Recipe Reccomender

Abstract

In this project, we created a recipe recommender system that is tailored to the user's personal taste. The data set used featured users' interactions with recipes in the form of ratings, and so to get a binary indication of whether a recipe is something that should be recommended to a user or not, a simple rating threshold was used. If the rating was above a threshold, then the item should be recommended, otherwise, it shouldn't. For this reason, the primary focus of this project was on rating prediction. Based on estimated ratings, we decide whether we are going to recommend a recipe to a user or not. This recommendation is based on the users' preference indicated by their rating history, as well as the implicit nutritional biases they have.

1 Introduction

With the increasing demand for food deliveries such as Doordash and UberEats, we can see people's eating habit is changing. They tend not to cook at home compared to the past. One of the attributes is that it is time-consuming to select recipes and prepare and cook a meal in busy daily life. We believe that there is a need for decision-aid systems that can suggest personalized recipes while taking into account the user's preferences and eating history.

The most popular recommendation technique is collaborative filtering. this approach uses the users' preferences like the rating or review to predict what they might like. These preference ratings are often based on the similarity between any given item, and items previously rated by the user. There are two typical types of collaborative filtering, namely, memory-based and model-based filtering. A Memory based approach is based on some notion of 'similarity' among items. That is, an item is recommended to a user because it is similar to one that they have recently consumed. Conversely, a model-based approach adapts the regression and classification problems of estimating interactions between users and items. In other words, we are concerned with fitting models that take a user *u* and item *i* as inputs in order to estimate an interaction label *y* such as rating.

We focus on predicting a user's rating for new recipes using collaborative filtering. For the memory-based approach, we used recipe steps and ingredients of recipes to represent each item's property. For the model-based approach, we focus on estimating interactions between users and items (as well as making use of other nutritional features of each item) using a latent-factor model. Our overall model uses a blending technique to merge the outputs of these separate models into one definitive prediction.

2 Dataset and its Basic Properties

For the purposes of this project a data set of recipes from food.com (previously known as GeniusKitchen). This data set includes the metadata seen in Table 1.

Recipe	
recipe name	string
recipe id	int
minutes	int
nutrition	list(float)
n steps	int
steps	string
n ingredients	int
ingredients	list(str)
tags	list(str)
submitted	time
description	str

Interaction	
user id	int
recipe id	int
rating	int
review	str
date	time

Table 1: Metadata present in dataset

These tables were merged to create the overall interaction data set used throughout this project.

2.1 Numerical Data

A number of the features of the data used in this project consisted of numerical values. The rating of a given recipe is an integer value from 0 to 5. The recipe nutrition information consists of 7 components including; calories, total fat, sugar, sodium, protein, saturated fat, and carbohydrates. Also, the cooking time and the number of recipe steps also provided in numerical format.

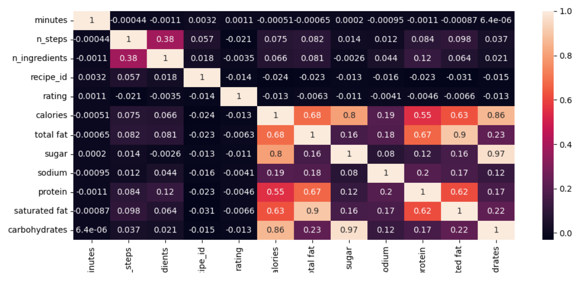


Figure 1: Correlation map between numerical features

Figure 1 shows the correlation between ratings and the given numerical features. Clearly, there is a high correlation between nutritional information. This means that even though this information isn't directly correlated with ratings, if the dietary preferences of a user can be sufficiently modeled, then similar recipes can easily be found and recommended to them.

2.2 Recipe ID, User ID

The recipe IDs and user IDs are given as unique integers for any given recipe.

