Food for Thought: How Do You Build a Better Recipe Recommender?

1 Dataset

For our assignment, we used two datasets: *RAW_recipes.csv* and *RAW_interactions.csv*, both of which were collected from Food.com by the research team led by Majumder et al. For further details on Majumder et al.'s research project and their data usage, please refer to the Literature Review section 4.

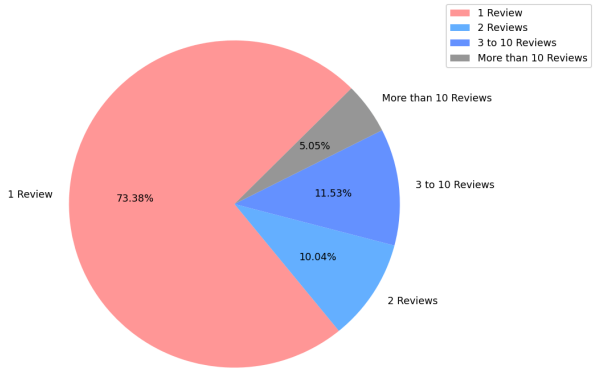
1.1 User Interaction Dataset

The user-recipe interaction dataset from *RAW_interactions.csv* contains 1,132,367 reviews spanning an 18-year period (2000–2018). This dataset covers a total of 231,637 unique recipes and includes interactions from 226,570 distinct users. Each data entry consists of: user_id, recipe_id, date, rating, and review (refer to Table 1 below).

User Interactions Dataset	
Column Name	Description
user_id	Unique id of user
recipe_id	Unique id of recipe
date	Date of review *(YYYY-MM-DD)*
rating	Recipe rating (scale: 0 to 5)
review	Text of recipe review
Total Reviews: 1,132,367	
Years of data collection : 2000 - 2018	

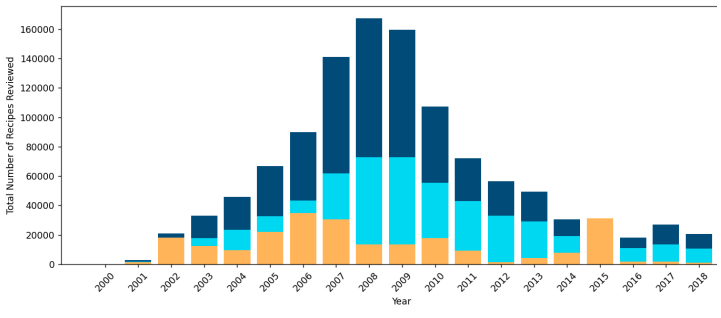
Figure 1 illustrates the distribution of reviews per user. It shows that 73.38% of users in the dataset contributed only a single review, while 5.05% of users made more than 10 reviews. This highlights an important characteristic of the dataset: despite having a user-item history, we have relatively limited information about individual users, which could impact the performance of certain models.

Figure 1:



In addition, our Figure 2 illustrates our analysis of the distribution of **all** reviews per year (dark blue bar), **all unique** reviews per year (unique - none share a recipe they correspond to)(light blue bar), and all unique reviews that were their recipe's **first received review** within the 18-year data span (yellow bar). Figure 2 illustrates that after around 2010, there was a noticeable decline in the number of first reviews for recipes. However, 2015 stands as an exception, as almost all reviews that year were for recipes that had never been reviewed before then.

Figure 2:



Most reviews received a rating of 5, as shown in Figure 3. However, there was a general decline in the average rating for recipes starting around 2008. As shown in

Figure 4, while the average ratings have remained consistently above a rating of 4 each year, there has been a gradual decrease beginning in 2011, with the most significant drop occurring in 2017. By 2018, there was no drastic recovery in average ratings. This decline is likely attributed to the reduction in the overall number of reviews after 2011, meaning the average rating is increasingly based on a smaller and potentially less varied sample of data each year.

