Food for Thought: A Recipe Recommender System

Joseph Gargano
Student,
George Mason University
4400 University Drive
Fairfax, VA 22930
jgargan@gmu.edu

Krupanka Vijay
Student,
George Mason University
4400 University Drive
Fairfax, VA 22930
kvijay@masonlive.gmu.edu

Surya Srirangarajan
Student,
George Mason University
4400 University Drive
Fairfax, VA 22930
ssrirang@masonlive.gmu.edu

Lydia Cooley
Student,
George Mason University
4400 University Drive
Fairfax, VA 22930
lcooley@gmu.edu

ABSTRACT

This paper proposes a method to provide recipe recommendations under certain budgetary constraints. This study explores popular recommender system approaches including collaborative-based filtering, content-based filtering, and association rule algorithms to develop a method to find the top new recipes that a user will like. Once the model generates recipe recommendations, the information is passed off to a constraint optimizer to find the optimal combination of recipes that stay within the user's provided budget. This study utilizes an open source dataset based on recipes and user ratings from Food.com and web-scraped ingredient prices to test and create the recipe recommender system that fits the budget constraint.

Categories and Subject Descriptors

CCS Concepts: recipes, food prices

Keywords

Recipe, Recommender Systems, Food Prices, Collaborative Filter, Content Filter, Aprori, Constraint Problem

1. INTRODUCTION

Recipe Recommender systems solve real-world everyday problems with meal planning and budgeting. According to the United States Department of Agriculture, United States food prices rose .9% from 2018 to 2019 with the fresh vegetable category increasing the most at 3.8% [16]. Food-away-from-home spending outpaced food-at-home spending accounting for 54.8% of U.S. food spending in 2019. In 2020, the COVID-19 pandemic shifted the spending trend away from restaurants and toward local supermarkets. In March 2020 monthly revenue in food retail jumped up by 25%, returning to relative levels not seen since 1995 in America [1]. In April 2020 food retail remained 10% higher than pre-pandemic levels. Shoppers exceeded their usual weekly household grocery budgets by 33% on average, bringing the average household grocery expenditures from \$120 to \$161 between March 21 and April 2, 2020 [15].

These trends indicate Americans pre-pandemic had relied more on restaurants to prepare meals, and therefore may not be experienced in meal planning and preparation. Many Americans' budgets are also strained, and they may require assistance providing meals within budgetary constraints.

This increased spending on food retail takes place amid overall economic uncertainty exacerbated by the COVID-19 pandemic. Pew Research Center [12] found that 53% of lower-income adults could not pay all of their bills on time, up from 44% pre-pandemic; one-in-four adults have had trouble paying their bills since the COVID-19 outbreak started; 17% have gotten food from a food bank. The Feeding America organization has reported that among children, the projected food insecurity rates for 2020 range from 15.0% (North Dakota) to 32.3% (Louisiana and Nevada). California would see both the largest absolute increase in the number of children living in food-insecure households (+864,100) as well as the largest total number of children living in food insecure households (2.2 million) [6]..

The goal of this study is to explore popular recommender systems and formulate a design for a recipe recommender system that takes into consideration a person's budgetary constraints. With this approach, people can cook recipes that they would enjoy while staying within their provisioned budget.

The first part of this paper reviews popular recommender systems that help solve similar recommendation problems. These systems include collaborative-based filtering, content-based filtering, and association rule algorithms. This review also inspects previous works and methodologies to solve the budgetary constraint problem. The following section explores and discusses key findings from the open source Food.com dataset that includes recipe information and user reviews. Next, the paper walks through the proposed methodology for the system to solve the recommendation and constraint problems. The remaining of the paper examines the results from the various algorithms and discusses future enhancements and ideas.

Related Work

Recommender Algorithms

Recommender Systems are developed in parallel with the web and intend to collect information on the explicit (i.e. ratings) or implicit (i.e. behaviors such as making purchases) preferences of users to recommend items . Bobadilla et. al's 2013 study of 300 papers in the field of recommender systems found that the most used algorithms used collaborative filtering [3]. Collaborative filtering is based on the human tendency to base our own decisions on the decisions of others. Recommender systems tend to layer content-based, social, or knowledge-based filters on top of collaborative filtering.

