Cuisine Recommendation System

1st Himang Chandra Garg
Computer Science and Engineering
Indraprastha Institute of Information Technology
Roll no.- 2022214
himang22214@iiitd.ac.in

2nd Nishil Agarwal
Computer Science and Artificial Intelligence
Indraprastha Institute of Information Technology
Roll no.- 2022334
nishil22334@iiitd.ac.in

***Abstract*—In this report, we present the design and implementation of a cuisine recommendation system aimed at assisting users in discovering recipes that align with their tastes and dietary requirements. Leveraging machine learning techniques such as content-based and collaborative filtering, our system analyzes user preferences, ingredient profiles, and historical interactions to generate personalized recipe suggestions.**

I. GitHub Repository

The source code for our project can be found at: https://github.com/himangg/Cuisine-Recommendation-System

II. Problem being addressed

The problem being addressed in this project is the need for a robust cuisine recommendation system that can assist users in discovering recipes aligned with their tastes, preferences, and dietary requirements. With the vast amount of recipe data available online, users often struggle to find recipes that suit their specific needs and interests. Additionally, users may also desire personalized recommendations based on their past interactions and feedback.

III. Relevant Literature

Existing research in cuisine recommendation systems has focused on collaborative filtering, content-based filtering, and hybrid approaches. However, these methods face challenges such as sparsity of user-item interaction data, limited diversity in recommendations, and scalability issues. The cold-start problem remains a key challenge, especially for new users or items with sparse data. Future research should aim to address these limitations by developing more effective recommendation techniques, enhancing diversity in recommendations, and improving scalability and efficiency. Also, there is no significant way to evaluate the accuracy of the recommendations. We have tried to do so by using a cosine similarity heatmap and precision at k. Additionally, advancements in deep learning, natural language processing, and user modeling can offer new opportunities for enhancing the accuracy, diversity, and personalization of recipe recommendations.

IV. Dataset

For our project, we utilize a comprehensive dataset from Kaggle, comprising a diverse collection of Food recipes showcasing various regional cuisines and culinary traditions. The dataset includes essential information such as recipe names, ingredients, preparation time, nutritional content, and user ratings. In the earlier deadline we stated that we will be using. https://www.kaggle.com/datasets/kritirathi/indian-food-dataset-with/data but due to less rows we have changed our dataset to https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews.

V. Data Preprocessing

In this section, we detail the preprocessing steps applied to the recipe dataset to prepare it for further analysis and modeling.

A. Reading and Filtering Dataset

The recipe dataset is initially read using the Pandas library, loading it into a DataFrame named `data`. Subsequently, irrelevant columns are filtered out to focus on essential attributes relevant to the recommendation system to reduce noise prevent unnecessary complexity.

B. Removing Unnecessary Rows and Columns

To enhance data consistency, we removed entries with missing ratings, incorrect/erroneous data etc. using the `drop` function.

C. Selecting Diverse Rating Dishes

Further, we stratified the dataset based on aggregated ratings, selecting a balanced mix of highly rated and lower-rated recipes for model training.

D. Modifying Column Formats, Handling Missing and Erroneous Data

Additionally, we brought the entries of date_of_publication, cooking_time, ingredients columns into the required format for efficient handling of data. We also handled missing values in various columns, ensuring consistency and completeness.

E. Removing Duplicate Entries

Duplicate recipe names are removed from the dataset, ensuring uniqueness and avoiding redundancy in recommendations.

F. Merging Datasets

The preprocessed recipe dataset is merged with the reviews dataset based on `RecipeId`, facilitating the integration of user ratings into the recommendation system.

G. Filtering Reviews

Reviews from reviewers with less than five reviews are removed from the dataset, ensuring that only substantial reviews are considered in the analysis.

H. Saving Processed Data

The final preprocessed dataset is saved to a new file for further analysis and modeling.

VI. Data Visualization

In this section, we present visualizations of various aspects of the processed dataset to gain insights into recipe categories, rating distribution, publication trends, calorie distribution, and top recipes and authors based on review count.

A. Distribution of Recipe Categories

The distribution of recipe categories is visualized using a pie chart. Rare categories, representing less than 1 percent of the dataset, are grouped under 'Others' to simplify the visualization.