Figure 3:

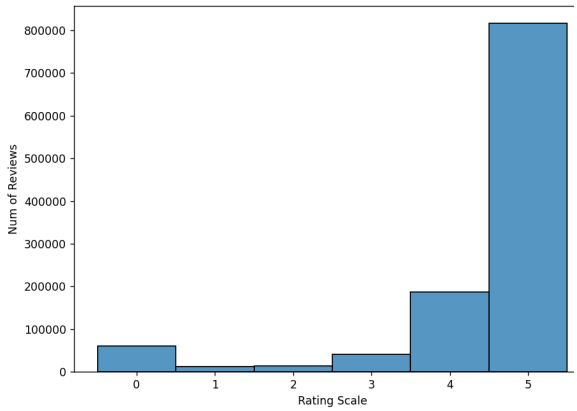
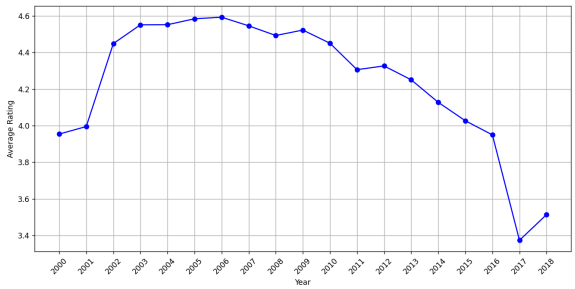


Figure 4:



Before starting to create a model, we wanted to gain an overview of the "positive" reviews and the "negative" reviews. We did this by creating 2 sets of word clouds. The first approach categorized reviews as positive or negative based on the rating (3 or higher for positive, and 2 or lower for negative).

The second approach for categorizing positive vs negative reviews was based on the actual content of the review itself, using sentiment analysis. For this, we applied TextBlob to calculate the sentiment polarity of each review, which assigns a score to the text indicating whether the sentiment is positive, negative, or neutral. Based on the polarity score, we classified the reviews as positive (if the score was greater than 0), negative (if the score was less than 0), or neutral (if the score was exactly 0). This allowed us to classify the reviews independently of their ratings. Unfortunately, neither set of word clouds (Figure 5 and 6) provided a deeper insight into the reviews.

Figure 5:

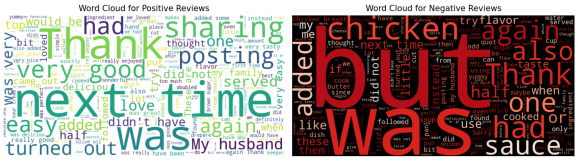


Figure 6:

1.2 Recipe Dataset

The recipe dataset from *RAW_recipes.csv* contains 231,637 data entries spanning a 19-year period (1999 – 2018) from 27,926 distinct users.

Each data entry included the following columns: name, id, minutes,

contributor_id, submitted, tags, nutrition, n_steps, steps, description, ingredients, and n_ingredients. For a list of column descriptions please refer to Table 2 below.

Recipe Dataset	
Column Name	Description
name	Name of recipe given by contributor
id	Unique id of recipe
minutes	Duration to make recipe
contributer_id	Unique if of user who submitted recipe
submitted	Date recipe submitted (*YYYY-MM-DD*)
tags	Food.com tags for recipe
nutrition	Nutrition information (calories (#), total fat (PDV), sugar (PDV) , sodium (PDV) , protein (PDV) , saturated fat
n_steps	Number of steps in recipe
steps	Text of recipe instructions
description	Description of recipe
ingredients	List of ingredients
n_ingredients	Number of ingredients in recipe
Total Entries: 231,637	
Years of data collection : 1999 - 2018	

Most recipe entries have an average of 8 ingredients, with the maximum number of ingredients recorded at 20 (Figure 7). The wide range of ingredient counts suggests a mix of simple and more intricate recipes.

