Food recipes recommender system using Neural Collaborative Filtering content based

Sebastián Salgado
Eduardo Contreras
Gonzalo Fuentes
ssalgadn@uc.cl
econtreraslazcano@alumni.uc.cl
gonzalo.fuentes@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Región Metropolitana, Chile

Abstract

Food is a daily necessity, and individuals often seek ways to diversify their meals to avoid monotony. Numerous online platforms provide a vast collection of recipes, making recipe recommendation systems an esential tool for assisting users in discovering new culinary options. In this study we explore personalized recipe recommendation systems by combining content-based and collaborative filtering approaches. Food.com provides a dataset which includes over 1.4 million reviews for over 500 thousand recipes with detailed attributes.

We implemented three variations of a Neural Collaborative Filtering (NCF); **NCF Basic**, **NCF Nutrients** and **NCF Full**. These models increasingly include features such as user reviews, nutritional information and recipe embeddings derived from images and instruction processed using ResNet50 and BERT. We compared our models against standard baselines, including Random, Most Popular, FunkSVD, LightGCN, and MultiVAE. Evaluation metrics, such as Recall@10 and NDCG@10, indicated that MultiVAE outperformed other methods, while Most Popular achieved competitive results, outperforming other methods in Precision@10.

Our NCF models demonstrated incremental performance improvements as more content were added, providing valuable insights into the impact of content diversity on recommender systems.

CCS Concepts

• **Networks** → **Network performance analysis**; • **Applied computing** → Marketing.

Keywords

Recommender Systems, Neural Collaborative Filtering, Content Based Recommender Systems, Machine Learning

ACM Reference Format:
Sebastián Salgado, Eduardo Contreras, and Gonzalo Fuentes. 2024. Food recipes recommender system using Neural Collaborative Filtering content based. In *Proceedings of ACM (Final project, RecSys).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

Final project, RecSys, December 09, 2024, Santiago, Chile

ACM ISBN 978-1-4503-XXXX-X/24/12
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Social media has fostered an entire culinary world unique to the internet, ranging from extravagant dishes like those prepared by Gordon Ramsay to more traditional ones crafted by Yuhui Lee, a renowned contestant from MasterChef Chile. The Food.com platform serves as a social hub where users can share their own recipes and rate those of others. Within this context, facilitating the search for personalized recipes for each user becomes an intriguing challenge, as everyone has unique tastes and preferences when it comes to food.

Our methodology combines collaborative filtering with content-based techniques to enhance the recommendations further. Beyond user ratings, we incorporate diverse content features such as visual and text representations with structured data

By integrating these components, our recommender system aims to provide highly personalized and accurate recipe suggestions, catering to the unique preferences of each user. The ultimate objective is to enrich the user experience on Food.com, turning it into a platform that not only connects food enthusiasts but also serves as a reliable tool for culinary inspiration and exploration.

2 Related work

In the domain of recipe recommendation systems, various approaches have been proposed integration innovations from Machine Learning techniques to enhance personalization and accuracy in ranking recommendations. Prominent works in this subject are:

(1) *Enhancing Personalized Recipe Recommendation through Multi-Class Classification*: Neelam and Veerella [6] address personalized recipe recommendation using data mining techniques, such as association rule analysis and multi-class classification. Their system considers recipes and ingredients belonging to multiple categories, employing algorithms like Apriori and FP-Growth to uncover meaningful patterns.
(2) *Restaurant Recommendation System Based on Novel Approach Using K-Means and Naïve Bayesnt Recommendation System Based on Novel Approach Using K-Means and Naïve Bayes*: Joshi (2019) [4] presents and approach for restaurant recommendation system that combines K-Means clustering and Naïve Bayes classification to deliver precise suggestions. This

system analyzes data from Zomato a Indian delivery plication, using reviews information to make the recommendations.