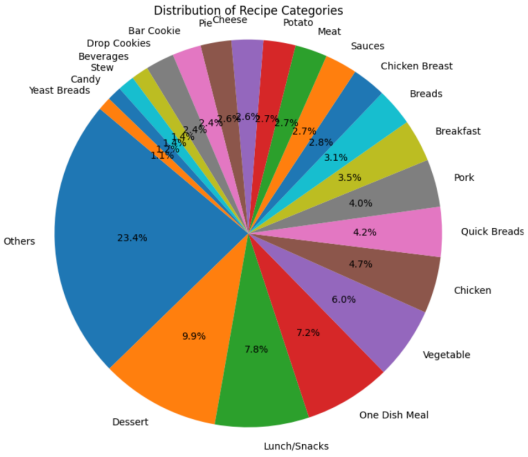

Fig. 1. Distribution of Recipe Categories

B. Distribution of All Ratings

A count plot is used to visualize the distribution of all ratings. This plot provides an overview of the distribution of ratings assigned to recipes in the dataset.

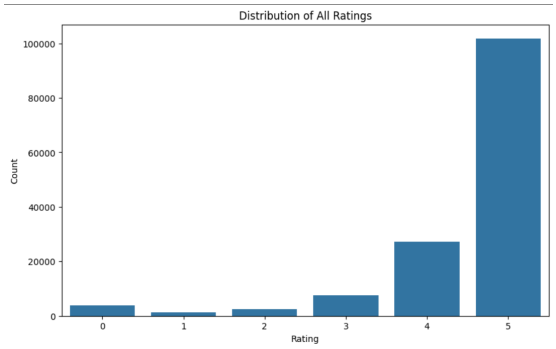

Fig. 2. Distribution of All Ratings

C. Monthly Recipe Publication

A line plot is utilized to illustrate the monthly publication of recipes over time. This visualization helps identify trends and patterns in recipe publication frequency.

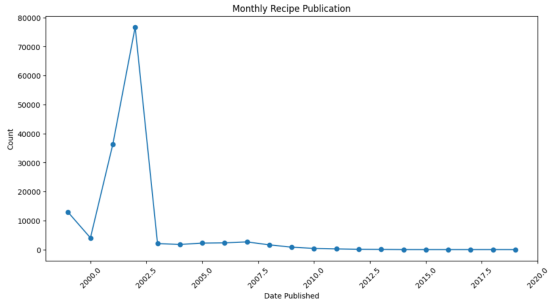

Fig. 3. Monthly Recipe Publication

D. Distribution of Calories

The distribution of calories in recipes is displayed using a pie chart, categorizing calorie ranges into bins. This visualization offers insights into the distribution of calorie content across recipes.

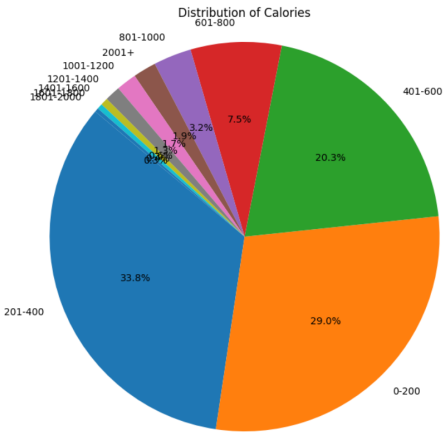

Fig. 4. Distribution of Calories

E. Top 10 Recipes by Review Count

A bar plot is employed to showcase the top 10 recipes based on review count. This visualization highlights recipes that have received the highest number of reviews from users.

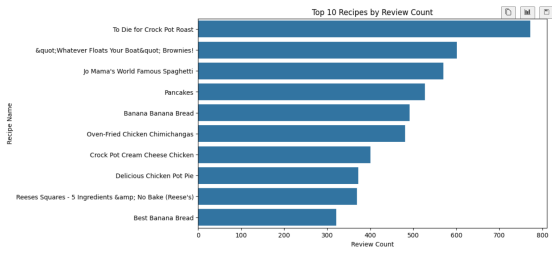

Fig. 5. Top 10 Recipes by Review Count

F. Top 10 Authors by Review Count

Similarly, a bar plot is utilized to present the top 10 authors based on review count. This visualization identifies authors who have contributed the most reviewed recipes in the dataset.