Figure 7:

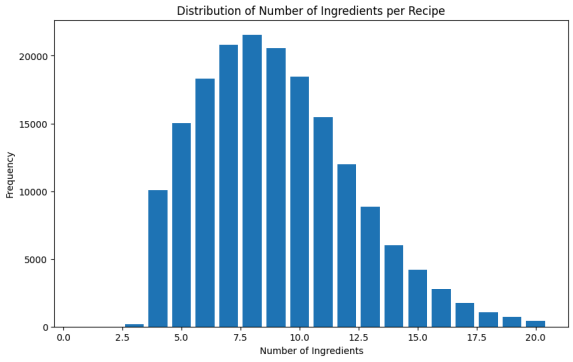
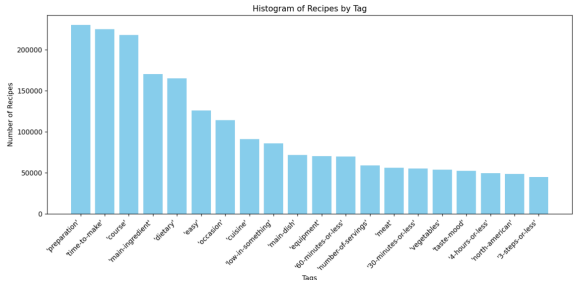


Figure 8:



The “tags” column had 209,115 unique tags. Figure 8, shows the top 20 tags across the recipe dataset. We then performed a brief thematic analysis and placed the top 20 tags in the following categories: Time-related, Course, Dietary, Cuisine-related, Difficulty/Convenience, and Occasion-related. A breakdown of these categories can be found in Table 3 below.

Categories for Top 20 Tags	
Categories	Tags
Time-related	Preparation Time to make 60 min or less 30 min or less 4 hours or less 3 steps or less
Meal-related	Course Main dish Num of servings
Ingredient-related	Main ingredient Meat Vegetables
Dietary	Dietary Low in something
Difficulty/Convenience	Easy Equipment
Occasion-related	Occasion Taste mood
Cuisine-related	Cuisine North American

2 Predictive Task

The task we have selected for our assignment is to predict the rating that a

user will give for a recipe, on a scale of zero to five.

2.1 Features

To properly predict how any given user would rate a recipe, a series of features has been extracted and created from the given dataset; namely the user_id, recipe_id, and rating fields. To properly use these features to train and evaluate our model, we performed a simple 80%/10%/10% split for our train/validation/test datasets and utilized them for both our baseline models and our main models.

2.2 Evaluation Methodology

To properly evaluate how well our model's predictions perform compared to their respective true rating, we will calculate the Mean Squared Error (MSE), which is defined below:

$$MSE = \frac{1}{n}\sum_{i}^{n}(Y_i - \widehat{Y_i})^2$$

where Y_i is the true rating value and $\widehat{Y_i}$ is the estimated rating value.

Our exploratory data analysis showed that both the average rating users give and the average rating a recipe receives give a reasonable approximation for predicting future rating values. Based on this fact, we defined three simple baseline models to get a sense of the room for further improvement.

2.3 Baselines

User-Average Model

The user-average model predicts ratings based on the user's past behavior. It first checks if the user has a history of providing ratings by looking up the user in the dictionary of user averages previously constructed. If the user is found, it returns the average rating that this user typically gives to items. This approach assumes that users tend to be consistent in their ratings, so their average rating can be a good predictor of their future behavior. However, if the user is new or unknown (i.e., not found in the user dictionary), the model defaults to using the global average, which is the average rating across all users and all items. This fallback ensures that the model can still make a reasonable prediction even when there is no specific data about the user.

Item-Average Model

The item-average model works similarly but focuses on the item instead of the user. It looks at the average rating for the specific recipe or item being predicted. If the item has been rated by users before, the model returns the average rating for that item, assuming that its average rating reflects its overall quality. However, if the item is unknown or hasn't been rated before, the model defaults to the global average. This method is useful when items have distinct patterns of ratings but can be less effective if there isn't enough information on a particular item.

Jaccard Similarity Model

The Jaccard Similarity model takes a more nuanced approach by incorporating the idea of similarity between items, using a metric called the Jaccard similarity. The formula is given as follows:

$$r(u,\ i) = \overline{R}_i + \frac{\Sigma_{j \in I_u}(R_{u,j} - \overline{R}_j) \cdot Jaccard(i,j)}{\Sigma_{j \in I_u} Jaccard(i,j)}$$

This model first looks at other items that the user has rated and calculates how similar those items are to the item in question based on the overlap of users who rated both items. The ratings of these similar items are adjusted (weighted) by their similarity to the target item. If the weighted sum of these ratings can be computed, the model predicts the rating based on this information. However, if the user hasn't rated any similar items (i.e., there's no overlap in users who rated both items), the model defaults to the global average. This method allows for more personalized predictions by factoring in how similar the user's previous items are to the new ones.

