Enhancing Multi-Criteria Recommendation Systems with Hot Deck Imputation and User-Specific Similarity Measures

Nithish Chouti[1], Sakshi Kusale[1], Sanket Mishra[1], Shashank Rajora[1], Bam Bahadur Sinha[2], Prabhu Prasad B M[1], Manjunath K V[1]

Indian Institute of Information Technology Dharwad, Karnataka - 580001, India[1]
National Institute of Technology Sikkim, India - 737139[2]

Corresponding author: Bam Bahadur Sinha, Email: bahadurbam43@gmail.com

Recommender systems are pivotal in the modern digital landscape, where consumers are inundated with choices, leading to decision fatigue. These systems offer personalized suggestions, alleviating the burden of choice by tailoring recommendations to user preferences. One of the challenges faced by recommender systems is dealing with sparse user-item matrices, often resulting from limited user preferences. To overcome this, we adopt a novel approach by categorizing users into three groups based on their rating activities: cold start users, middle users, and heavy users. This categorization enables us to provide more accurate and personalized recommendations. Furthermore, we utilize multi-criteria similarity measures to identify user preferences. Unlike many existing recommender systems that focus on average or overall criteria, our approach considers individual user behaviors, leading to more precise recommendations. It is worth noting that while multi-criteria recommender systems exist, they do not often group users together to enhance results. By categorizing users, we can better understand their preferences and tailor recommendations accordingly. To address sparse matrices, we employ hot deck imputation, filling in missing values by borrowing information from similar records. This strategy ensures that our recommendations are based on a more complete dataset, improving the overall accuracy and effectiveness of our recommender system.

Keywords: Multi-criteria Recommender systems, Collaborative filtering, Hot deck imputation, Similarity measures.

2025. In Mukesh Saraswat & Rajani Kumari (eds.), *Applied Intelligence and Computing*, 313–324. Computing & Intelligent Systems, SCRS, India. DOI: https://doi.org/10.56155/978-81-955020-9-7-30

1 Introduction

In today's digital landscape, characterized by rapid technological advancements and heightened consumer expectations, the shopping experience has evolved significantly. Consumers now have access to a vast array of choices, both online and offline. However, this abundance of options can often lead to decision fatigue, where consumers feel overwhelmed by the sheer number of choices available to them. This can result in prolonged comparison processes and, ultimately, a less satisfactory shopping experience.

Recommender systems play a crucial role in addressing these challenges by offering personalized suggestions based on user preferences. These systems use algorithms to analyze user behavior and past interactions to recommend items that are likely to be of interest to them. By tailoring recommendations to individual preferences, recommender systems help streamline the decision-making process for consumers, making it easier for them to find products that meet their needs and preferences.

One of the key challenges faced by recommender systems is dealing with sparse user-item matrices. Sparse matrices occur when many users have rated only a few items or have provided ratings sporadically. This can make it difficult for recommender systems to accurately predict user preferences and provide relevant recommendations [1][16].

To address this challenge, our approach involves categorizing users into three groups based on their rating activities: cold start users, middle users, and heavy users. Cold start users are those who have rated fewer than five items, middle users have rated between five and 15 items, and heavy users have rated more than 15 items. This categorization allows us to better understand user behavior and preferences, enabling us to provide more accurate and personalized recommendations.

Furthermore, we utilize multi-criteria similarity measures to identify user preferences. Unlike many existing recommender systems that focus on average or overall criteria, our approach considers individual user behaviors, leading to more precise recommendations. By incorporating these measures, we can better capture the nuances of user preferences and provide more relevant suggestions.

It is worth noting that while multi-criteria recommender systems exist, they do not often group users together to enhance results. By categorizing users, we can better understand their preferences and tailor recommendations accordingly.

To address sparse matrices, we employ hot deck imputation, filling in missing values by borrowing information from similar records. This strategy ensures that our recommendations are based on a more complete dataset, improving the overall accuracy and effectiveness of our recommender system.