Figure 2 shows how many recipes there are with any given number of interactions. It is evident that the majority of recipes have a very low number of interactions, meaning they have only been reviewed a handful of times. This may indicate that there is more of a need to find a similarity rating between recipes since it is unlikely for a lot of the recipe that many users will have interacted with them.

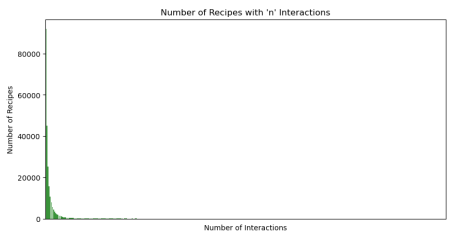


Figure 2: A graph showing the number of recipes with any given number of interactions.

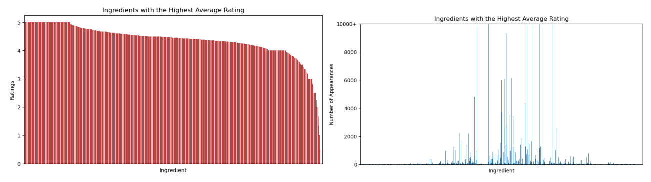
2.3 Ingredients

The list of ingredients is given as a string of the form: `"['ing1', 'ing2', 'ing3',...]"`. So in order to pre-process it all that needed to be done is to parse the string into a list of strings.

2.3.1 Ingredient to Rating Relationship

The first step of this analysis was to take the average rating for each ingredient across the whole data set. The relationship between ingredients and their average ratings (having sorted the list in descending order of average rating) is given in Figure 3a.

This is a surprising result since there are many extreme values for average ingredient ratings. When looking at the number of occurrences for each of the ingredients in the first graph (see Figure 3b), we can see that the ratings for the ingredients with fewer occurrences tend to have much more extreme average ratings.



(a) Average ingredient rating. (b) Ingredient occurrences.

Figure 3: Graphs showing the average rating (Figure 3a) and number of occurrences (Figure 3b), of ingredients. Both graphs are ordered in descending order of average rating.

In Figure 4 we can see that the effect of removing ingredients with fewer than 500 occurrences reduced this effect and the average ratings are all more densely packed around the mean rating of ~ 4.2.

2.4 Steps

The steps for each recipe are given in as a string of the form: `"['step1', 'step2', 'step3',...]"`. In order to create an embedding for each word in the recipes' steps, there was

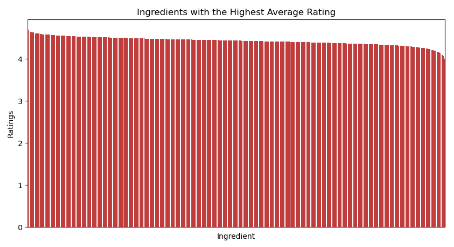
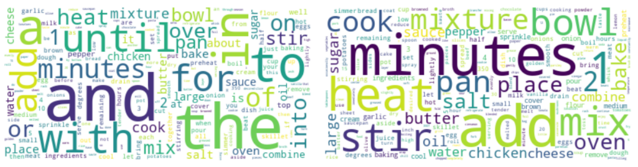


Figure 4: Average rating distribution when least common ingredients removed.

no need to break the steps feature into a list of strings. Instead, punctuation was removed, and every letter was converted to lowercase. This left a string of words from every step in any given recipe.

Finally, before analyzing the data, stop-words were removed since they generally have little importance in a document. A comparison of the word clouds representing word frequencies in the steps feature over the whole data set with and without stop-words being removed can be seen in Figure 5.



(a) All words. (b) Without stop-words

Figure 5: Word-clouds made from all recipe steps with stop-words removed (Figure 5b) and without stop-words removed (Figure 5a).

It is clear from Figure 5 that the most common words in the recipe steps are stop-words (e.g. 'and' and 'the').

