Recipe Recommendation

Sathwik Reddy Rajidi
University of North Texas
Denton, USA
sathwikreddyrajidi@my.unt.edu

Aniv Chakravarty
University of North Texas
Denton, USA
anivchakravarty@my.unt.edu

Blessy Kuriakose
University of North Texas
Denton, USA
blessykuriakose@my.unt.edu

***Abstract*— The objective of this project is to develop a recipe recommendation system that suggests top n recipes based on the ingredients provided by the user. The recommendation system is based on user-based collaborative filtering, which involves clustering users based on their reviews of different recipes. We will be using two different types of clustering, in two different implementations for recommendation. The two types are KNN and KMeans.**

Keywords—Recommendation, Collaborative clustering, KNN, KMeans, Vader, Sentiment analysis, Filtering

I. INTRODUCTION

People can spend hours looking up a workable recipe on the web only for them to find that they are missing crucial ingredients or for it to end up not suiting their palette. The job of this system is to alleviate these concerns and make cooking an easy and fruitful process. This project aims to develop a recipe recommendation system that suggests new recipes to users based on their ingredient preferences and the preferences of other users with similar tastes. By leveraging data with computational prowess, we can provide users with a personalized list of recipes that are highly correlated to their tastes. The system is designed to save users time and effort when looking for new recipes and help them discover new and exciting meals that they'll love.

This system should be able to…

- Filter recipes by ingredients.
- Analyze user preferences using sentiment analysis on past reviews.
- Cluster reviewers to predict new recipes the user might enjoy.
- Rank recipes using the above factors and display them as options.

A. Dataset

We will be using a dataset from Kaggle with 18 years of data collected from food.com. This contains two files, the recipes csv and the interactions csv, both of which will be utilized in this project.

1) Recipes csv: The recipe csv is a 280 MB file with 231637 rows and 12 columns. The details of the data are below:

- id: (Integer) A unique identification number given to the recipe.
- minutes: (integer) The time it takes to cook in minutes.
- contributer_id: (Integer) A unique identification number of the person who contributed the recipe.
- submitted: (Date) The date the recipe was added in mm/dd/yyyy format.
- tags: (list) A list of possible tags describing the recipe.
- nutrition: (list) A list of seven floating point values describing the nutritional values of the recipe.
- n-steps: (Integer) The number of steps in the recipe.
- steps: (list) An order list of text describing each step of the recipe.
- description: (String) Text describing personal notes of the contributor.
- ingredients: (list) ingredients in the form of strings.
- n-ingredients: (Integer) number of items in the ingredients.

2) Interactions csv: The second csv consists of a record of collaborative interactions and is a 333MB file with 1048575 rows and 5 columns. The details of the data are below:

- user_Id: (Integer) the unique identifier of the user
- recipe_Id: (Integer) the identifier from Id in the other dataset.
- date: (Date) the date of the review
- rating: (Integer) A rating system of value from 1 to 5 where 5 means good and 1 means bad, with 0 used for not rated.
- review: (String) Textual description of a review from the user.

It is important to note that the id from the Recipes csv and the recipe_id from the Interactions csv refer to the same encoding. We will later merge the two files based on this.

II. RELATED RESEARCH

Recommendation systems are implemented in many fields to successfully help lower the resistance for completing a task or process, but despite the varied use-cases, the considerations given to creating such a system have common layouts that can be categorically understood, as outlined in “Recommendation System –Understanding the Basic Concepts” [1]. There are two such categories to speak of: content-based and collaboration-based. Of the two, the latter has two further subcategories: item-based and user-based. Content-based refers to recommendations made based on the subject's own historical data. Item-based is similar, but from a collaborative standpoint, that is, the perspective of multiple users is considered when considering similarities between items and the more similar items to that in the target user’s profile are pushed forward as a recommendation. User-based is where user-clusters are made, and recommendations are given based

on the experience of users like the target user. (This last type is what we will be covering within this project.)

"Personalized Recipe Recommendation System using Hybrid Approach" [2] claims to take a step further from the conventional recommendation approach by adding various categories of filter. This paper uses a web crawler to retrieve 10,971 recipes and their reviews from food.com. (Same source as the dataset from Kaggle.) Their task takes the approach of predicting user ranking given recipe content and historical user ranking, through which they cluster users via a KNN model. They also implement a second approach using stochastic gradient descent. Our approach is similar in the clustering ideology; however, we aim to cluster groups based on reviews instead of recipe content.