In conclusion, our approach aims to enhance the effectiveness of recommender systems by addressing common challenges such as sparse matrices and limited user preferences. By categorizing users, employing multi-criteria similarity measures, and using hot deck imputation, we can provide more accurate and personalized recommendations, ultimately improving the user experience.

2 Literature Review

Recommender systems play a crucial role in modern digital platforms, assisting users in navigating the overwhelming abundance of choices available to them. These systems are often classified into two main categories: collaborative filtering and content-based recommender systems [2]. Collaborative filtering systems identify groups of users with similar preferences and recommend items based on the preferences of these groups, while content-based recommender systems recommend items similar to those previously liked by the user [3][15].

Collaborative filtering encompasses various techniques, including memory-based and model-based approaches. Memory-based approaches calculate similarity among users using metrics such as cosine similarity or Pearson correlation. Model-based approaches utilize machine learning algorithms, such as PCA, SVD, or matrix factorization, to track users' ratings and predict their ratings for unrated items [4].

One important aspect of collaborative filtering is neighborhood models [10][11], where the system uses the behavior of similar users to make recommendations. These models rely on identifying a subset of users who are similar to the active user and combining their ratings to provide personalized recommendations [5]. These approaches can be classified into user-based or item-based recommendation, depending on whether they focus on similarities between users or items [6].

Hot deck imputation is a technique commonly used to handle missing data in datasets, including those used in recommender systems. This method replaces missing values with observed values from similar cases, ensuring that the imputed values are consistent with the underlying patterns in the data. Hot deck imputation is advantageous because it preserves the structure of the original dataset and does not rely on complex modeling, making it less sensitive to model assumptions [7][12].

Recent research in recommender systems has focused on enhancing the effectiveness of multi-criteria recommendation systems. These systems integrate user preferences across multiple dimensions, providing more detailed information about user preferences and items. Multi-criteria recommendation systems have shown promising results in improving predictive accuracy and adapting to various real-world scenarios [8][13].

3 Problem Description

Multi-criteria rating systems have garnered significant attention due to their ability to more accurately capture user preferences across various dimensions. However, existing research predominantly treats all users as part of the same homogeneous group, failing to account for the diverse needs and behaviors of users at different activity levels, such as cold-start, middle, and heavy users. Moreover, the challenge of determining the most suitable measure for each user group and selecting appropriate imputation meth-

ods to address similarity calculation further complicates the effectiveness of existing approaches.

In response to this issue, a more nuanced approach is proposed, wherein users are categorized into distinct groups based on their activity levels. This segmentation enables the application of tailored methodologies for each user group, thereby enhancing the system's performance, particularly for new or cold-start users. Techniques such as Pearson correlation and multi-dimensional distance metrics are commonly employed to assess similarity among users within multi-criteria rating systems.

4 Proposed Algorithm

In our proposed multi-criteria recommender system, we start by dividing users into three groups based on their overall number of movie ratings: cold-start users, middle-level users, and heavy users. Cold-start users are those with fewer than 5 ratings, middle-level users have rated between 5 and 15 movies, and heavy users have rated more than 15 movies. This categorization helps us understand and cater to the varying levels of user activity and experience within the system.

One of the challenges in recommender systems is dealing with missing ratings, which can lead to sparse matrices and affect the accuracy of recommendations. To address this, we employ KNN hot deck imputation. This technique uses the ratings of similar users (neighbors) to predict missing ratings for active users. By considering the ratings of similar users, we can better estimate the preferences of users with missing ratings.

Additionally, we calculate the similarity among users for each criterion. For example, if we consider movie genres as criteria, we calculate the similarity between users based on their ratings for movies in each genre. This process is repeated for all criteria, and the individual similarities are then aggregated to form an overall similarity between users. This aggregated similarity helps us predict overall ratings for active users, taking into account their preferences across different criteria.