(3) *RecipeRec: A Heterogeneous Graph Learning Model for Recipe Recommendation* : Tian et al.[7] introduced RecipeRec, a heterogeneous graph learning model designed for recipe recommendation. This model constructs a graph connecting users, recipes, and ingredients, capturing both recipe content and the relationships among these elements. Experiments demonstrate that RecipeRec outperforms previous methods in recommendation accuracy like classical collaborative filltering schemes FunkSvd and ALS.

Combining the idea of using the reviews and the content of the recipes from these sources, we came up with a model that mixes these two areas. However there is not previous study or results in the dataset that we are going to use.

3 Dataset

The dataset employed in this study is sourced from Food.com, containing extensive information categorized into two groups: recipe data and reviews data.

The recipe dataset includes 522, 517 recipes across 312 distinct categories, this first dataset contains information about each recipe such as the author, instructions, cooking times, nutritional information, ingredients, images and more.

The reviews dataset consists of 1, 401, 982 reviews authored by 271, 907 unique users, containing information for each review about the author, rating, recipe that is being reviewed, review data and timestamps, the latter was not used.

There are several noteworthy characteristics about this dataset. First, of the 522, 517 recipes, 52% have at least one review. Additionally, users tend to be generous about the rating given, with an average of 4.4 out of 5. The distribution of ratings is illustrated in Figure 1. Second, not all recipes include images, in fact, only 30% of all recipes have images associated.

When preprocessing the dataset for model evaluation, we consider only users having more than 3 reviews. For images embeddings, we included only recipes that have at least one image, then one random image was selected for being part of the embedding.

Finally, during the train-test split of the reviews dataset, 38, 436 reviews were excluded, primarily from the test set. This was due to the absence of the corresponding users in the training data.

4 Methodology

4.1 Overview

The methodology used in this study aims to evaluate how a personalized recommender system behaves based on the amount of features integrated from the dataset, integrating collaborative filtering and content-based approaches. The system leverages recipe attributes such as images, instructions and nutritional information, user interactions and deep learning techniques to provide recommendations.

The proposed model is built on a Neural Collaborative Filtering (NCF), which incorporates heterogeneous recipe features as mentioned above. Since this data is presented in various formats,ensuring robustness and relevance is essential. This is achieved through data

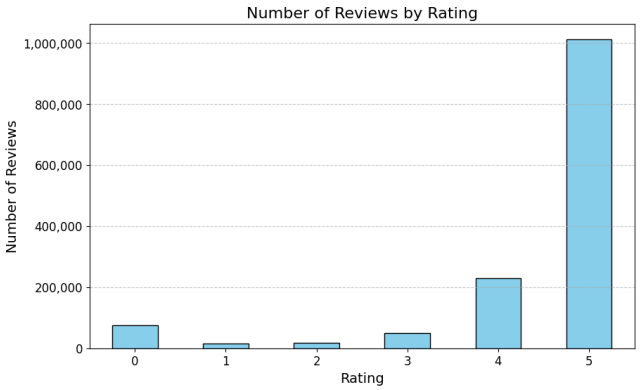

Figure 1: Distribution of Reviews by Rating, ranging from 0 to 5. The vertical axis represents the number of reviews, while the horizontal axis denotes the rating levels. The chart shows a significant concentration at 5 stars reviews.

preparation, which involves transforming raw data into feature vectors using pre-trained models such as ResNet50 [2] for image embeddings and BERT [1] for text encoding.

With these features, three different models were developed, representing three variations of NFC which incremental features:

- **NFC Basic:** Using only user-review interactions
- **NCF Nutrients:** Added nutritional information as features to generate the recommendation
- **NCF Full:** Built by combining all features, user-review interactions, nutritional information, image embeddings and instructions encoding.

We evaluated the models using standard metrics such as Precision@10, Recall@10 and NDCG@10, comparing the performance of our NCF models against baselines approaches. Furthermore, novelty and diversity were incorporated to ensure the recommendations provided by the NCF models were unique and varied.