Looking at Figure 6, we can see the number of words remaining in the steps feature of the data set with varying levels of pre-processing. The number of unique words before removing stop-words was 56786 whereas removing stop-words resulted in 56468 unique words. This indicates that although the number of stop-words was relatively small (318), they made up the majority of the words in the recipe steps. If stemming was also used, the number of words fell more drastically to 45323. However, the stemmer used often removed vital information from the words, meaning the overall performance of any model using the data would perform worse.

3 Identified Predictive Task and Analysis of Relevant Features

We focus on predicting ratings for new recipes to determine whether to recommend a recipe to a user or not. First, We used ingredients and steps of recipes to get similarities between recipes (named the memory-based approach in subsection 3.1). To get representations of ingredients and steps, we leveraged word2vec and TF-IDF methods respectively. Second, we made use of latent factor machines (named the model-based approach in subsection 3.2) to model user-item interactions while also making use of nutritional features from the recipes.

The significance of a feature was determined by the MSE when predicting ratings for unseen interaction data using only that feature. If the MSE obtained did not achieve a good R^2 score (i.e. the method resulted in predictions no better than always guessing the mean), then the feature was regarded as insignificant. The final MSEs for the models using each feature can be seen in subsection 6.2.

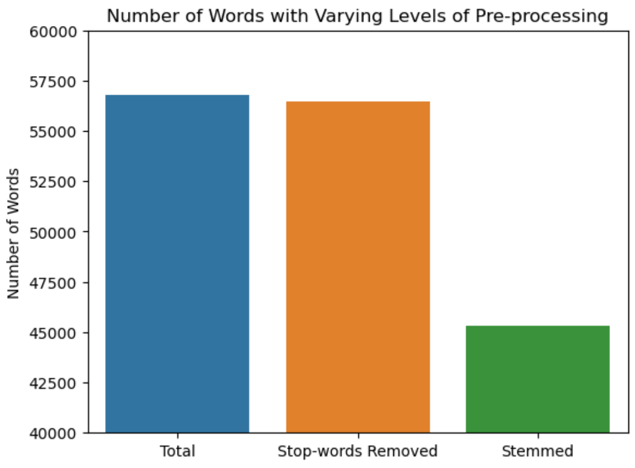


Figure 6: Number of words in steps feature with varying levels of pre-processing.

3.1 Memory Based Approach

Equation 1 shows the memory-based collaborative filtering approach for rating prediction. Given an interaction between a user u and an item i, we want to predict the rating attributed to it r_{ui}. This equation is in essence a weighted average of the ratings in the user us history. The weight for this average is given by a similarity function (some functions tested are given later in this section).

$$r_{ui} = \frac{\sum_{j \in I_u} sim(i,j) \cdot rating(j)}{\sum_{i,j} sim(i,j)} \tag{1}$$

3.1.1 Word2Vec Representation for Ingredients

Figure 4 Shows that once we remove the least common ingredients the average rating per ingredient globally is not the most informative (since the averages are very densely centered around the mean value). For this reason, there a more meaningful word representation was required that indicated the similarity between any two ingredients.

Given the pre-processing done for the ingredients feature in section 2, there is a list of ingredients present for any given recipe. Therefore it is possible to create a matrix of ingredients and the ingredients surrounding them in the given recipes, as seen Equation 2.

$$\mathbf{X} = ingredients' \left\{ \underbrace{\begin{bmatrix} 0 & 1 & 1 & \dots \\ 1 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{ingredients} \right. \tag{2}$$

Now we can create low-dimensionality representations for each ingredient as well as the ingredients surrounding it in the recipe. Since the matrix **X** is binary, these low dimensionality vectors need to be passed through a sigmoid function, as seen in Equation 3.

$$\mathbf{X} = \sigma\left(\begin{bmatrix} \gamma_{ingredient'_1} \\ \gamma_{ingredient'_2} \\ \vdots \end{bmatrix} \cdot \left[\gamma_{ingredient_1}, \gamma_{ingredient_2}, \dots\right]\right) \quad (3)$$

For this, we used the `word2vec` model to find and use these low-dimensionality ingredient vectors. The package allowed for the retrieval of ingredients similar to each other.