When it comes to building clusters and calculating distance on a large scale is time consuming especially when incorporating collaborative recommendation and preferences. So instead, we obtain a subset of the preferences and their centroid representing the users' latent vectors which are represented on the item latent vector. Essentially finding key overlapping points that can be used to save on the overall computation time. In "Clustering and Constructing User Coresets to Accelerate Large-scale Top-K Recommender Systems" [3], they evaluated their model on recommendations and predictions on some well-known datasets like Wikipedia and Amazon to see the computation speed up rate which resulted in almost double to triple speed up compared to other methods like GMIPS, SVDS, FGD and L2S.

Hierarchical Graph attention network model (HGAT) uses relation level attention module on embeddings on a heterogeneous recipe graph where each is of three types which are recipe, ingredients, and users while edges are relations between the nodes. Attention modules are used on a node and relation level. In "Recipe Recommendation With Hierarchical Graph Attention Network" [4], they claim that the use of a neural network approach by considering the relations helped them achieve better performance to their baseline approaches like BPR, IngreNet, GAT etc., since they were able to capture more feature details along with their score and ranking based optimization functions.

Sentiment analysis or opinion mining is one of the major tasks of NLP (Natural Language Processing). Sentiment analysis has gained much attention in recent years. In this paper [5], they aim to tackle the problem of sentiment polarity categorization, which is one of the fundamental problems of sentiment analysis. Sentiment polarity categorization is nothing but classifying the statements (reviews,reports etc) as positive, negative or neutral. A general process for sentiment polarity categorization is proposed with detailed process descriptions. Data used in this study are online product reviews collected from Amazon.com. Experiments for both sentence-level categorization and review-level categorization are performed with promising outcomes.

III. Project Design

The basic outline of the methodology is as follows:

1. Input Ingredients Filtering: The user inputs a list of ingredients, which are used to filter the plausible recipes that contain all the specified ingredients. The plausible recipes are then stored for further analysis.
2. Adding Common Ingredients: If there are not enough plausible recipes returned in step 1, the system adds common ingredients that are present in both the high-ranked recipes and the recipes that the user has previously reviewed.
3. Plausible Recipes Filtering: The plausible recipes DataFrame is filtered again based on the common ingredients. Only the recipes that contain all the common ingredients are retained.
4. Sentiment Analysis on reviews: The reviews for the plausible recipes are the analysis using the sentiment intensity analyzer to generate the DataFrame column sentiment score.
5. Recipe Reviews by Common Cluster: The reviews for the plausible recipes are then clustered based on their polarity scores using a clustering algorithm, trained on the reviews of the users.
6. Ranked Results: The ranked results are obtained by averaging the ratings of the plausible recipes obtained from the KNN model. This is by using the sentiment score generating in step 4 and the clustering score generated in step 5. The top n recipes with the highest averaged ratings are recommended to the user.

The flowchart in **Figure 1** shows an overview of how the final model functions.

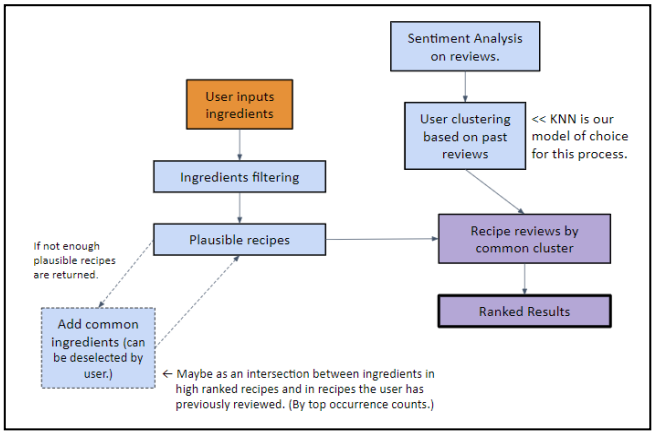

Fig. 1. This is a figure caption. It appears directly underneath the figure.

IV. Project Milestones

The steps taken in completing this project can be summarized in the six steps outlined in this section.