4.2 Models architecture

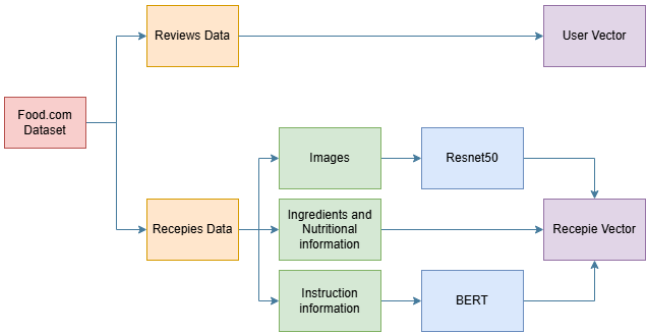

Figure 2: All features transformations in his respective embeddings for model NCF Full. NCF Basic doesn't use any of added features to recipe vector and NCF Nutrients only the nutritional information

First, embeddings for matrix factorization (MF) were created for each user and recipe. Then, for the multilayer perceptron (MLP) embeddings, a base embedding was created for each user and recipe. To incorporate the content data into the recipe vector, it was concatenated with:

(1) ResNet50 representation of the images: Not every recipe has an image. Approximately 160,000 items include one.
(2) Ingredients and nutritional information: Numerical and categorical features.
(3) BERT embedding representation of the concatenation of the recipe instructions.

NCF Basic doesn't use content data, NCF nutrients use (2) and NCF Full use (1), (2) and (3). The NCF Full had approximately 140 million parameters.

4.3 Training process

We split the data in training (75%) and validation (25%) resulting in almost 1 million reviews for training and 300k reviews for validation. In the validation set, only users that have at least 3 reviews got split. Then we train each model for 10 epochs with batch size of 64 and using the Adam optimizer with a learning rate of 0.001. Every 2k batches we register the validation data. The complete training took 8 hours for each model. For the NCF Full model we use a weight decay of 10^{-5} and added a dropout in each layer of the MLP with probability of 0.3 after a batch normalization. The loss function we use is the Mean Square Error of the rating.

The training was made in a local machine with a NVIDIA 3060 12GB GPU, with 32 GB DDR4 RAM. We made one jupyter for each model and in a separated jupyter we loaded the weights to made the inference. The inference (return the top 10 list) took almost 1 hour for each model, this means, returning the top 10 list takes an approximate of 1 minute for a given user.

4.4 Baselines

To evaluate the effectiveness of the proposed model, several models were trained and evaluated with the same dataset, these models include some baselines methods and state-of-art recommendation models. This helps to provide a reference point for assessing the performance of our model and understanding its limitations. The following baselines were implemented:

Normal Random: A non consistent model that, for each user, takes 10 random recipes. This approach does not consider any features and serves as a control, representing the worst-case scenario, Any model scoring lower than this indicates poor implementation.

Most Popular: A simplistic model that ranks recipes based on overall popularity, measured by number of interactions. Every user has the same recommendation list, this means that it does not consider individual preferences. This model is a fundamental benchmark for comparison.

Most Popular(+): An enhanced version of the Most Popular model with an added condition that dictates whether a recipe has a rating lower than 4, then it is excluded from the recommendation list. This modification ensures recommendations are both popular and highly rated.

FunkSVD: A matrix factorization [8] technique that approximates the user-item interaction matrix by learning latent features for users and items. It is a popular choice for collaborative filtering tasks. It captures patterns in user preferences and items attributes. However, it does not consider additional features such as image embeddings nor instructions, making it a useful baseline for evaluating feature integration.

LightGCN: A graph-based collaborative filtering method [3] that leverages user-item interactions in a lightweight graph structure. Captures efficiently relationships in a recommendation context, making it a strong baseline and competitor for systems with collaborative signals. However, it has a high memory requirement, for this reason, it was needed batch processing during metric calculations.