Melville et. al [12] acknowledge most recommender systems use Collaborative Filtering (CF) or Content-based (CB) methods. Their approach combines both methods by applying a content-based predictor to existing user data, and then providing suggestions using collaborative filtering. As Melville et. all demonstrate in their study of movie ratings, CF systems have been successful, but exhibit two fundamental drawbacks: 1) sparsity; most users do not rate most items, and 2) first-rater problem; an item cannot be recommended unless a user has rated it. Both of these problems inform this study since sparsity is likely and newer recipes will not have been rated. Further, Melville et. all in their CB predictor use a bag-of-words Bayesian text classifier to profile movies and predict the rating of unrated movies. The data sets in this study also include text fields for the raw user ratings. The user data in this study is also similar in that it includes ratings in the user data. Applying CF to weight users with respect to similarity to the active user could also be useful to predict ratings.

The most common domain at the time of this study was movie recommender systems, but the literature studied also contained recommender systems for books, music, television, documents, and web searches.

The first reference to a recipe recommender system this study reviewed was in the 2011 proceedings of a workshop on context-aware search and retrieval [18]. This recommender system was ahead of it's time as it used data collected from, "Accelerometers embedded into these utensils," a smart fork not yet connected to the internet well before " internet of things" was a buzzword. Bobadilla et. al in 2013 predicted that recommender systems would be influenced further by the internet of things. The study doesn't indicate specifically how, but it does seem future datasets could be provided by smart kitchens directly to collect information on the food people prepare and eat at home.

Yajimi and Kobayashi [20] score recipes based on how "easy" they are. This paper is informative because of the brilliance in its simplicity. Each recipe is scored by adding an ingredient score, seasoning score, and cooking method score. The recommendation is based only on a layered query. This paper does reference a related linear optimization model but the work cited is provided only in Japanese.

Ueda et. al [12] use the dispersion of ingredients in each recipe to identify recipes to meet a user's preference based on their browsing history and past recipe use. The paper is flawed because liked or disliked ingredient quantity is not the only factor impacting someone's food preferences (some people like any food if it's fried crispy.) However this paper is still useful since it takes not only ingredient lists but quantities into consideration when considering preferences, pointing out that 100 grams of black pepper in a dish is quite different from 100 grams of potato. This is something to consider when looking at the pricing of the ingredients, perhaps when looking at the density of expensive ingredients to consider the cost per serving.

Majumder et. all [4] combined natural language processing and recommender systems into a learning system system with personalized text generation as output. This model generates new recipe instructions based on a few user inputs and the user's prior interaction with recipes. This paper informs the approach here since the recipe name or ingredients are represented as a series of tokens.

A tension exists in recommender systems between accuracy and performance. For k-nearest neighbor models, for example, most use cases require speed and accuracy. The algorithm needs to work quickly to keep users engaged online, and must also provide good results to keep users interested in returning. Sarwar et. all address this challenge by offering an item-based k-nearest neighbors algorithm instead of a like-minded-user-based model. This algorithm uses lower volumes of data that is more static in nature, since relationships among items change less often. Using this data makes the model faster. Sarwar et. all found that the item-based models provide better quality than user-based models at all sparsity levels. They also tested regression-based models, finding these perform well with sparse data but the quality decreases with more data, perhaps indicating over-fitting. These experiments inform this paper as the data sets include both user ratings and items. Sarwar et. al's results indicate items-based recommender systems may provide better quality results than like-minded user based systems [17].

Other examples of possible recommender systems include ruled-based algorithms, also known as association algorithms. These algorithms aim to understand the behavior of users and attempt to translate common behaviors (such as recipes that are commonly reviewed together or items frequently bought together) into rules based on their associations. Borgelt and Kruse [5] explain one of the most popular rule-based algorithms known as Apriori. Apriori uses the notation that no superset of an infrequent itemset can be frequent.. To determine frequent itemsets, the algorithm first finds sets of items that meet the minimum support and then generates rules based on the confidence that these items will appear together. Support, confidence, and lift are key terms for the Apriori algorithm. Support is the percentage of an itemset's appearance to the overall itemset. Confidence is the conditional probability that item X will appear given item Y. Lift is the opposite of confidence so that the conditional probability that item Y will appear given item X. While this algorithm helps find complex rule sets, it does suffer from speed as it is quite computationally expensive to navigate a tree of possible itemsets. Nevertheless, people use Apriori to help find recommendations for its users. Bandyopadhyay, Thakur, and Mandal (2019) used an Apriori algorithm to calculate product recommendations for users on e-commerce websites. Their implementation of the algorithm resulted in recommendations based on the rule sets that were 60% correct. This shows that an Apriori algorithm could be successful in recommending recipes to users based on discovered associations.

