best of both worlds, interaction data, and recipe characteristics, and combine them into a single model. The idea follows this question:

$$\gamma_{u,i}^{Hybrid} = \alpha \cdot \gamma_{u,i}^{CF} + (1 - \alpha) \cdot \gamma_{u,i}^{CBF}$$

Figure 4: Hybrid Model Formula

Simply put, both models will be used simultaneously and the strength of the alpha value will be optimized to yield the best results. The motivation behind this model is that it can take advantage of the vast number of features per recipe, and the large pool of interaction data between user and recipe.

4. Literature

Our dataset came from a dataset on Kaggle that crawled Food.com, an online recipe aggregator, for its data. This dataset has been used to build multiple recommender models such as recipe recommenders, diet recommenders, and recipe generators. Since this dataset is segmented into three parts, train, validation, and test, it allows models to be holistically completed through training it to build the initial models, optimizing and fine-tuning it through the validation set, and testing it to see how accurate the system is.

There are also a lot of other datasets using recipes but used to construct models for a different purpose. For instance, the recipe ingredient dataset on Kaggle was used in a competition to train a predictor to predict the type of cuisine a recipe was given the ingredient list. One solution opted to use semantic clustering and the relative distribution of ingredients within each cuisine to predict what type of cuisine an unknown recipe would be.

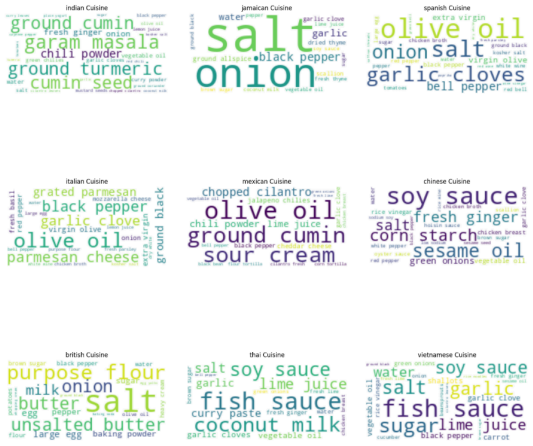

Figure 5: Word Cloud Visualization of Ingredient Distribution per Cuisine

As depicted in the above figure, if the unknown recipe had a similar ingredient distribution to

one of the trained distributions by the model, it would be predicted to belong to that cuisine.

Another example of a similar dataset would be the recipe box dataset from Eight Portions, which includes similar information as the dataset used in our research, with the addition of pictures. This enabled this dataset to be used for a variety of deep learning projects including a recipe prediction using word and emoji embeddings, a title generator given the ingredients and instructions of a recipe, and a food image generator using Generative Adversarial Networks (GANs). The food image generator model utilized a Deep Convolutional GAN (DCGAN) and a Wasserstein GAN (WGAN) to generate novel images of food. Each of the GANs was fitted on the recipe images and only the images.

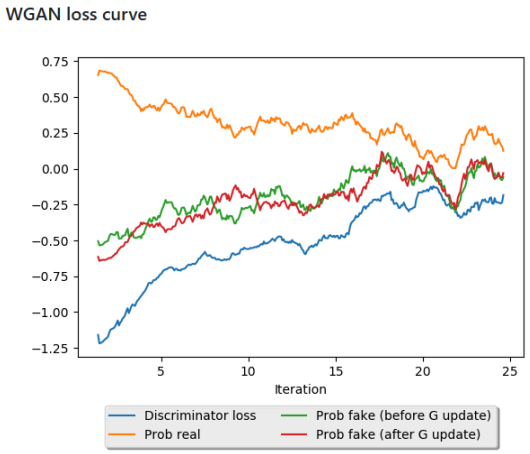

Figure 6: WGAN Loss Curve

As shown in the figure above, as the number of iterations increases for the model training, the loss curve converges to zero, improving its accuracy.

As described in this section, there are many methods currently employed to build machine-learning models around recipe datasets, including but not limited to semantic clustering, word embedding, and GANs. Both semantic clustering and word embedding fall into the field of Natural Language Processing (NLP) which has been a field of interest for many scholars within Artificial Intelligence for the past few decades and is still becoming more advanced. NLP can be used for recipe datasets by making predictions on likely words that would appear in a query recipe. Contrary to the methods relating to NLP, GANs are relatively new to machine learning and there exist many variants of GANs that are more specialized towards a specific kind of image or product generation. For recipe datasets, models can use GANs to generate images for a recipe based on existing images for recipes.