2.4 Performance Evaluation

Model	MSE
User-Average	1.3087376
Item-Average	1.4270654
Jaccard Similarity	1.4209063

After evaluating our baseline models, we found the MSE values for each baseline model to be well over 1.3. To see if we could further improve the accuracy of our prediction, we developed and evaluated a User/Item Average Hybrid Model, a Latent Factor Model, and a Neural Network-based Model, which we explore in the next section.

3 Model

User/Item Average Hybrid Model
The very first approach we tried was combining the user rating average and recipe rating average with weight factors. Through a hyperparameter tuning process of weight values, we achieved an MSE value of 1.29767. Although the result was an improvement from the baseline models, it was quite insignificant, so we delved further into more complex models.

Latent Factor Model
Each user-recipe interaction is represented with (1) global bias; (2) user bias; (3) item bias; (4) latent factors for item and user.

Justification
The Latent Factor Model is simplistic and effective, particularly with sparse data that only contains user-recipe interactions. The Latent Factor Model performs well without requiring us to parse and provide explicit features from raw data files which can be quite challenging. It is great at personalizing predictions by learning patterns solely from user-recipe interactions, using latent factors to capture preferences and characteristics. Additionally, our familiarity with the Latent Factor Model from class made it a natural choice. We have seen that this method performs well on the type of problem we are solving.

Optimization
The optimization is performed using the L-BFGS-B algorithm, which minimizes the loss function:

$$Cost = MSE(predictions, labels) + Regularization\ Terms$$

We have added regularization terms for user and item biases and latent factors to prevent the model from overfitting and ensure decent generalization to unseen data.

Then we performed a grid search to find the combination of regularization terms that would decrease the validation error of our model.

Issues:
We have run into issues with the train, validation and test sets that were already provided with the data set. These sets were organized in a way that the newest user review would go into test, second newest to validation, and the rest of the reviews under that user - to training[1]. We learned from the data analysis that the majority of users only have 1 review. This means that there isn't a lot of overlap between the training set and validation or test sets. This led to the model defaulting to a certain baseline, that would predict the same rating no matter how we changed the parameters. This resulted in an MSE of 1.92. We managed to solve this issue by reshuffling the given datasets and performing our own splitting. This introduced overlap which made the model perform a lot better, achieving an MSE of 0.90. This reshuffling approach was also useful for us when experimenting with other models, including the Neural Network Model.

In addition to this, after we tuned the parameters, we wanted to find paths to further improve our model. We started to think of ways how we could use extra features from raw data to our advantage. We reviewed the paper "Improving Latent Factor Models via Personalized Feature Projection for One-Class Recommendation" [9], which discusses an approach to expand traditional latent factor models by incorporating user-specific personalized projection matrices for item features, rather than using generalized latent factors. This approach appeared promising, as it would allow us to model more complex, user-specific relationships with the item features. However, due to time constraints and the challenges involved, we decided to explore other models that would allow us to use raw data features.

Neural Network Model
The final approach we attempted is a neural network that takes in various features as inputs and predicts a single rating as output. In total, we attempted two different models, one with some basic features, and another with more advanced features. The architecture for both models is similar. Model 1 has an input layer, a dense layer with 128 neurons with relu activation and 0.5 dropouts, a dense layer with 64 neurons with relu activation and 0.5 dropouts, and finally a single output layer. Model 2 uses the same architecture but with an extra 64 neuron Dense layer.

We wanted to try using a neural network to better capture some of the more complex relationships between the features that we have to work with.

The following is a description of the features that we used:

Number of Steps (Both Model 1 and 2)
We took the similarity between the number of steps that the user prefers (this is the average # of steps in ratings the user rated 4 or higher in the training data) and the number of steps in the recipe as a feature.