Constraint Problem

Beside finding the best recommended recipes for a user, this study aims to find recipes that can meet a user's given budgetary constraints.

Sarkar [14] describes using linear and mixed integer programming techniques to solve discrete optimization problems. By setting a minimum or maximum integer for some parameters, the algorithm can minimize or maximize a result given specific constraints. It's possible to use this method with the budget aspect of this study's model, i.e. minimize cost given a specific set of nutritional or caloric constraints.

Kumar [9] explains how constraint satisfaction problems take a set of variables and constraints to find an optimal solution that satisfies all constraints. Within the research, Kumar explains different methodologies to use when solving constraint satisfaction problems including the generate-and-test method and backtracking. The backtracking method first attempts to find a valid combination but upon any constraint violation backtracks to the next logic value that does not violate that constraint. While backtracking is more efficient than the generate-and-test method, Kumar notes that it can still be exponentially computational due to thrashing, or the searching in the different areas or spaces to find the valid combination.

Sabin, Freuder and Wallace [7] expand on the constraint satisfaction problem. In their research, the authors used a modified backtrack search algorithm, a forward checking algorithm, and an arc-consistency algorithm to validate constraints and to check validity of future variables. By combining the algorithms to check past and future variables while maintaining consistency in the search tree, the authors increased the efficiency in solving a constraint satisfaction problem versus the traditional backtracking or generate-and-test approaches.

2. Datasets

This section reviews the tools, data sets, data exploration steps, and overall approach to the analysis.

Tools

This architecture is repeatable and low-cost. The tools are open-source and have student and trial versions available. Tools include python, Kaggle notebooks, and google file shares.

Key Datasets

The solution requires recipe, user, and pricing datasets.

The food.com open source dataset on Kaggle [8], contains information on 230,186 recipes collected from Food.com. The recipe dataset contains information about each recipe including name, id, minutes to cook, contributor id, submitted data, descriptive tags, nutrition information, number of steps, specific steps, recipe description, ingredients, and the number of ingredients. The dataset also contains a tokenized version of this information to help speed up processing and complex queries. This study will use a combination of both datasets to process and visualize the results.

The dataset also contains information on 1,132,367 user reviews from 226,570 different users. This dataset contains information on each user including their unique id, techniques of recipes that user interacted with, the recipes that that the user interacted with, the total count of recipe interactions per user, the ratings each user gave to each recipe that he or she interacted with, and the text review of that particular rating for each recipe.

The dataset contains three sets of data collected over 18 years on Food.com

The first set is the Interaction dataset which is then split into train, test and validate datasets.

Table 1. The Interaction dataset is split into train, test, and validate datasets.

Attribute name	Type of value	Description
User_id	Number	User ID
Recipe_id	Number	Recipe ID
Date	Date	Date of interaction
Rating	Number	Rating given
U	Number	User ID, mapped to contiguous integers from 0
I	Number	Recipe ID, mapped to contiguous integers from 0

Table 2. The RAW_recipes.csv and RAW_interactions.csv files contain recipe data from users of food.com

Attribute name	Type of value	Description
Name	String	Recipe name
Id	Number	Recipe ID
Minutes	Number	Number of minutes to prepare recipe
Contributor_id	Number	User ID who submitted this recipe
Submitted	Date	Date recipe was submitted
Tags	String	Tags for recipes on Food.com
Nutrition	String	Nutrition information (calories (#), total fat (PDV), sugar (PDV), sodium (PDV), protein (PDV), saturated fat)
N_steps	Number	Number of steps in recipe
Steps	String	Step by step text for recipe
Description	String	User provided description
ingredients	String	List of ingredient names
N_ingredients	Number	Number of ingredients

Table 3. The PP_recipes.csv and PP_users.csv files contain pre-processed data where recipe text metadata is tokenized via the GPT subword tokenizer.

Attribute name	Type of value	Description
Id	Number	Recipe ID
#i	Number	Recipe ID mapped to contiguous integers from 0
Name_tokens	Number	BPE-tokenized recipe name
Ingredient_tokens	Number	BPE-tokenized ingredients list (list of lists)
Steps_taken	Number	BPE-tokenized steps
Techniques	Number	List of techniques used in recipe
Calorie_level	Number	Calorie level in ascending order
Ingredient_ids	Number	IDs of ingredients in recipe

Table 4. PP_users.csv

Attribute name	Type of value	Description
#u	Number	User ID mapped to contiguous integer sequence from 0
techniques	Number	Cooking techniques encountered by user
Items	Number	Recipes interacted with, in order

#n_items	Number	Number of recipes reviewed
ratings	Number	Ratings given to each recipe encountered by this user
n_ratings	Number	Number of ratings in total

The last piece of the dataset is the pricing information for the ingredients in each recipe. There was not an open source dataset that readily contained this information; however, this study developed various web scrapers to gather price information from a popular grocery store website for the ingredients in each recipe and then appended the total recipe price to each recipe. The total number of unique ingredients per recipe name was 14,890.