Finally, we evaluate the performance of our recommender system using precision metric. These metrics help us assess how well our system is able to recommend movies to users based on their preferences. By analyzing the performance of different similarity measures for each user group, we can determine the most effective approach for improving the accuracy of our recommendations.

5 Experiment

5.1 Dataset Description

The dataset used in this study consists of user reviews for hotels, with the following attributes: 'review_id', 'member_id', 'hotel_id', 'rating', 'recommend_list', and 'review_-

text'. The 'recommend_list' contains multiple criteria ratings (e.g., Value, Location, Sleep Quality, Rooms, Cleanliness, Service). Dataset description is as follows:

- Total users: 3453
- Total hotels: 1832
- Total reviews: 21826
- Maximum hotels rated by one user: 24
- Minimum hotels rated by one user: 1
- Average hotels rated by one user: 5.91
- Columns: ['review_id', 'member_id', 'hotel_id', 'rating', 'recommend_list', 'review_-text']

  where the column 'recommend_list' is string having following attributes:

  Value;5:Location;5:Sleep Quality;5:Rooms;5:Cleanliness;4:Service;5:Checkin;3:Business Service

5.2 Preprocessing on the dataset

Following preprocessing steps are done on the dataset:

1. Dropping the unnecessary rows, having different values in the 'recommend_list'.
   - We have in our dataset an overall rating and 6 classes (Value, Location, Sleep Quality, Rooms, Cleanliness, Service) for rating the hotel in recommend_list column.
   - Few users have rated the hotels on the basis of 'business' and 'check' which will lead to inconsistencies proceeding further in our algorithm.
   - Such users have been dropped from the dataset
2. Dropping duplicate rows
   - Users who have rated the same hotel more than once are dropped.
   - In other words, we only retain unique user- hotel pairs.
   - Rating of user for the hotel done at the first time is retained.
3. Dropping irrelevant columns from our dataset
   - Columns 'review_id' and 'review_text' are not useful for further calculations and are thus dropped.

4. Renaming columns
    - We rename the 'rating' column as 'overall'
5. Appending rating criteria as separate columns to the original dataset
    - 'recommend_list' contains the rating criteria → 'value', 'location', 'sleep_quality', 'rooms', 'cleanliness', 'service'
    - We split these string values into separate columns and append it to columns 'member_id', 'hotel_id' and 'overall' columns

5.3 Data Split into different groups based on number of user ratings

We are splitting our data into 3 groups based on the number of ratings done by a user. This splitting is done using a function which calculates the number of ratings of each user and classifies them accordingly.

The data configuration after splitting is as follows:

Table 1: Data statistics after cold start, middle, heavy users division

	User Activity		
	Cold start users	**Middle users**	**Heavy users**
Total users	1909	1446	15
Total hotels	1635	1346	238
Total reviews	4455	11640	259

5.4 Train Test Split

In our approach, we use a 90-10 train-test split ratio, where 90% of the data is used for training and 10% for testing. To ensure the accuracy of our predictions, we remove hotels and users from the test set that are not present in the training set. This is because it is difficult to predict ratings for hotels that have not been previously rated by any user.

We implement this removal process using a function that checks each entry in the test set. If the entry is not found in the training set, it is removed from further consideration. This ensures that our model is evaluated only on data that it has been trained on, improving the reliability of our predictions.

Consecutively the cold start users, middle users and the heavy users are spilt into train and test sets.

Following are the statistics of this split.