We used the normalized similarity measure below:

```
def normalized_similarity(x, y):
    denominator = max(abs(x), abs(y), 1)
    return 1 - abs(x - y) / denominator
```

Number of Ingredients (1 and 2)
We used the same method as above to use the similarity in the number of ingredients in user preferences and the recipe.

Jaccard Similarity (1 only)
We used the maximum of the Jaccard similarities between recipes that the user has rated and the recipes that the users of the target recipes have rated as a factor.

Technique Vector (1 and 2)
One part of the dataset is a list of techniques found within recipes. We use the normalized version of the vector that represents the techniques the user has encountered.

Calorie Level (2 only)
We use the normalized calorie level of the recipe as a feature.

Ingredients Jaccard (2 only)
We take the Jaccard similarity of the ingredients in the recipe and the set of preferred ingredients of the user (set of all ingredients in recipes the user rated 4 or higher) as a feature.

Ingredients TFIDF (2 only)
We run the following TFIDF function on the list of ingredients that the user prefers, and use the TFIDF of that ingredient as a feature:

```
for ingredient in target_user['favorite_ingredients'].keys():
    tf = target_user['favorite_ingredients'][ingredient] / len(target_user['favorite_ingredients'].keys())
    idf = math.log(len(PP_recipes_list) / ingr_id_to_count[ingredient] + 1)
    ingredients_tfidf = max(tf*idf, ingredients_tfidf)
```

Evaluation Results
The first model yielded an MSE of 0.88.
The second model yielded an MSE of 0.79.

These results, while good, are not a significant improvement over previous methods, in particular compared to the LFM. This combined with the low improvement in loss after each epoch suggests that, at least for the features that we used, creating a full-fledged neural network may not be an efficient use of computation power for this task.

The two main challenges of using neural networks were dealing with the computational requirement and selecting effective features. My laptop couldn't handle training a neural network locally, so I had to import my code onto Kaggle and use their GPU resources.

The long runtime it takes to generate the features and train the network meant that it was difficult to tune hyperparameters and experiment with features and model architectures.

Selecting the correct features to maximize performance was also a challenge, as some features that intuitively seemed like they would be good were a detriment. For example, the Jaccard similarity didn't improve the model that much after experimentation, and was scrapped in the second model. Also, a version of the technique vector that took into account which techniques the recipe had increased the loss, so the simpler version that only included user techniques was used.

4 Literature Review

Between 2006 and 2023, 70 research papers have focused on food recommendation systems. The main objectives of these studies include improving users' health by providing nutritional information, enhancing the quality of datasets and recommender system research, and generating personalized recommendations based on factors such as preferred cuisine, ingredients, and available cooking time due to users' careers [7].

In food recommendation systems, the three most common filtering techniques are Content-Based Filtering (CBF), Hybrid Filtering (HF), and Collaborative Filtering (CF) [7].

Recent methods for personalizing food-related data often use an *encoder-decoder framework*, where data is tokenized to allow the machine to process and expand it. *Aspect-aware learning* is then employed to extract latent representations from users' past reviews, such as preferred cooking methods, food types, or motivations for trying a recipe. This enables the model to infer user preferences. The *attention fusion layer* further highlights specific review details, such as cooking techniques, that are crucial for generating personalized recommendations. [1][2]

Other approaches leverage neural networks to learn latent representations of user preferences and item features [3][4][5]. Some methods have also combined Collaborative Filtering with Generative Concatenative Networks [6].

4.1 Context of Dataset Used

We used the dataset collected by Majumder et al. from Food.com. For more details on the dataset's properties, please refer to Section 1, "Exploratory Analysis."

In their study, Majumder et al. developed a model to generate personalized recipes based on a dish name, the desired calorie level, and a partial list of ingredients. This approach is more realistic, as users often only know a few core ingredients. The personalized framework also incorporates the user's most recent recipe interactions (reviews) and preferred cooking techniques, such as boiling, grilling, baking, or mixing. The model's goal is to generate a coherent, personalized recipe.