A. *Data cleaning*

We first load in these files using Pandas DataFrame. From there we begin by deleting unnecessary columns. The columns of interest for this project include user_id, recipe_id, rating, and review from the Interactions csv, and name, id, and ingredients from the Recipes csv. In the code-implementation, several other columns from the Recipes csv were kept for future developments, but they were not used in the course of this project.

After cleaning up the columns, we checked for null values, and ended up deleting rows of data in the Interactions csv that do not contain a review, as textual reviews are necessary to do sentiment analysis for clustering. Of the columns of interest

from the Recipes csv, there was only one null value in the name column, which was not of significance to the process of recommendation, so we opted to leave that row in.

After the basic cleaning, the two datasets are merged into one dataset called df, using the recipe_id from the cleaned Reviews csv and the id from the cleaned Interactions csv.

When the data was set up for use in the described manner, we had a total of 803334 rows of reviews on 204930 recipes from 12486 users.

It is notable that there was a significant imbalance on the star ratings, as can be seen in **Figure 2**.

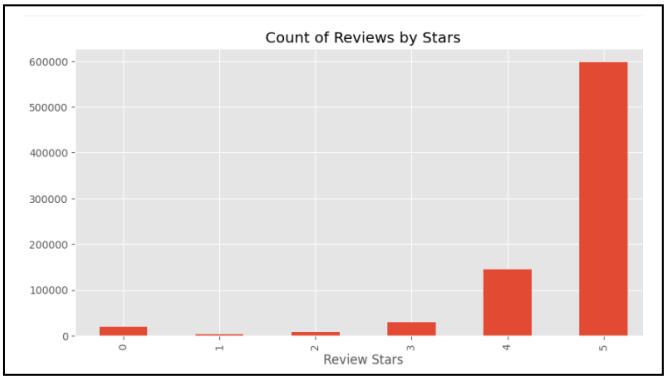

Fig. 2. Imbalance of ratings in original data.

B. Sentiment Analysis

For sentiment analysis we use nltk's SentimentIntensityAnalyzer() to get the compound Vader score for each review, pinning it as it's own column within the df dataset (dataset formed after merging). (As this process took a while to complete, this new DataFrame is saved for retrieval.) This score is a value that between -1 to 1 that tells how intensely negative or positive the review is.

After analysis, we took a brief look into the unrated reviews –namely the rows where the recipe 'rating', on a scale of 1 to 5, was given as 0. From a brief analysis, we found that some of these reviews contained sentiments that were not directed at the recipes themselves. While there is the option of pinpointing which reviews were aimed at the recipe and which were not, we opted to directly delete from the dataset, df, all reviews where the rating was 0. The dataset was large enough to accommodate this deletion, and any hand-coded rules for pinpointing were likely to have errors.

In **Figures 3 and 4** looked briefly at the remaining 765709 rows of reviews to see the distribution of star-ratings and Vader scores.

We see that the imbalance between negative reviews and positive reviews is not solved, however the existence of a continuous Vader score, opposed to the discrete ratings of 1 to 4, might assist in making distinctions when it comes to ranking. The sentiment analysis might also help in clustering between users of similar types and, most notably, in overcoming 'errored' ratings –whether intentional of not—by the reviewers.

Fig. 3. Vader score spread on dataset (after binning)

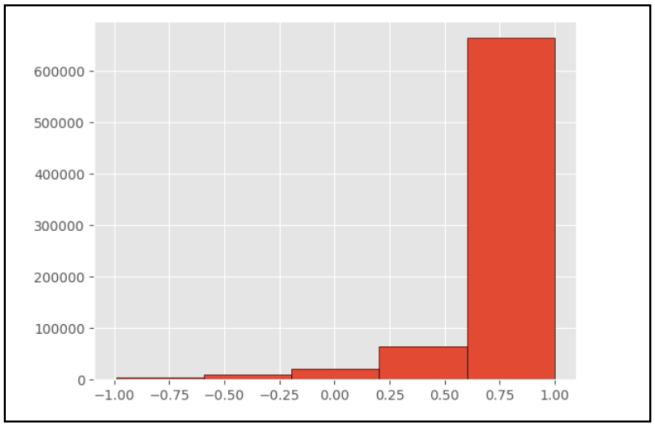

Fig. 4. Star ratings spread on dataset

C. Clustering with KNN

There were two clustering types used separately in this project, the first of which was KNN. As this method takes a long time to run, we used a 0.05 subset of the dataset to run a recommendation.