6 Implementation Method

First, we construct a user-item matrix for each criteria , where each row represents a user and each column represents a movie. The values in this matrix correspond to the

Table 2: Data statistics after cold start, middle, heavy users division

	Split values		
	Cold start users	**Middle users**	**Heavy users**
Train set users	1425	1909	15
Test set users	379	874	13
Train set hotels	1297	1602	217
Test set hotels	351	694	26
Train set reviews	4009	10476	233
Test set reviews	421	1164	26

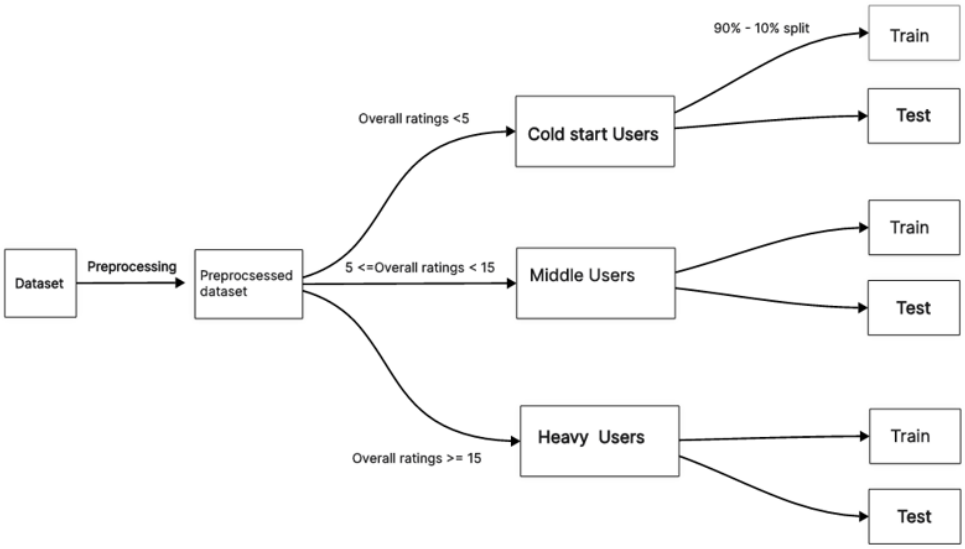

Figure 1: Illustrative representation of dataset splitting

overall rating given by each user to each movie.

After constructing the user-item matrix, we apply hot deck imputation using the K-nearest neighbors (KNN) imputer to handle missing values. Since the matrix can be very sparse, imputation helps to fill in missing values and improve the accuracy of our results. We use a KNN imputer with a parameter setting of K=3, meaning that for each missing value, we consider the three nearest neighbors with known ratings for that item and average their ratings to impute the missing value. This approach helps to maintain the integrity of the matrix and ensure that our imputed values are based on similar users' ratings.

6.1 Similarity calculation

6.1.1 Multi-dimensional Distance Metrics

We can calculate similarity using multidimensional Distance Metrics.

In our methodology, we aim to quantify the similarity between users based on their ratings for items across multiple criteria.

Each user's rating for an item is represented as a vector $\mathbf{R(u, i)} = \{r_0, r_1, r_2, ..., r_k\}$ where k denotes the number of criteria considered. first we will calculate distance between two users for their common rated item on all criteria $drating(R(u_1, i), R(u_2, i)$ The multi-dimensional distance(Euclidean [9], Manhattan, Chebyshev) can be used to calculate the distance and overall distance can be calculated using equation 30.1

$$d(u_1, u_2) = \left(\frac{1}{|I(u_1, u_2)|}\right)\left(\sum_{i \in I(u_1, u_2)} drating(R(u_1, i), R(u_2, i)\right) \tag{30.1}$$

where $i \in I(u_1, u_2)$ is list of items both u_1 and u_2 have rated

Distances and similarity are inversely proportional to each other.

Hence the distance between all the users is computed ($d(u_1, u_2)$) so and then the similarity between them is calculated using the following equation 30.2:

$$sim(u_1, u_2) = \frac{1}{1 + d(u_1, u_2)} \tag{30.2}$$

The constant '1' is added in the denominator for numerical stability (to avoid divide by zero error).

6.1.2 Pearson / Cosine similarity

In addition to the multidimensional similarity approach, we also use Pearson similarity and cosine similarity to compute the similarity between users' ratings.