Data Exploration

Before delving into the problem set, this study explored the dataset to find additional insights. The below figures and their descriptions represent important facets and key findings from the dataset.

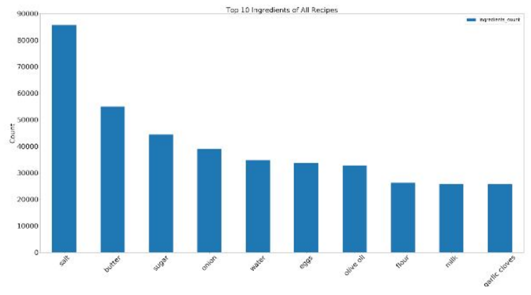


Figure 1 shows the top 10 ingredients of all recipes in the dataset. Salt is the most active ingredient of all the recipes. Other top ten ingredients include recipe building blocks like butter, sugar, onions.

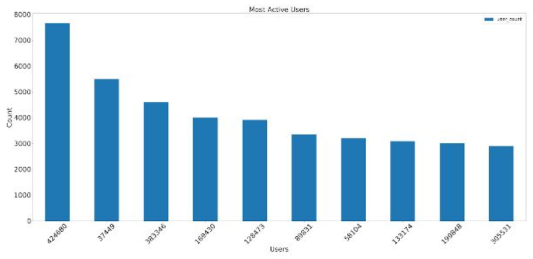


Figure 2 shows the top ten users and their counts of ratings. The top user rated over 7,500 recipes.

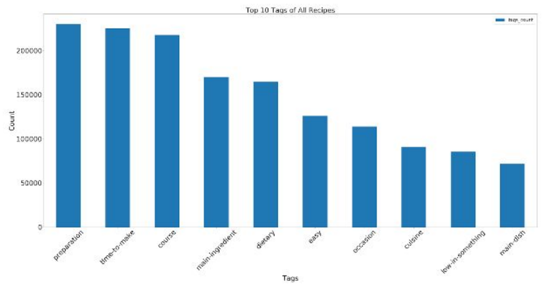


Figure 3 denotes the top 10 most common metadata tags found in the recipes. Users appear to be greatly concerned with how long or how little a recipe takes to cook as the top two tags deal with time factors like "preparation" and "time-to-make". Other tags in the top 20 include additional time elements like "30-minutes-or-less" and "60-minutes-or-less". In addition, users also appear to care about how difficult a recipe is to make based on common metadata tags like "easy" and "main-ingredient". Health is another important metadata category as "dietary" and "low-in-something" both appear in the top 10 tags

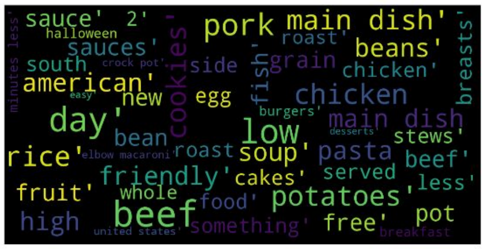
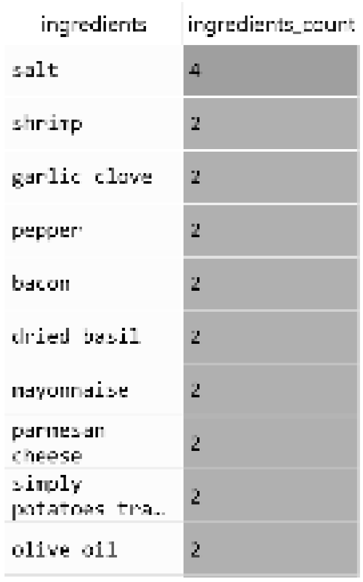
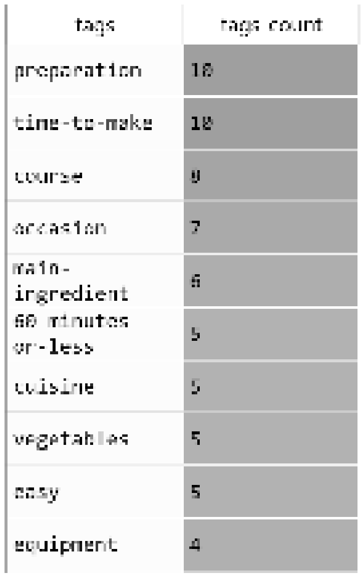