MultiVAE: A multi variational auto encoder [5] created specifically for collaborative filtering. It employs probabilistic techniques to model user-item interactions. This model is particularly effective in capturing latent user preferences. This model have a high use of memory, for this reason, and hardware restrictions, this model could only be trained by 20.000 users, less than 10% of the dataset's total user population.

5 Results

The following metrics are those we used to compare how good the predictions of each model were:

$$\text{Precision@K} = \frac{\sum_{i=1}^{n} Rel(i)}{n}$$

Precision@K measures the proportion of relevant items among the top K recommended items. $Rel(i)$: Indicator function (1 if item i is relevant, 0 otherwise), n: Number of items in the top K.

$$\text{Recall@K} = \frac{\sum_{i=1}^{n} Rel(i)}{\#Rel_u}$$

Recall@K measures the proportion of relevant items retrieved out of all relevant items for a user,$Rel(i)$: Indicator function (1 if item i is relevant, 0 otherwise), $\#Rel_u$: Total number of relevant items for user u.

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}$$

Normalized Discounted Cumulative Gain (NDCG@K) evaluates the ranking quality of recommendations by comparing the ranking of relevant items to an ideal ranking.DCG@K: Discounted Cumulative Gain at rank K,IDCG@K: Ideal Discounted Cumulative Gain at rank K.

$$\text{Novelty} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{L} \sum_{i=1}^{L} \log\left(\frac{1}{pop_i}\right)$$

Novelty measures how "novel" or "unpopular" the recommendations are by penalizing frequently recommended items. $|U|$: Total number of users, L: Length of the recommendation list for each user, pop_i: Popularity of item i, often calculated as the number of users who interacted with it.

$$\text{Diversity} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{L(L-1)} \sum_{i=1}^{L} \sum_{j=i+1}^{L} (1 - \text{sim}(i, j))$$

Diversity measures the dissimilarity between recommended items, encouraging a more varied recommendation list.$|U|$: Total number of users, L: Length of the recommendation list for each user,$\text{sim}(i, j)$: Similarity between items i and j.

These formulas are commonly used in recommendation system evaluations to measure precision, recall, ranking quality, novelty, and diversity in the recommendations. Each metric focuses on a specific aspect of recommendation quality, ensuring a comprehensive evaluation. The following tables contains the results for each models in the metrics exposed:

	Precision@10	Recall@10	NDCG@10
NormalRandom	0.00022	0.0006	0.0003
Most popular	**0.005**	0.016	0.011
Most popular (+)	**0.005**	0.017	**0.012**
FunkSVD	0.0007	0.0021	0.0016
LightGCN	0.0029	0.017	0.010
MultVAE* (20k)	0.003	**0.02**	**0.012**
NCF Basic	0.002	0.0008	0.002
NCF Nutrients	0.0018	0.0007	0.002
NCF Full	0.0027	0.0015	0.0022

Table 1: Comparative results in Precision@10, Recall@10 and NDCG@10 metrics

	Novelty	Diversity
NCF Basic	117	3.47
NCF Nutrients	118	3.47
NCF Full	110	3.56

Table 2: Results of Novelty y Diversity for NCF models

6 Conclusions and future work

Non of our models could surpass the baselines. MultVae-20k and Most popular had the best results and the difference in performance between all NCF models indicates the content does not contribute much if any at the performance. With almost half of the images not existing and a huge bias in the distribution of ratings this task was not easy. For time and computer power limitations we couldn't make a complete ablation, that is an important aspect that could be explored in future work. Due to the amount of time it takes to infer a single user top 10 list this recommender system isn't optimal for deploying in production. Also, a model ensemble between NCF Full, MultVae-20k and Most popular where MultVae gives a list and the other two methods reorder it to enhance the nDCG metric. Finally we made a qualitative analysis on the recommendations for specific users and an ethics analysis of our work in the apendix section.