SkLearn's TfidfVectorizer() was used on the reviews and put in for KNN clustering using Sklearn's NearestNeighbors library. Using k=10 nearest neighbors we found the clusters and added it to the df.

D. Clustering with KMeans

We used sklearn's KMeans clustering to form 50 clusters on the entirety of the df dataset. The variables of 'user_id', 'recipe_id', and 'vader_compound' were inputted and the outputted clusters were added to the df as a second cluster column.

Upon review, we noted that this resulted in users falling into multiple, but distinct clusters. **Figure 5** shows the clusters for one of the users as well as how many of their reviews were marked with the given cluster.

```
[ ]  #...though a single user can belong to multiple clusters,
     #there are distinctions, as seen here. This user centers
     #around seven clusters
     train2[train2['user_id']==128473]['cluster2'].value_counts()

     27    1066
     30    1008
     26     733
     6      402
     3      394
     21     225
     Name: cluster2, dtype: int64
```

Fig. 5. KMeans clustering -- sample by user.

E. Filtering

For filtering, the first step is to get a list of ingredients from the user. This list is then compared to ingredients noted in the data. If the inputted ingredients are not enough to make a recipe out of, the program adds in ingredients to the list, then runs the filter again, until coming to a list of recipe_ids that include the usage of the ingredients inputted by the user, in addition to other common ingredients.

It should be noted that the recipes dataset used for the KNN model and the KMeans model are different, since the KNN might not have clustering data for some recipes due to the smaller data size used and thus cannot be used within the ranking of those recipes.

All the matches are forwarded for consideration.

F. Ranking

Since two different clustering models were used, two different ranked outputs were also produced.

1) In the KNN implementation, we do not use user clustering to create a custom recommendations, instead we group together the sentiments of same-cluster individuals and use the means of that to rank the candidate recipes in sub-training set. For this reason, this approach is namely based on the ingredients inputted by the current user, and historical data already present within the system.

2) In the KMeans implementation, there is a necessity for a user_id to be selected and role-played. There are two methods of ranking, but before this is done, the program first recognizes the clusters the current user are in, and then tracks down other users who are in the same cluster. All users who are within the clusters of the role-played user_id will get a say in the ranking by route of the sentiment analysis output of their reviews for the recipes under consideration. If there are enough of these associated users, then the program first homes in on the recipes that these users have previously reviewed, and then takes the mean (average) of only their Vader scores to rank the these second-level candidates. If there are not enough users within the same cluster as the roleplayed user, the program also takes the Vader scores of those outside the cluster, but divides it by 8 to lower the intensity/influence of their sentiments while also considering all candidate recipes returned by the filter. (This is expected to work since the Vader score is centered at 0.) Thus, a preference is created for the sentiments of those within the same cluster, though they may not be many, but the sentiments of other reviewers are still considered. This method of having a threshold on the number of same-cluster users necessary, helps keep the recipe recommendations tailored to the current user, but allows for a sufficient number of recipes to be considered in total, which is of great help to newer users with less historical data to assign them to a cluster.

```
ingredients = input('Enter ingredients (use comma`,` to separate: ').split(',')
run_recommendation(ingredients, user=13292)

Enter ingredients (use comma`,` to separate: beef,spinach


Results using KNN for clustering (on a smaller subset of the cleaned dataset):
Top 10 recipe recommendations:
1. cheesy ground beef spinach sour cream noodle casserole
2. the best italian manicotti
3. healthy manicotti
4. texas toast steak sandwiches
5. stuffed london broil
6. cheesy spinach and beef casserole
7. parmesan meatball soup
8. spinach beef spaghetti pie
9. florentine meatloaf
10. you ll never miss the noodles  lasagna


Results using KMeans for clustering (on the cleaned dataset):
Top 10 recipe recommendations:
1. creamy greek tomato noodle soup
2. sausage with beans and macaroni
3. the best italian manicotti
4. pasta e fagioli soup with ground beef and spinach
5. healthy manicotti
6. layered beef noodle bake
7. individual italian meat loaves
8. tuscan soup
9. ground beef florentine
10. chef berry s stuffed manicotti

ingredients = input('Enter ingredients (use comma`,` to separate: ').split(',')
run_recommendation(ingredients, user=1329782)

Enter ingredients (use comma`,` to separate: beef,spinach


Results using KNN for clustering (on a smaller subset of the cleaned dataset):
Top 10 recipe recommendations:
1. cheesy ground beef spinach sour cream noodle casserole
2. the best italian manicotti
3. healthy manicotti
4. texas toast steak sandwiches
5. stuffed london broil
6. cheesy spinach and beef casserole
7. parmesan meatball soup
8. spinach beef spaghetti pie
9. florentine meatloaf
10. you ll never miss the noodles  lasagna


Results using KMeans for clustering (on the cleaned dataset):
Top 10 recipe recommendations:
1. moroccan meatball stew
2. turkish spinach and lentil soup
3. lina s awesome lebanese spinach  beef   rice
4. homemade stock
5. savory sausage and sweet potato stew
6. low carb mexican beef and spinach casserole
7. rollasagna
8. beefy spinach and rice
9. baked manicotti
10. steak gorgonzola with balsamic reduction over pasta
```