Now that we have low-dimensional representations for each ingredient, we can use this information as a similarity rating between users in Equation 1. The way that the similarity rating between item i and item j was calculated is shown in Algorithm 1. Note that this algorithm uses the `ingListVec` function, which gets the `Word2Vec` vector of all the ingredients and returns their average vector.

Algorithm 1 Getting similarity between users using recipe ingredients feature.

```
$r\_pred \leftarrow 0$
$norm\_fact \leftarrow 0$
**if** u **in** userHistory **then**
    **for** j **in** userHistory **do**
        $v1 \leftarrow ingListVec(recipeIngredients[i])$
        $v2 \leftarrow ingListVec(recipeIngredients[j])$
        $sim \leftarrow Cosine(v1, v2)$
        $r\_pred \leftarrow r\_pred + sim * allInteractions[(u, i2)]$
        $norm\_fact \leftarrow norm\_fact + sim$
    **end for**
    $r\_pred \leftarrow r\_pred / norm\_fact$
**else**
    $r\_pred = 3.0$
**end if**
```

Algorithm 1 uses the Cosine similarity between two averaged ingredient list vectors to define the similarity between two recipes.

3.1.2 Item2Vec Representation for Recipe IDs

A similar dimensionality reduction to the one done previously for ingredients can be done for recipe IDs. The starting matrix for this new reduction is given in Equation 4. Note that in this matrix we only include 'nearby' recipes that were reviewed within a certain timescale of the given recipes. To achieve this the recipes were ordered in terms of the date they were rated before passing into the word2vec model. The reason for this is to make use of temporal patterns in the data.

$$\mathbf{Y} = recipes \left\{ \underbrace{\begin{bmatrix} 0 & 1 & 1 & \dots \\ 1 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{nearby\ recipes} \right. \quad (4)$$

Now we can create low-dimensionality representations as done previously. For this, we again use the `word2vec` model to use these low-dimensionality recipe vectors to find recipes similar to each other. Note that in this case the `window` functionality in the model, so that only reviews submitted in a closer time-frame to each recipe are included in the matrix **Y**.

With these low-dimensional representations for each recipe, we can again use this information as a similarity rating between users in Equation 1. The way that the similarity rating between item i and item j was calculated is shown in Algorithm 2.

Algorithm 2 Getting similarity between users using recipe ids feature.

```
**if** u **in** userHistory **and** i **in** mostSim.values **then**
    $r\_pred \leftarrow 0$
    $norm\_fact \leftarrow 0$
    **for** (j, sim) **in** mostSim(i, topn=n) **do**
        **if** j in userHistory **then**
            $r\_pred \leftarrow$
            $r\_pred + sim * allInteractions[(u, j)]$
            $norm\_fact \leftarrow norm\_fact + sim$
        **end if**
    **end for**
    $r\_pred \leftarrow r\_pred / norm\_fact$
**else**
    $r\_pred = 3.0$
**end if**
```

Note that in Algorithm 2 we are only using the n most similar recipes to each other. If we look at Figure 7, we can see that as we increase the value of 'n', the MSE of the model decreases, and the number average number of recipe matches for each query increases. However, we can see that on average there is only around 1 recipe in any given user's history that matches the query (which agrees with the analysis done in section 2). Overall this means that the use of recipe IDs as a similarity metric is not the most informative, and the MSEs are only ever marginally below if the mean value was always predicted.