Figure 4 represents a random selection of tags used on the recipes presented as a word cloud visual. Recipes in the dataset have a variety of tags and meanings including the various styles a recipe may be in, the contents of the recipe, or other descriptive metadata tags.



ingredients	ingredients_count
salt	4
shrimp	2
garlic clove	2
pepper	2
bacon	2
dried basil	2
mayonnaise	2
parmesan cheese	2
simply potatoes tra...	2
olive oil	2

Figure 5 ranks the most common ingredients in the top 10 rated recipes. Comparing Figure 5 to Figure 1, higher rated recipes include flavorful ingredients like salt, shrimp, garlic, and pepper. All other ingredients were only shared with one other recipe in the top 10.



tags	tags count
preparation	10
time-to-make	10
course	9
occasion	7
main-ingredient	6
60 minutes or-less	5
cuisine	5
vegetables	5
easy	5
equipment	4

Figure 6 shows that all the top 10 recipes share the "preparation" and "time-to-make" tag, highlighting the temporal aspects are important to users rating recipes.

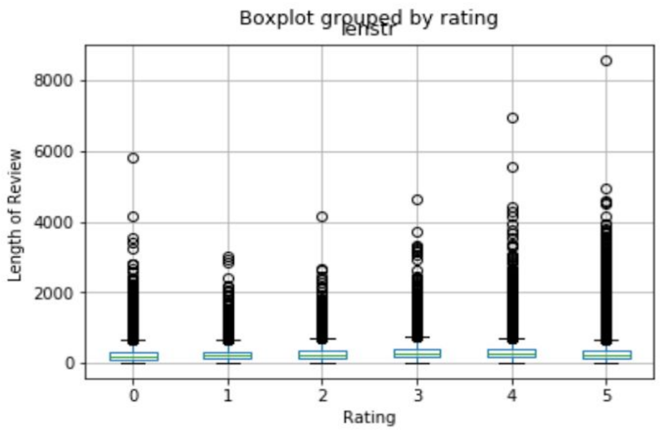


Figure 7 shows that the reviews that are lengthy have higher rate. The boxplot for 5 point rating has the highest number of characters in its length.

3. Methodology

The approach to this challenge includes three major parts: calculating the total cost of a recipe, the recommender algorithm to find recipes a user may like, and an optimizer to meet specific budget and user preference constraints (Figure 8). This study utilized Python to code the various models and calculate the recommendations for the target user. A web scraper pulled pricing information for each ingredient and then calculated the total sum for each recipe. The recipe total influences our budget constraints. This study also used various recommendation algorithms to attempt to find the most accurate user recommendations including collaborative filter user-based, collaborative filter item-based, an ingredient content-based, and an association rule based algorithm.

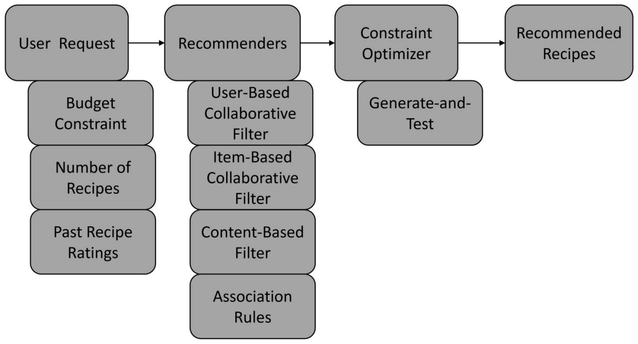


Figure 8 denotes the flow of the recipe recommender system starting with the user request, various recommenders, constraint optimizer, and ending with the recommended recipes

The goal of a user-based collaborative filter is to give an input of recipe ratings, find other users who also have rated those recipes similarly and recommend recipes that the similar user also enjoyed to the target user. This can be done utilizing matrix factorization and calculating similarity scores between users. Then the model can find similar users to the target user and estimate values for the recipes that are missing from the target user given the ratings of the similar user. The model used to conduct the matrix factorization in the study was a single value decomposition (SVD) model.