Fig. 6. Results of two sample runs role played as different users.

V. Results

Due to the unsupervised nature of this task, we did not get into accuracy metrics, but we can look at the results produced by the models.

The recommendation system successfully and accurately recommends the recipes that contain all the input ingredients and ranks them based on their averaged ratings by the users in the same cluster.

As we have noted while working on this project, the K Nearest Neighbors algorithm of sklearn, takes longer to run than the KMeans algorithm of the same library, so by computational comparison, we might consider the latter to be better, however, further analysis is necessary to know which one does a better job in terms of recommendations. It should be noted that the approaches of how the clustering labels are

used to make a recommendation also vary, as denoted under the Ranking subsection of the project milestones, so it is not an apples-to-apples comparison in the current code. (Though it can be made into one quite easily we opted to keep more variety for the turn in.)

VI. Conclusion and future works

We made a working model that does what we aimed for it to do, however there are several improvements that could be made to further improve this project.

On the filter side, there could be more tailoring to home in on recipes more suited to the ingredients given by the user. The filter could also add in other items such as time requirements or calorie counts, that can be (optionally) decided by the user. The added ingredients (used in the case that the user input ingredients are insufficient) could be tailored to the ingredients previously used by the user instead of common ingredients at a more general level.

User clustering methods could be improved by using different algorithms or different metrics for sentiment analysis. On the side of sentiment analysis, it may be possible to be more precise in terms of what sentiments belong to the recipe and what belong to external factors faced by the user. (We only addressed this by dropping reviews with ratings of 0, but a more sophisticated method might be possible using taggers and other NLP libraries.)

Overall, this approach led to the creation of a successful program that helps recommend recipes based on user reviews.

References

[1] Shubham Kumar Agrawal. “Recommendation System -Understanding The Basic Concepts.” Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/recommendation-system-understanding-the-basic-concepts/ [Accessed Mar. 10, 2023]

[2] Gaudani, Hetal. (2016). “Personalized Recipe Recommendation System using Hybrid Approach.” International Journal of Innovative Research in Computer and Communication Engineering. 5. 192-197. 10.17148/IJARCCE.2016.5642. [Accessed Mar. 10, 2023]

[3] Jyun-Yu Jiang, Patrick H. Chen, Cho-Jui Hsieh, and Wei Wang. 2020. Clustering and Constructing User Coresets to Accelerate Large-scale Top-K Recommender Systems. In Proceedings of The Web Conference 2020 (WWW '20). Association for Computing Machinery, New York, NY, USA, 2177–2187. https://doi.org/10.1145/3366423.3380283 [Accessed Mar. 10, 2023]

[4] Tian Y, Zhang C, Metoyer R and Chawla NV (2022) Recipe Recommendation With Hierarchical Graph Attention Network. Front. Big Data 4:778417. doi: 10.3389/fdata.2021.778417 https://www.frontiersin.org/articles/10.3389/fdata.2021.778417/full

[5] Fang, X., Zhan, J. Sentiment analysis using product review data. Journal of Big Data 2, 5 (2015). https://doi.org/10.1186/s40537-015-0015-2