This approach differs from previous studies, which typically provide models with a full list of ingredients and the dish name, without any personalized framework. In contrast, prior recipe generation models are evaluated based on how well they use the full ingredient list

and the coherence of the generated recipe. While Majumder et al.'s study also evaluates coherence, they take it a step further by assessing whether the personalized framework enhances the "semantic plausibility"—the believability and accuracy—of the generated recipes.

4.2 Food Datasets in Prior Studies

According to the literature review by Mahajan et al., of the 70 papers reviewed, 20 used datasets that are not publicly available due to privacy concerns or terms of service restrictions. As a result, researchers often have to collect their own food review datasets [7]. This is evident in our case, as the dataset we used was manually collected by Majumder et al.'s research team.

The 2024 paper, *"An interactive food recommendation system using reinforcement learning"* [8] also utilized a dataset from Food.com, which they discovered via the publicly available Kaggle site. This is the same platform we found the dataset we are working with.

5 Conclusions

In conclusion, in this report we explored two datasets from Food.com, focusing on user-recipe interactions and recipe details. Through an extensive exploratory data analysis, we uncovered key characteristics of user behavior and recipe features, such as the predominance of single-review users and the decline in new recipe reviews after 2010. Our predictive task involved building and evaluating several models to predict user ratings of recipes, starting with baseline models like user-average, item-average, and Jaccard similarity-based predictions. These baseline models provided foundational insights but left room for improvement, as indicated by their Mean Squared Error (MSE) scores.

To improve accuracy, we developed more sophisticated models, including a User/Item Average Hybrid Model, a Latent Factor Model (LFM), and a Neural Network-based model.

The LFM, with an MSE of 0.90, effectively modeled sparse data using global, user, and item biases alongside latent factors, revealing user preferences and recipe characteristics without extensive feature engineering. Regularization ensured generalization and avoided overfitting.

The Neural Network, leveraging complex features like steps, ingredient similarities, and calorie levels, achieved the best MSE of 0.79 but showed marginal improvement over the LFM. High computational costs and limited gains suggest neural networks may not be optimal for this task. While the LFM offered simplicity and interpretability, the neural network's non-linear features made its outputs less transparent, emphasizing the importance of feature selection, efficiency in model design and the challenges of incorporating complex features into food recommendation systems

Our review of related literature revealed similar findings in other studies, particularly in the growing use of neural networks and collaborative filtering for personalized food recommendations.

These findings affirm the broader research trends, emphasizing the importance of leveraging user preferences and recipe characteristics to enhance recommendation accuracy. Overall, our analysis demonstrated the strengths and limitations of various approaches to food recommendation tasks, paving the way for future advancements in personalized recipe generation and recommendation systems.

6 References

[1] Majumder, Bodhisattwa Prasad, et al. "Generating personalized recipes from historical user preferences." arXiv preprint arXiv:1909.00105 (2019).

[2] Ni, Jianmo, and Julian McAuley. "Personalized review generation by expanding phrases and attending on aspect-aware representations." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2018.

[3] Wu, Chao-Yuan, et al. "Joint training of ratings and reviews with recurrent recommender networks." (2017).

[4] Catherine, Rose, and William Cohen. "Transnets: Learning to transform for recommendation." Proceedings of the eleventh ACM conference on recommender systems. 2017.

[5] Li, Piji, et al. "Neural rating regression with abstractive tips generation for recommendation." Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 2017.

[6] Lipton, Zachary C., Sharad Vikram, and Julian McAuley. "Capturing meaning in product reviews with character-level generative text models." arXiv preprint arXiv:1511.03683 (2015).

[7] Mahajan, Pratibha, and Pankaj Deep Kaur. "A Systematic Literature Review of Food Recommender Systems." SN Computer Science 5.1 (2024): 174.

[8] Liu, Liangliang, et al. "An interactive food recommendation system using reinforcement learning." Expert Systems with Applications (2024): 124313.

[9] Zhao, Tong, Julian McAuley, and Irwin King. "Improving latent factor models via personalized feature projection for one class recommendation." Proceedings of the 24th ACM international on conference on information and knowledge management. 2015.