3.1.3 TF-IDF Representation for Recipe Steps

In Figure 5b, where stop-words are removed from the data set, it is evident that there are still many common words that dominate the term frequencies even though they might not be the most informative (e.g. 'cook' or 'heat'). Therefore, in

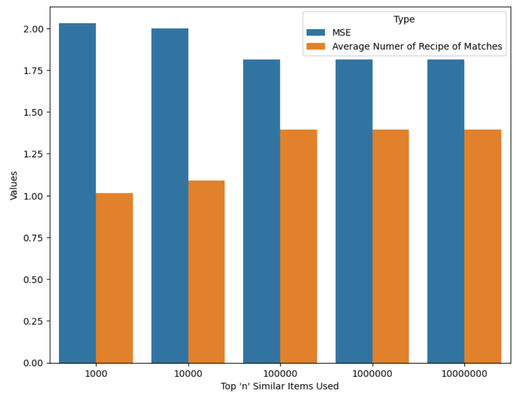
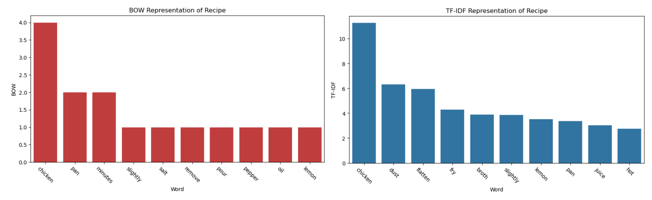


Figure 7: A plot showing the relationship between the top ‘n’ most similar recipes being chosen and the MSE/Avg Number of recipe matches.

order to obtain a more informative representation of each step, the TF-IDF rating was taken for each word in the recipe steps.

Given that every word that appears in the data set is given an embedding, we can compare the TF-IDF representation and the BOWs representation of the recipe for ‘15 minute garlic lemon chicken’ in Figure 8.



(a) BOW. (b) TF-IDF.

Figure 8: Two plots showing the BOW representation (Figure 8a) and the TF-IDF representation (Figure 8b), of the recipe for ‘15 minute garlic lemon chicken’.

Figure 8 clearly shows that TF-IDF extracts words from the recipe steps that convey more information. For example, words such as ‘fry’, ‘broth’, and ‘lemon’ are much more highly rated in TF-IDF, whereas terms like ‘salt’, ‘pepper’, and ‘oil’ were more highly rated in the BOW representation.

Now that we have the TF-IDF representation of any given recipe, we can find the similarity between two recipes in an algorithm similar to Algorithm 1. The only difference would be that rather than having a statement such as `ingListVec(recipeIngredients(i)`, we would instead simply use `tfidf(i.steps)`. As well as this the number of words used for the recipe embeddings was reduced to the 1000 highest TF-IDF words to reduce memory usage. It was found that this simplification did not have a drastic effect on the MSE of the model.

3.2 Model Based approach

A latent factor model was used in the model-based approach, the equation for which is given in Equation 5. Each term is explained in the following sections.

$$f(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i + (\beta_{cal} + \gamma_u \cdot \gamma_{cal}) \tag{5}$$

3.2.1 User-Recipe Interaction

Model-based recommender is that there is some underlying low-dimensional structure among the interactions. Interaction matrices R capture real-valued interaction signals which are the ratings. Using matrix factorization, R can be simplified in lower dimensional space, given by γ_U and γ_I. Intuitively, γ_U might be thought of as describing the ’preferences’ of the user u, whereas γ_I describes the ’properties’ of item i. Their interactions are modeled via inner product, as seen in Equation 6.

$$\underbrace{\begin{bmatrix} \quad R \quad \end{bmatrix}}_{|U|X|I|} = \underbrace{\begin{bmatrix} \gamma_U \end{bmatrix}}_{|U|XK} \times \underbrace{\begin{bmatrix} \qquad \gamma_I^T \qquad \end{bmatrix}}_{KX|I|} \tag{6}$$

3.2.2 User-Calorie Interaction

To get a more detailed feature representation of a user’s preference, we added a latent factor for Calories. As calories are a real-valued feature, we divided calories into five levels based on minimum, first quartile, median, third quartile, and maximum value. We assigned each calorie value to one of the integer levels to generate a latent factor for calories (γ_{cal} in Equation 5). The way this was done is in a very similar way to what was done for user-recipe interactions in subsubsection 3.2.1.