Similarly, to the user-based collaborative filter, this study attempted to utilize K-Nearest Neighbors (KNN) algorithm to calculate an item-based collaborative filter. This algorithm attempts to group similar items together based on user ratings and user history.

For the content-based recommender, this study examined the ingredients found in the recipes. This was done by vectorizing ingredient lists and creating a matrix of vectors representing recipes and the associated ingredients. Then the model calculates the cosine similarity between each vector, creating a cosine similarity matrix determining how similar a recipe is to the other recipes in the dataset. By utilizing the similarity matrix, the model would theoretically be able to recommend a recipe based on ingredients input into the model.

The last recommender algorithm used an Apriori algorithm to find rule based associations in the item set. This discovered patterns based on users who rated the same recipes and determined which recipes were likely to be rated together. These can then be fed as recommendations to other users who may have rated one of the recipes in the discovered ruleset.

The next step consists of building an optimizer to adjust the recommendations based on the budget constraint and other user preferences. The methodology for finding the recipes for given constraints fit the constraint satisfaction problem. The goal of constraint satisfaction problems is to find the best combinations of outputs that fit under a constraint to maximize or minimize certain variables; in this case, maximize the total number n of recommended recipes' ratings while staying under a certain budget so that:

n # of recipes = # of user's defined recipes

sum(recipe costs) <= user's defined budget

For this study, the algorithm first took the top 20 recipes recommended to a user and checked to see if the sum of those recipes were less than or equal to the user's defined budget. If the sorted list met the budget constraint, then the n number of recipes were returned to the user. If the first n number of recipes did not meet the budget constraint, then the algorithm attempts to find all possible combinations of the top 20 recipes that meet the constraint using the generate-and-test method. The next step of the algorithm sorts the valid combinations found in the generate-and-test method by sum of ratings and returns the highest rating combination output to the user.

4. RESULTS

Recipe Pricing

This study scraped pricing data for individual ingredients from Wegmans.com and collected 12,423 prices out of 14,890 total ingredients in our dataset. In order to maintain more recipe information in the dataset, this study used mean substitution with a mean of $4.89 for the missing ingredients.

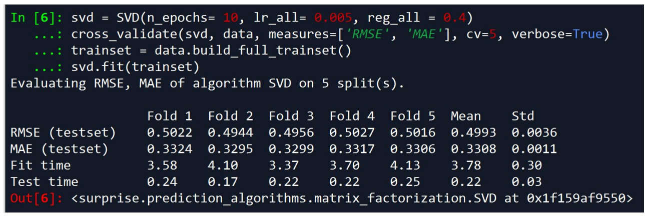
Collaborative Filtering Recommenders

Additional Data Cleaning

For the following collaborative based recommenders results, this study took extra data cleaning steps to help reduce the noise in the dataset, improve accuracy of results, and increase computational speeds. First, users who rated less than 100 recipes were removed from the dataset. Next, recipes which appear less than 10 times were also removed. This resulted in a dataset containing 142,545 user reviews with 7,552 unique recipes. Next, the model creates a matrix with the 142,545 users and 7,552 recipes. Despite the additional data cleaning, the matrix still resulted in a sparse matrix with a density of only 0.0054%. This means that the matrix contains mostly "n/a" values and that the remaining users did not commonly rate the same things.

User-Based Results

Nevertheless, matrix factorization occurred with a single value decomposition (SVD) model to predict ratings for users that had similar recipe ratings. In order to find the best hyperparameters for the model, this study used a grid search method, which determined 10 epochs, a 0.005 learning rate, and a regression all of 0.4. In addition, a cross validation with five folds produced a root mean square error (RMSE) of about 0.4997 (Figure 9). This indicates the model's predictions were only off by about 0.5.



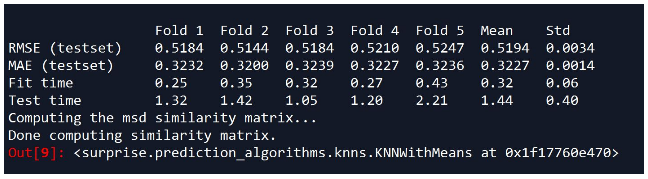
```
In [6]: svd = SVD(n_epochs= 10, lr_all= 0.005, reg_all = 0.4)
   ...: cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
   ...: trainset = data.build_full_trainset()
   ...: svd.fit(trainset)
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.5022  0.4944  0.4956  0.5027  0.5016  0.4993  0.0036
MAE (testset)     0.3324  0.3295  0.3299  0.3317  0.3306  0.3308  0.0011
Fit time          3.58    4.10    3.37    3.70    4.13    3.78    0.30
Test time         0.24    0.17    0.22    0.22    0.25    0.22    0.03
Out[6]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x1f159af9550>
```