Acknowledgments

To Robert, for the bagels and explaining CMYK and color spaces.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: pre-training of deep bidirectional transformers for language understanding. (2019). https://arxiv.org/abs/1810.04805 arXiv: 1810.04805 [cs.CL].

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. (2015). https://arxiv.org/abs/1512.03385 arXiv: 1512.03385 [cs.CV].

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. Lightgcn: simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '20). Association for Computing Machinery, Virtual Event, China, 639–648. ISBN: 9781450380164. DOI: 10.1145/3397271.3401063.

[4] 2020. *Restaurant recommendation system based on novel approach using k-means and naïve bayes classifiers*. (Apr. 2020), 609–620. ISBN: 978-981-15-2187-4. DOI: 10.1007/978-981-15-2188-1_48.

[5] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. (2018). https://arxiv.org/abs/1802.05814 arXiv: 1802.05814 [stat.ML].

[6] Harish Neelam and Koushik Veerella. 2024. Enhancing personalized recipe recommendation through multi-class classification. *International Journal of Computer Science Engineering and Information Technology*, 14, (Aug. 2024), 15–25. DOI: 10.5121/ijcseit.2024.14502.

[7] Yijun Tian, Chuxu Zhang, Zhichun Guo, Chao Huang, Ronald Metoyer, and Nitesh V. Chawla. 2022. Reciperec: a heterogeneous graph learning model for recipe recommendation. (2022). https://arxiv.org/abs/2205.14005 arXiv: 2205.14005 [cs.IR].

[8] Yuefeng Zhang. 2022. An introduction to matrix factorization and factorization machines in recommendation system, and beyond. (2022). https://arxiv.org/abs/2203.11026 arXiv: 2203.11026 [cs.IR].

7 Appendix

We realize a qualitative study on the recommendations of the NCF Full model in three cases. The first if for a user who (in training) most of the recipes reviewed were healthy. The second case of study is for a user that reviewed mostly vegetarian food.

A Healthy user

We choose the user 925 that in the training set reviewed:

- Healthy | Orange Soda Ice Cream
- Not-Healthy | Very Banana Chocolate Chip Muffins
- Not-Healthy | Best Darn Sugar Cookies Ever
- Healthy | Lemonade
- Healthy | Caramel Brownies
- Healthy | Fresh Peach Salsa
- Healthy | Country White Bread or Dinner Rolls
- Healthy | Pumpkin - Gingerbread Loaf
- Healthy | Fig Preserves
- Healthy | Depression Fudge Cake
- Not-Healthy | Banana Cake

The classification of Healthy or Not-Healthy is based on the keywords of the recipe if the keywords contains the category of Healthy respectively. This classification isn't accurate because this are tags given by the recipe creator. We can see this in the Depression Fudge cake, this recipe is classified as Healthy but the ingredients are

- sugar
- buttermilk
- flour
- baking soda
- salt
- vanilla
- butter
- powdered sugar

- hot water

No combination of those ingredients will yield a healthy recipe. Continuing the analysis, the test recipes which are the ground truth are:

- Healhy | Chinese Imperial Palace General Tso's Chicken
- Not-Healhy | Turkey Barley Soup
- Not-Healhy | Laura's Potatoes
- Not-Healhy | Breadstick Broomsticks
- Semi-Healhy | Balsamic Asparagus

The Semi-Healthy classification is given based on healthy-related keywords like *Low Calories* or *Low Cholesterol*. The ground truth is very different of what anyone could think user 925 would choose. The keywords that both train and test sets share mostly are *Easy* and *Under X min* (which we will call *Fast*). With this information a good, handcrafted recommendation based on training would be one that have keywords which indicates the recipe is easy, fast and healthy.