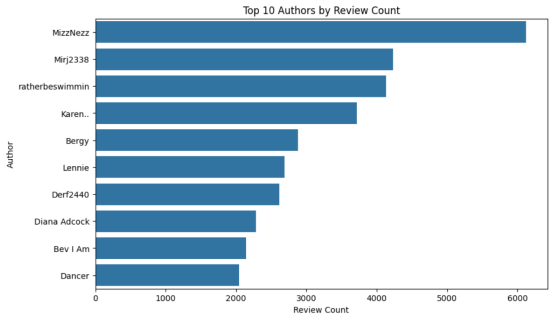

Fig. 6. Top 10 Authors by Review Count

Each visualization offers valuable insights into different aspects of the recipe dataset, aiding in understanding user preferences, recipe popularity, and publication trends.

VII. Content-Based Filtering

Content-based filtering is a recommendation technique that suggests items to users based on their preferences and item features. In this section, we describe the implementation of content-based filtering for recipe recommendations using ingredient information.

A. Extracting Unique Recipes

First, unique recipes are extracted from the dataset to ensure that each recipe is considered only once for recommendation.

B. TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is applied to the recipe ingredient parts to convert text data into numerical vectors. This process captures the importance of each ingredient in the recipe while considering its frequency across all recipes.

C. Computing Cosine Similarity and Indexing

Cosine similarity is calculated between the TF-IDF vectors of recipes to measure the similarity between them. An index is created to map recipe names to their corresponding indices in the dataset for efficient retrieval.

D. Recommendation Function

A recommendation function is defined to provide recommendations based on the similarity scores calculated using cosine similarity. Given a recipe title, the function returns a list of top recommended recipes with similar ingredient profiles.

E. Printing Recommendations

Recommendations are printed for sample recipe titles, such as 'Buttermilk Pie' and 'Potato Salad'. The function returns a list of recommended recipes based on the ingredient similarities with the input recipe.

—

Example Recommendations:

For the input recipe 'Buttermilk Pie', the following recommendations are provided:

- Chocolate Dessert Crepes
- Easy Red Velvet Cake
- Red Velvet Waffles
- Filled Coffee Cake
- Peanut Butter Fudge Cake

For the input recipe 'Potato Salad', the following recommendations are provided:

- Amish Potato Salad
- Mom's Danish Potato Salad
- Kathy's Macaroni Salad
- Curry Deviled Eggs With Cilantro
- Pickled Eggs

F. Calculation of Diversity Ratio

The diversity ratio is calculated as the ratio of the number of unique recommended items to the total number of recommendations made. This metric provides insights into the variety and diversity of recommendations generated by the content-based filtering model.

Diversity Ratio: 0.17871140939597316

G. Analysis of Results using Cosine Similarity Heatmap

A cosine similarity heatmap is generated to visualize the pairwise similarities between recipes before and after applying content-based filtering. This analysis provides insights into how the filtering process affects the overall similarity structure of the recipe dataset.