We first calculate the similarity between each pair of users for each criterion to form a similarity matrix for each criterion. Then, we aggregate all the similarity matrices to find the overall or average similarity between users.

This average similarity is computed separately for all three groups: cold-start, middle, and heavy users. This approach allows us to consider different user groups and their rating patterns when determining the similarity between users.

6.2 Rating Prediction

To predict the rating that user u would assign to item i, we initiate by identifying all users (N_u) similar to user u from the average similarity matrix. However, our consideration is limited to users who have previously rated item i.

Subsequently, we compute a weighted average of the similarity between user u and these similar users, each weighted by the rating the similar user assigned to item i. This weighted average serves as our predicted rating.

In cases where there are no similar users to user u for item i, we resort to returning the mean rating as the prediction. This strategy ensures a fallback mechanism for situations where direct similarities are insufficient for prediction. Mathematically it is discussed via equation 30.3

$$\hat{r_{u,i}} = \frac{\sum_{a \in N_u} r_{a,i} \; X \; sim(a,u)}{\sum_{a \in N_u} sim(a,u)} \tag{30.3}$$

Where n_u is the list of neighbours who have rated item i.

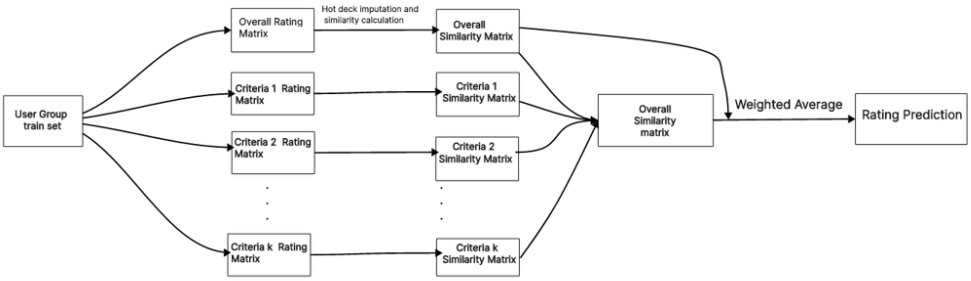

Figure 2: Process of rating prediction

7 Evaluation

For all user-item pairs in the test set, we predict the ratings using the method described earlier. These predicted ratings are then compared with the actual ratings in the test set to calculate the Root Mean Squared Error (RMSE). RMSE is a measure of the differences between predicted and actual values, providing a way to assess the accuracy of our recommendation algorithm. A lower RMSE indicates better performance, as it signifies that our predictions are closer to the actual ratings.

8 Results

In this section, we provide a detailed comparison of the performance of our proposed model with and without the hot deck imputation technique. The performance of the recommender system is evaluated using the Root Mean Squared Error (RMSE) metric,

which measures the differences between predicted and actual ratings. A lower RMSE indicates better performance.

Performance Analysis:
To assess the effectiveness of our proposed model, we compared the RMSE values for different user groups (cold-start, middle users, heavy users) using various similarity measures (Manhattan, Chebyshev, Euclidean, Pearson, Cosine). When hot deck imputation is applied, we compare the results with the previous implementation technique (without any amendments). This comparison allows us to assess the impact of hot deck imputation on the performance of our recommendation algorithm.

Table 3: Tabulated previous and updated results

	Previous results			Updated results		
	Cold start	Middle users	Heavy users	Cold start	Middle users	Heavy users
Manhattan	1.2537	1.0523	1.18695	1.0174	0.8966	1.1766
Chebyshev	1.2570	1.0510	1.18699	1.0187	0.8964	1.1766
Euclidean	1.2560	1.0512	1.18696	1.0180	0.8961	1.1766
Pearson	1.3400	1.0698	1.18717	1.0231	0.8986	1.1766
Average rating	1.4099	1.3206	1.2373	1.3926	1.3009	1.3008
Cosine	-	-	-	1.0232	0.8986	1.1766

The performance of our proposed model demonstrates significant improvements in prediction accuracy, as evidenced by the reduction in RMSE values across all user groups and similarity measures. The inclusion of hot deck imputation has been particularly beneficial, leading to a substantial decrease in RMSE for cold-start and middle users, which are traditionally challenging segments for recommender systems. This improvement underscores the effectiveness of our methodology in providing accurate and personalized recommendations, especially for users with sparse rating histories. The model's robust performance across different similarity measures further validates the versatility and reliability of our approach in handling various data scenarios.