(1) Semi-Healthy, Fast | Summer Vegetable Grill Packets
(2) Fast | Back Bayou shrip and corn
(3) Fast, Easy | Olive Garden Zuppa Toscana Soup
(4) Healthy, Fast | Famous Challah
(5) Fast | Banana Banana Nut Bread
(6) Fast | Scott Hibb's Amazing Whiskey Grilled Baby Back Ribs
(7) Fast | Sangria
(8) Fast, Easy | Broiled Parmesan Tilapia
(9) Fast, Easy | Seared Ahi Tuna Glazed With wasabi butter
(10) Healthy, Fast | Wait family Pizza Dough

The keywords seems to match with what we expected but the theme doesn't. The training set is mostly pastry recipes while the recommendation theme is dinner recipes, however the test set is also about dinner, this could be an indicator that the model is trying to balance the type of foods (Breakfast, Salads, Dinner, Beverages and Dessert) but we can't confirm that with this data.

B Vegetarian User

In this case we focus on the user 60437. The training reviews keywords are Vegan, Vegetarian, Easy and Fast. The keywords Vegan and Easy consider that the Keyword is explicitly in the dataset, on the other hand, Vegetarian means that the recipe does not contain explicitly the Vegan tag but it does not contain meat in the ingredients (this does not includes crustaceans and fishes). The Fast tag as we describe in the last section means that the recipe contain keywords with styles like *Under X time*. The train set for this user is:

- Vegan, Easy, Fast | Algerian Broad Beans And Garlic
- Vegetarian, Fast | Lima Beans and mushrooms
- Vegetarian, Easy, Fast | simple bran muffins
- Vegetarian, Easy, Fast | Jeniffer's Herb Dumplings
- Vegan, Easy, Fast | Only One Vegan Banana Bread
- Vegetarian, Easy, Fast | Mouthwatering Shrimp Manier
- Vegetarian, Easy, Fast | Spicy Tilapia With Pineaple, Jalapeno pepper relish
- Vegan, Easy, Fast | Roasted Green beans
- Vegan, Easy, Fast | Easy microwave szechuan green beans
- Vegan, Easy, Fast | Lime-herb dressing
- Vegan, Easy, Fast | An Avocado-licious sandwitch
- Easy, Fast | Sommer's Stroganoff
- Vegan, Easy, Fast | Lentil Stew with spinach and potatoes
- Vegetarian, Fast | Portabella mushroom burgers

Just one recipe contains meat in the training set and is the same in the ground truth:

- Vegetarian, Fast | Honey Cornmeal Bread
- Vegetarian, Easy | Western Noodles
- Vegan, Easy, Fast | Coconut Rice
- Easy, Fast | Black Bean Taco Salad

Since almost no meat is present in the ingredients, the recommender system should return mostly vegetarian food, but as we can see:

(1) Fast, Easy | Simply Irresistible tropical potato salad
(2) Fast | Mexican Stack up
(3) Fast, Easy | breakfast shepherd's Pie
(4) Fast | summer vegetable grill packets
(5) Fast | bacon lattice tomato muffins
(6) Fast | company's coming mashed potatoes
(7) Fast | back porch bayou shrimp and corn
(8) Fast, Easy | french quarter sweet potato cake
(9) Fast, Easy | olive garden zuppa toscana soup
(10) Fast, Easy | mile high cabbage pie

Every recipe contains meat. This definitely means that our model did not learn how to manage the content given.

C Ethics

Our work have a direct impact on the users health. Our results are far from optimal but in the ideal case, a food recipe recommender has two opposing paths. One of them is to return new recipes that aligns with the user history and the other is trying to push the user to have a more healthy diet. The first one would lead to a better RecSys for a business, because they need to engage the audience and the second, would lead to more ethic way of doing business. Our work just focus on the first path, but due to the way of classification of recipes where the authors just set the keywords they think is appropriate and have no way to measure the *healthiness* of the recipes it is impossible to take the second path.

D Category specific recommendations

with our model, we want to recommend a specific category of recipes if its ask by the user. It was a completely naive approach where instead of making a score for each recipe and then sorting we just skip every recipe that doesn't have the category in question. For the simplicity of the approach we made no further analysis on this results.