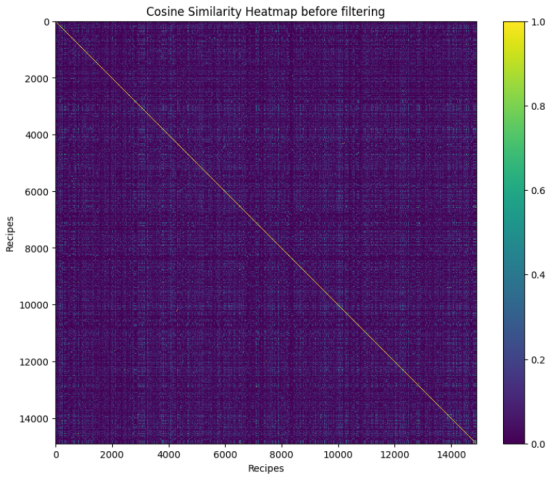

Fig. 7. Cosine Similarity Heatmap before filtering

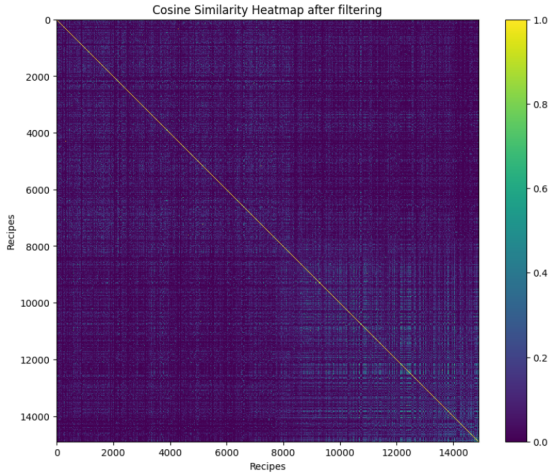

Fig. 8. Cosine Similarity Heatmap after filtering

Here, we can see that the lighter region is increased after doing content-based filtering, proving that our algorithm is working correctly.

VIII. Collaborative Filtering

Collaborative filtering is a recommendation technique that identifies patterns in user behavior and preferences to recommend items to users. In this section, we describe the implementation of collaborative filtering for recipe recommendations using a reviewer-recipe-rating matrix.

A. Creating Reviewer-Recipe-Rating Matrix

A reviewer-recipe-rating matrix is constructed from the processed dataset, where each row represents a reviewer, each column represents a recipe, and the values represent the ratings given by reviewers to recipes. Missing ratings are filled with zeros.
Ratings may also be normalised to penalise ratings of those who ovverrate/underrate irratically.

B. Calculating Cosine Similarity Matrix

Cosine similarity is computed between reviewers based on their rating vectors using the reviewer-recipe-rating matrix. This similarity matrix captures the similarity between pairs of reviewers, indicating how alike their preferences are.

C. Predicting Ratings for Missing Entries

Predictions for missing ratings for each user are made one by one. This involves identifying similar users and using their ratings for the target recipe to predict missing rating. The predictions are based on the cosine similarity scores between users.
It is used to in a way to assign optimum weights to each other reviewer's rating. The users having rating vector similar to target user will have higher weight and vice versa. Weighted sum of these ratings is taken to predict target user's rating for the recipe.

D. Printing Recommendations

Recommendations for user = "Bergy":

- Ham & Egg Casserole
- Shrimp Stir-Fry With Bok Choy, Mushrooms & Pep...
- Perch or Snapper Fillet With Tomatoes and Onion
- Filet of Sole With Spinach & Tomatoes
- Turtle Cookies

E. Analysis of results

Used k-fold cross validation and then further divided val set into 2 parts. One that serves for predicting likes and dislikes. Other that serves as ground truth.

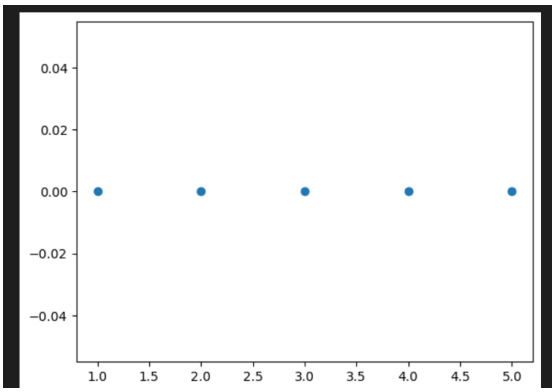

Fig. 9. Accuracy on various test datasets generated while applying cross validation

We got extremely poor results for accuracy as data is sparse in cosine similarity matrix. Most dishes being suggested are simply those which are liked by most as hardly any preference of users is matching due to sparseness.

Also, we are using the user's already rated dishes for ground truth. As most user have hardly tried any dishes, hence the suggested dishes (from among 14000 dishes) may not belong to the small set of ground truth anyways! There might exist many better dishes suited for user than the ones he has already tried (used as ground truth).
Hence, weights for each user's rating are not getting calculated as desired.