Figure 9 shows the user-based collaborative filter results using the best hyperparameters and five-fold cross validation

Item-Based Results

The next recommender system this study tries is an item-based collaborative filter recommender. This model uses a K-Nearest Neighbors (KNN) method to find recipes that have similar features based on their user ratings. Another grid search algorithm took place to find the best hyperparameters for the KNN model while using the same matrix of 142,545 users and 7,552 recipes. The grid search found the best parameters while computing the mean squared difference (msd) similarity between all pairs of recipes and a minimum support of three, which means at least three users must rate the recipe for the similarity score to not be set to zero. With another cross validation with five folds, the KNN model produced a RMSE of 0.519, slightly worse than the user-based model (Figure 10).



```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.5184  0.5144  0.5184  0.5210  0.5247  0.5194  0.0034
MAE (testset)     0.3232  0.3200  0.3239  0.3227  0.3236  0.3227  0.0014
Fit time          0.25    0.35    0.32    0.27    0.43    0.32    0.06
Test time         1.32    1.42    1.05    1.20    2.21    1.44    0.40
Computing the msd similarity matrix...
Done computing similarity matrix.
Out[9]: <surprise.prediction_algorithms.knns.KNNWithMeans at 0x1f17760e470>
```

Figure 10 shows the item-based collaborative filter results using the best hyperparameters and five-fold cross validation

Content-Based Recommender

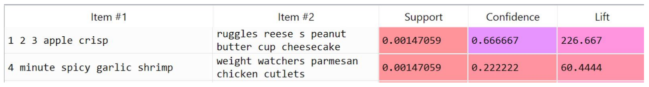
Ingredient-Based Results

Another methodology this study attempts is a content-based recommender utilizing the ingredients in each recipe. This method formed a term frequency-inverse document frequency (TFIDF), which evaluates the importance of each ingredient in a recipe. Then the algorithm calculates a cosine similarity score between each recipe based on the ingredient TFIDF. However, due to technology limitations, this study was unable to calculate similarity scores between each recipe based on ingredient lists. Because of the large number of recipes, each with multiple ingredients, the cosine similarity matrix exceeded the computing and storage capacity of the machines used in this study to run the model. Possible future research includes expanding technology capacity to continue study, implementation, and incorporation of this model.

Association Rules Recommender

Apriori Results

Attempting other recommender algorithms, this study used an Apriori approach to discover associations in the recipe set. Due to the sparsity of the matrix, the model used a low support of 0.001 and a confidence of 0.2. This resulted in 267 rule associations with two recipes and two rule associations with three recipes for a total of 269 rule associations (Figure 11).



Item #1	Item #2	Support	Confidence	Lift
1 2 3 apple crisp	ruggles reese s peanut butter cup cheesecake	0.00147059	0.666667	226.667
4 minute spicy garlic shrimp	weight watchers parmesan chicken cutlets	0.00147059	0.222222	60.4444

Figure 11 displays a sample of the rules calculated from the Apriori result.

Constraint Optimizer Results

The last portion of this study looks at solving a constraint problem that aims to maximize the total sum of recipe ratings while making sure the total sum of the recipes stay under the user's budget. This study took the top 20 recommended recipes for a user and with the generate-and-test method found all possible combinations that met the constrained budget. Next, the algorithm sorted the total sums of the ratings to find the optimal combination that provides the best rated recipes under the budget.

5. CONCLUSION

This study explores possible approaches to create a recommender system that meets a user's budgetary constraint. The best algorithm was the user-based collaborative filter that was able to accurately predict a user's recipe rating within 0.5. By passing the output of recommended recipes of a particular user to the constraint optimizer, the next position of the model, solved the remaining problem and provided a user with the optimal set of recipes that met their given budget. This system shows it is possible to solve the recipe recommendation budgetary constraint problem utilizing a variety of methodologies and algorithms.

6. FUTURE RESEARCH

With expanded technology capacity, future research could include calculating and combining an ingredient-based recommender to existing models. Other future works include exploring a recommender system based on the metadata recipe tags attached to each recipe. In addition, future research will study the impact in combining the various recommender systems into a hybrid recommender based approach to enhance recommendations. Furthermore, expanding the constraint optimizer by using better backtracking and graph based methods would improve the time it takes to find the optimal recipe recommendations; however, due to time constraints for this project, this study was only able to use the generate-and-test method.