$$argmin_{\gamma} \frac{1}{|R|} \sum_{(u,i) \in R} (f(u,i) - R_{u,i})^2 \tag{7}$$

So far we have modeled interactions in terms of latent user and item factors. In Equation 5 we also used offset term which is α and bias terms for users, items, calorie levels which are β_u and β_i and β_{cal} to prevent under(or over)predict ratings. We fit the model to predict the ratings by optimizing both latent factor parameters following the optimization equation as seen in Equation 7.

4 Description of Models Used

In section 3, we described various separate models used to predict ratings. Our final model aimed to improve the prediction accuracy above any single one of those models by combining them. The way this was done was by taking the output of each separate model and passing it into a parent model. This meant that we captured both the recipes' characteristics and the user's preference histories. In other words, the outputs of each model were used as features in a parent model, which blended them to create a final prediction.

The blending method used as the parent model was a linear regression model that took the outputs of each separate model and found the optimal weighting to give each prediction. As such the equation that dictates the final model output is given by Equation 8, where the optimal θs are being found by the linear regressor, and the r_{ui} terms are the ones being output by our previous models.

$$\begin{gathered} r_{ui} = \theta_{bias} + \theta_{ingredients} \cdot r_{ui}(ingredientSim) + \\ \theta_{steps} \cdot r_{ui}(stepsSim) + \theta_{latent} \cdot r_{ui}(latentModel) \end{gathered} \tag{8}$$

The evaluation of this model is also done using the MSE value. The aim was to achieve a superior MSE above the MSE of the individual models. The reason this was used as the metric is since if a lower MSE is achieved, then the rating predictions are more accurate to the actual ratings given by the users. This in turn means that the prediction of whether to recommend a recipe is more accurate since this data is produced by thresholding the actual review given by the users.

5 Relevant External Literature & Research

The data set used for this project is trawled from food.com. It was initially used in a paper by Bodhisattwa Prasad Majumder *et al.* [1] titled 'Generating Personalized Recipes from Historical User Preferences'. In this paper, a new task of personalized recipe generation to help these users was proposed. Given a name and incomplete ingredient details, complete natural text instructions aligned with the user's historical preferences was generated. A complete pipeline of this model is given in Figure 9.

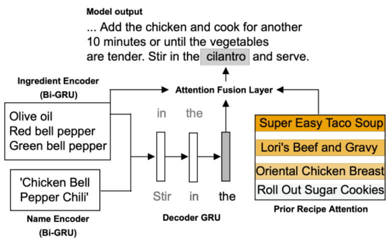


Figure 9: Sample data flow through model architecture. Emphasis on prior recipe attention scores (darker is stronger). Ingredient attention was omitted for clarity.

A paper written by M. B. Vivek *et al.* [3] used similarity between users and items to predict ratings of unseen interactions. Two approaches were implemented; an item-based approach and a user-based approach. For item based approach various similarity functions were used to compute similarities between different recipes. For the item-based approach, the Tanimoto Coefficient similarity (See Equation 9, where N_a is the number of customers who rates item A, N_b is the number of customers who rates item B, and N_c is the number of customers who rate both items A and B) and log-likelihood similarity were used, whereas for the user-based approach Euclidean Distance and Pearson Correlation between user histories were used.

$$T(a,b) = \frac{N_c}{N_a + N_b - N_c} \tag{9}$$

Both these methods are examples of memory-based collaborative filtering approaches (similar to the ones employed in this project), meaning recommendations were given to users based on preferences indicated by historical user ratings and interaction data. The data used in this case was trawled from allRecipes.com.