This issue might get rectified if all users have tried a lot of dishes. (User-Recipe-Rating matrix is not sparse)
However this is unlikely to be a practical case in an actual food recommender system.

IX. K-Nearest Neighbors (KNN) Clustering

A. Introduction

K-Nearest Neighbors (KNN) clustering is a popular machine learning technique used for recommendation systems. It works by finding the k nearest neighbors to a given data point based on a similarity metric, such as cosine similarity.

B. Methodology

In our recommendation system, we apply KNN clustering to identify recipes with similar ingredient profiles. The methodology involves preprocessing the dataset, vectorizing recipe ingredients using TF-IDF, and fitting a KNN model to find nearest neighbors.

C. Example Recommendations

We present below the example recommendations generated by the recommendation function using the K-Nearest Neighbors (KNN) clustering technique:

1) For the Recipe 'Buttermilk Pie':

- Fried Cornbread II
- Easy Red Velvet Cake
- Chocolate Dessert Crepes
- Whole Wheat Buttermilk Rusks
- Sunrise Cherry Pie

2) For the Recipe 'Potato Salad':

- Potatoes Tapas in Garlic Mayonnaise (Potatoes Aioli)
- Cold Picnic Potato Salad
- Quick Baked Potato
- Cheesy Hash Browns for One
- Irish Fadge (Potato Cakes)

These recommendations demonstrate the effectiveness of the K-Nearest Neighbors (KNN) clustering in providing relevant and diverse recipe suggestions based on ingredient similarity.

D. Analysis of Results using Cosine Similarity Heatmap

To analyze the effectiveness of the recommendation system, a cosine similarity heatmap is generated to visualize the pairwise similarities between recipes after applying content-based filtering.

The heatmap is generated using the cosine similarity matrix, where each cell represents the similarity score between two recipes. Lighter cells indicate higher similarity scores, suggesting that the corresponding recipes are more similar.

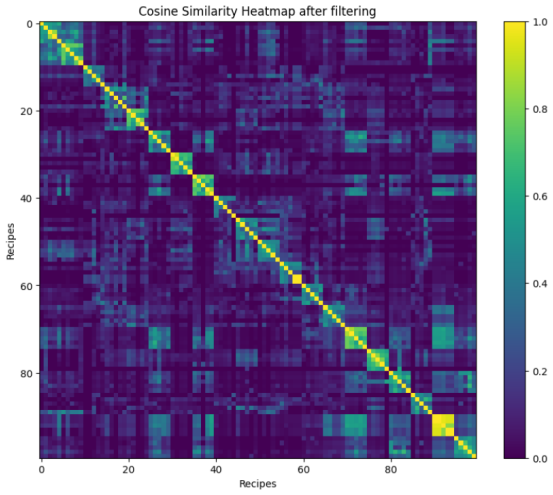

Fig. 10. Heatmap after KNN clustering

We can observe that squares of sides equal to cluster size are formed along the diagonal. This shows that clusters are created with similar recipes within then.

This analysis provides insights into how the content-based filtering affects the similarity structure of the recipe dataset and helps evaluate the quality of the recommendations.

X. Neural Network

Feedforward neural networks, also known as multilayer perceptrons (MLPs), are a fundamental type of neural network architecture. They consist of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer.

In a feedforward neural network, information flows forward from the input layer through the hidden layers to the output layer without any feedback connections. Each neuron in the network computes a weighted sum of its inputs, applies an activation function to the result, and passes the output to the next layer.

A. Scaling input data for training of NN

We had to first scale data to the form that is required i.e. decimal and we also standardised the data.

B. Define structure of NN

We had to define hidden layers on NN. We chose after some experimenting having 2 hidden layers. 1st one having 32 nodes and 2nd one having 8.

The output layer has only a single node representing the rating that has been predicted.

C. MSE Loss

We trained model for 5 epochs and plotted a graph for MSE.