ACKNOWLEDGMENTS

Python Software Foundation. Python Language Reference, version 3.7. Available at http://www.python.org

REFERENCES

[1] Association, T. F. (2020, 10 6). "Food Retailing Industry Speaks." Retrieved from fmi.org: https://www.fmi.org/our-research/research-reports

[2] Bandyopadhyay, Soma & Thakur, S.. (2019). Product Recommendation for E-Commerce Data Using Association Rule and Apriori Algorithm: Proceedings of the International Conference on Modelling and Simulation (MS-17). 10.1007/978-3-319-74808-5_51.

[3] Bobadilla, J. O. (2013, July). Recommender Systems Survey. Knowledge-Based Systems, 46, 109-132.

[4] Bodhisattwa Prasad Majumder, S. L. (2019). Generating Personalized Recipes from Historical User Preferences. Association for Computational Linguistics. Hong Kong: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).

[5] Borgelt C., Kruse R. (2002) Induction of Association Rules: Apriori Implementation. In: Härdle W., Rönz B. (eds) Compstat. Physica, Heidelberg. https://doi.org/10.1007/978-3-642-57489-4_59

[6] Feeding America. "Feeding America Study Projects Local Food Insecurity Rates Amid Pandemic Could Reach Up To 1 in 3 Adults and 1 in 2 Children." Feeding America. Retrieved from: https://www.feedingamerica.org/

[7] Freuder, F. "Lexicographically-Ordered Constraint Satisfaction Problems." Constraints : an international journal 15.1 (2010): 1–28. Web.

[8] kaggle. (2020, 10 6). Retrieved from Food.com Recipes and Interactions: https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions

[9] Kumar, V. (1992) "Algorithms for Constraing-Satisfaction Problems: A Survey." AI Magazine, Spring 1992. 32-44. Print.

[10] Mayumi Ueda, S. A. (2014). Recipe Recommendation Method by Considering. Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 Vol I, . Hong Kong: IMECS.

[11] Melville, Mooney. "Content-Boosted Collaborative Filtering for Improved Recommendations." Eighteenth National Conference on Artificial Intelligence. American Association for Artificial Intelligence, 2002. 187–192. Print.

[12] Parker, K. (2020). "About Half of Lower-Income Americans Report Household Job or Wage Loss Due to COVID-19." Pew Research Center. Retrieved from: https://www.pewsocialtrends.org/2020/04/21/about-half-of-lower-income-americans-report-household-job-or-wage-loss-due-to-covid-19/

[13] Parker, K. (2020). "Economic Fallout From COVID-19 Continues To Hit Lower-Income Americans the Hardest." Pew Research Center. Retrieved from: https://www.pewsocialtrends.org/2020/09/24/

economic-fallout-from-covid-19-continues-to-hit-lower-income-americans-the-hardest/

[14] Sarkar, T. (2019, April 19). Retrieved from Towards Data Science: https://towardsdatascience.com/linear-programming-and-discrete-optimization-with-python-using-pulp-449f3c5f6e99

[15] Supermarket News. (2020, 10 6). Retrieved from It's a new scene for grocery shopping as pandemic changes behaviors: https://www.supermarketnews.com/consumer-trends/it-s-new-scene-grocery-shopping-pandemic-changes-behaviors

[16] USDA.gov. (2020, 10 6). Retrieved from USDA Economic Research Service: https://www.ers.usda.gov/data-products/ag-and-food-statistics-charting-the-essentials/food-prices-and-spending/

[17] Sarwar, Karypis. "Item-Based Collaborative Filtering Recommendation Algorithms." Proceedings of the 10th International Conference on World Wide Web. ACM, 2001. 285–295. Web.

[18] Wagner, J., Geleijnse, G., & van Halteren, A. (2011). Guidance and support for healthy food preparation in an augmented kitchen. 2011 Workshop on context-awareness in retrieval and recommendation.

[19] Wing, J. M. (2020, 10 6). The Data Life Cycle. Retrieved from Harvard Business Review MIT Press: https://hdsr.mitpress.mit.edu/pub/577rq08d/release/3

[20] Yajima, A. (2009). "Easy" Cooking Recipe Recommendation Considering User's Conditions. 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. Milan, Italy: IEEE.

Appendix

Paper url:

https://github.com/gmu-food-for-thought/food-for-thought