A paper by Ueta, T. *et al.* [2] proposed a goal-oriented recipe recommendation system to enable users to search for recipes with to improve specific health conditions. An example of such a request is 'I want to cure my acne'. The co-occurrences of 45 common nutrients with nouns such as cold, acne, bone etc. were listed. Then we created a recipe database of recipes from www.cookpad.com was obtained and the system analyzed each recipe to calculate the amount of a nutrient in any given dish. A simplified version of this approach was used in this project since in our case the user's nutritional needs are implied in the latent factor model. In this paper, results were compared to recipes obtained by calculating the nutrient information manually.

6 Results & Analysis

In this section, there is an outline of the analysis done during the production of the final model, as well as the individual models that feed into it. As well as this there is a comparison of the results from the individual models with the final model created.

6.1 Hyper-parameter Tuning

6.1.1 Memory based Approach

To get the TF-IDF representation of words in steps, we used 1000 most common words in all recipe steps. This gave a similar MSE to using every word while reducing the vector representation size (unlike when tests were done with stemming).

6.1.2 Latent Factor Model

There are four hyper-parameters that needed to be tuned in this model;

- **The number of latent factors(K)**

  We used 10 as K. This allowed for the model to not be too powerful, which would be much easier to over-fit to training data, while still being able to capture meaningful representations.

- **The regularization term(lambda)**

  For regularization, we use 0.000001 as lambda for all latent factors including beta and gamma. This term prevented the learned weights from being too large, allowing for the model to perform better on unseen data.

- **The learning rate(lr)**

  0.001 was used as the lr, which was ideal to converge steadily, making sure not to overshoot the minimum.

- **The batch size**

  A batch size of 1024 was used.

6.2 Final Results

We tested each model with 2000 test samples, and the resulting MSEs for each model can be seen in Table 2. As you can see, the model-based approaches generally got lower MSEs than the similarity-based approaches. This is because latent factor approaches are able to model much more complex interaction patterns than simply observing a user's history.

For the model-based approach, we hypothesized that the performance may improve if we consider more interaction terms like user-calorie. However, the performance got worse than when we only used the user-recipe interaction information. This may be because the patterns found by the second model-based approach in the training data didn't generalize to the testing data due to over-fitting or lack of supporting information.

However, when we combined the memory-based approach and the second model-based approach in the blended (hybrid) final model, the MSE was far lower than any of the individual models. We concluded that the patterns found in the users' review history by the memory-based approach, as well as the user-item and user-calorie patterns found by the model-based approach were all useful features for predicting the rating. The linear regression model was able to find an optimum combination of all these features to create a superior recommender system.

6.3 Future Work

For future work on the model-based approach, we can find influential interaction terms using factorization machines that model arbitrary pairwise interactions between features, rather than using latent factor models. As well as this, more of the recipe features such as nutritional information could be used to create an even more powerful model.

Approach	**Feature**	**MSE**
Memory-based	Ingredient (word2vec)	1.644
-	Steps (TF-IDF)	1.683
Model-based	User, Recipe	1.465
-	User, Recipe, Calorie	1.513
Hybrid	*all above*	**1.439**

Table 2: The MSE on testing data for various models

For the memory-based approaches, experiments could be done to change the averaging method to get the recipe embedding vectors. Additionally, experiments could be done using text analysis on other recipe features such as tags to see if they can be used to improve the prediction MSE.

References

[1] MAJUMDER, B. P., LI, S., NI, J., AND MCAULEY, J. Generating personalized recipes from historical user preferences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 5976–5982.

[2] UETA, T., IWAKAMI, M., AND ITO, T. A recipe recommendation system based on automatic nutrition information extraction. In *Knowledge Science, Engineering and Management* (Berlin, Heidelberg, 2011), H. Xiong and W. B. Lee, Eds., Springer Berlin Heidelberg, pp. 79–90.

[3] VIVEK, M., MANJU, N., AND VIJAY, M. Machine learning based food recipe recommendation system. In *Proceedings of international conference on cognition and recognition* (2018), Springer, pp. 11–19.