The conclusions drawn from these existing works are different than ours because they use different models for different motivations. Since our recommender system didn't fall into the fields of Natural Language Processing or Generative Adversarial Networks nor have a similar predictive task, it is difficult to draw concrete conclusions that compare our work and others. However, since we used multiple models and each of our models outputted the accuracy of the models run on the test set, we were able to compare our models with each other and draw conclusions based on them which will be further discussed in the following section.

5. Results

Of the models built and tested, the model that performed the best was the popularity model based on the number of users that made the recipe, with an accuracy of 68.97%. The worst-performing model made was our Content-Based Filtering model which had an accuracy of 50.2%. All our models were above 50%, however, the delta positive for both our logistic regression and Content-Based Filtering

model was too small to show any significance. This result was caused by a few key things.

A possible interpretation of the Popularity-By-Made model performing relatively well is that this is because users are more likely to try recipes that are already popular, so any popular recipe is more likely to be made by an unknown user. This interpretation is supported by the "Popular" tab on Food.com to find popular or trending recipes, which would point users towards already popular recipes rather than a more uniform random spread.

Also, the Popularity-By-Made model was one of the few models that was able to fully capitalize on the large dataset. Due to its simplicity and space efficiency, it was able to effectively process through all 700k data points. Not only that but the number of unique users and recipes in the training set was only ~25K meaning that it was able to learn about the popularity of each recipe very well. One note about this model is that it revealed a large discrepancy between the interactions_train.csv and the interaction_test.csv files. By running this model on the test set, it was revealed that there was only about a 1% overlap between the recipes in the training data and recipes in the test data. This would cause the model to yield a perfect 50% accuracy since there were no recipes it recognized. A train and test set were created by the interaction_train.csv file to fix this, but later models were effected by this issue.

The nature of the dataset did not allow more complex models like content-based filtering to thrive. A statistic of this is shown within the content-based model where out of the 24,000 users in the test set, only 1,274 were known. This means it had a massive cold-start problem where a majority of the users were unknown. This led the model to predict the default value of "not made" for many of the users, and since the negative data comprises exactly half of the test set, an accuracy close to 50% is expected.

Another issue was the size of the dataset itself. The cleaned train data for our predictive task contained 694,702 entries. In our CBF and CBF-CF-Hybrid model, we lacked the processing power to process all 700k entries as the feature matrix would reach into the billions of features. Therefore, we had to settle on ~10k entries to create the feature matrix out of. Although this may sound like plenty, there are ~25k unique users within the training set. This inevitably causes us to miss users exacerbating the cold-start problem.

As mentioned in the first section, since there was such a heavy skew towards "good recipes" or recipes that were highly rated (5 stars out of 5), there wasn't much data for "bad recipes" to train the model on information that would push the user away from a recipe. This information would benefit the CBF model since it would allow it to filter out "negative" content, or content that would be predicted to deter the user from a given recipe.

With these findings, recommending recipes based on popularity by the number of times it was made is the best solution out of the features tested in this study. Utilizing a different dataset that is crawled differently could potentially make a different model perform better. Further optimization of the models tested would be hyperparameter tuning or the inclusion/exclusion of different features provided by the dataset.

For future works, natural language processing would be a good next step on this dataset. The RAW_recipes.csv file is rich in text data in the form of recipe titles, descriptions, steps (in words), and tags. All of these features are rich in information that can be tapped into using NLP in combination with a classic recommender.

References

[1] Kaggle. (2017, January 19). *Recipe ingredients dataset*. https://www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset

[2] Li, S. (2019, November 8). *Food.com recipes and interactions*. Kaggle. https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions/data

[3] rtlee9. (n.d.). *RTLEE9/Food-Gan: Novel Food Image Generation Using gans*. GitHub. https://github.com/rtlee9/food-GAN

[4] Ryan Lee. (2017, March 14). *Recipe box*. Eight Portions. https://eightportions.com/datasets/Recipes/#fn:1

[5] suraj5424. (2024, February 21). *Cuisine_prediction_semantic_solution*. Kaggle. https://www.kaggle.com/code/suraj5424/cuisine-prediction-semantic-solution