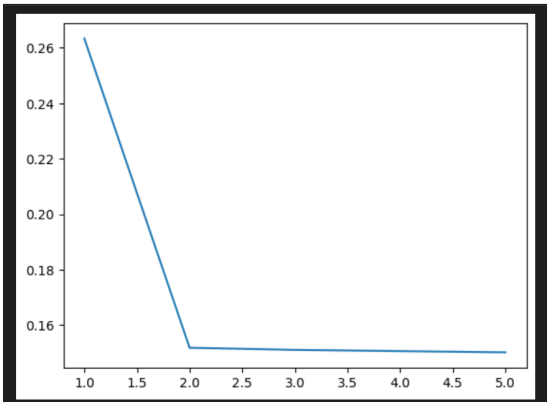

Fig. 11. MSE Graph after training

D. Example Outputs

For user ID = 1060485 ("allyop") We recieved the following results :

- Chicken Armenian
- Gracie Allen's Classic Roast Beef
- Italian Beef
- Old Fashioned Oatmeal Cookies

XI. Comparison and Conclusion

Each recommendation method has its strengths and limitations. However, since we cannot directly compare them because this is a problem of unsupervised learning, we have listed their pros and cons. Also, the results can be manually confirmed by user testing, but this method is very exhaustive. Here are the details of each method:

A. Collaborative Filtering

Collaborative filtering excels in capturing user preferences and generating personalized recommendations. By analyzing user-item interaction data, it can identify similar users and recommend items based on their preferences. However, collaborative filtering may suffer from the cold start problem, where new users or items have limited interaction data. Additionally, the sparsity of user-item interaction data can affect the accuracy of recommendations.
This is the case with our dataset which results in extremely poor accuracy.

B. Content-Based Filtering

Content-based filtering leverages item features, such as ingredients in recipes, to provide recommendations based on item similarity. By analyzing the attributes of items, it can recommend items with similar attributes. However, content-based filtering may struggle with surprising or showing you something unexpected, as it primarily relies on the characteristics of items.

C. K-Nearest Neighbors Clustering

K-Nearest Neighbors clustering groups similar items and recommends items based on their neighbors within the same cluster. By identifying closely related items, it can provide recommendations based on item similarity. However, KNN clustering may not capture diverse recommendations, as it tends to recommend items within the same cluster.

D. Deep Learning

Deep learning is the most advanced technique known till date. It has the potential to learn complex patterns and representations from data, leading to highly accurate and personalized recommendations. However, deep learning models require large amounts of data and computational resources for training, making them challenging to deploy in real-world applications. We find that NN works quite well with out dataset and also shows significantly less error.

E. Conclusion

In conclusion, each recommendation method offers unique advantages and challenges. A hybrid approach that combines multiple methods can overcome individual limitations and enhance recommendation quality. Furthermore, manual confirmation through user testing can validate the effectiveness of recommendation methods in real-world scenarios. Future research may focus on developing hybrid recommendation systems and exploring advanced techniques to address specific challenges in recommendation systems.
In our manual trial runs based on various dishes and users, we find that potentially due to sparsity of rating data, the Content based Filtering and Neural Network methods work the best.

XII. Individual Contributions

A. Himang Chandra Garg

- Implemented data visualization techniques using Matplotlib and Seaborn libraries.
- Developed the content-based filtering algorithm using TF-IDF vectorization and cosine similarity.
- Used the K-Nearest-Neighbour approach to recommend based on clusters.
- Showed that the above techniques give similar results by plotting the graph of Cosine similarity Heatmap.

B. Nishil Agarwal

- Implemented data preprocessing pipeline, including filtering, cleaning, and formatting of the dataset.
- Developed the collaborative filtering algorithm for generating user-based recommendations by using cosine similarity matrix.
- Showed accuracy results through k-fold cross validation and sub-sampling on collaborative filtering
- Developed Neural Network method and generated accuracy graphs for it.

References

[1] https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews
[2] https://www.nature.com/articles/s41598-022-10358-x
[3] https://a-elkhattam.medium.com/imdb-movie-recommendation-chatbot-942f84dfa0dc
[4] https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/
[5] https://www.geeksforgeeks.org/cross-validation-machine-learning/
[6] https://medium.com/@azadmeshram/evaluation-of-collaborative-filtering-model-6559b1a49b77
[7] https://prabhatm27.medium.com/creating-a-hybrid-neural-network-for-movie-recommendations-using-tensorflow-recommenders-fdad13aa4979
[8] https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/