9 Conclusions

Our observations suggest that combining hot deck imputation with multi-criteria similarity matrices and dividing users based on the number of ratings they have done leads to better results. This approach improves the accuracy and effectiveness of the recommendation algorithm compared to using a single criterion or no imputation methods or any other imputation methods. From our analysis, we have found that different similarity measures perform differently across user groups. Specifically, Pearson similarity yields the best results for cold start users, while Euclidean similarity is most effective for middle users. Interestingly, for heavy users, the choice of similarity measure appears to have less impact, as most similarity measures produce similar results. The accuracy

for heavy users is slightly lower than anticipated, which can be attributed to the limited number of data points available in our dataset. This limitation affects the algorithm's ability to be effectively trained and tested, resulting in slightly lower accuracy for heavy users.

References

1. Z. Wang, X. Yu, An improved collaborative movie recommendation system using computational intelligence. J. Visual Lang. Comput. 607–675 (2014)
2. Z. Tan, L. He, An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle. IEEE Access 27211–27228 (2017)
3. Sinha, B. B., & Dhanalakshmi, R. (2022). Evolution of recommender paradigm optimization over time. Journal of King Saud University-Computer and Information Sciences, 34(4), 1047-1059.
4. A. Agarwal, Similarity measures used in recommender systems: a study. Int. J. Eng. Technol. Sci. Res. 619–626 (2017)
5. M. Wasid, M. Ali, An improved recommender system based on multi-criteria clustering approach clustering approach. Procedia Comput. Sci. 93–101 (2018)
6. G. Adomavicius, New recommendation techniques for multi-criteria rating systems. IEEE Intell. Syst. 48–55 (2007)
7. B. Dai, A. Qu, Smooth neighborhood recommender systems. J. Mach. Learn. Res. 1–24 (2019)
8. L.M. De Campos, J.F. Huete, M. Fernández-luna, Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian networks. Int. J. Approximate Reasoning 785–799 (2010)
9. A. Gogna, A. Majumdar, A comprehensive recommender system model improving accuracy for both warm and cold start users. IEEE Access 2803–2813 (2015)
10. J. Antony, K. Akshay, Collaborative filtering using Euclidean distance in recommendation engine. Indian J. Sci. Technol. 37–42 (2016)
11. Valcarce D, Parapar J, Finding and analysing good neighbourhoods to improve collaborative filtering. Knowl. Based Syst. 193–202 (2018)
12. H. Chandrashekhar, B. Bhasker, Personalized recommender using entropy based collaborative filtering. J. Electron. Commer. Res. 214–237

13. Sinha, B. B., Dhanalakshmi, R., & Regmi, R. (2020). TimeFly algorithm: a novel behavior-inspired movie recommendation paradigm. Pattern Analysis and Applications, 23(4), 1727-1734.

14. Dhanalakshmi, R., & Sinha, B. B. (2019). Hybrid Cohort Rating Prediction Technique to leverage Recommender System.

15. Mikhaylov, A., Bhatti, I. M., Dinçer, H., & Yüksel, S. (2024). Integrated decision recommendation system using iteration-enhanced collaborative filtering, golden cut bipolar for analyzing the risk-based oil market spillovers. Computational Economics, 63(1), 305-338.

16. Klimashevskaia, A., Jannach, D., Elahi, M., & Trattner, C. (2024). A survey on popularity bias in recommender systems. User Modeling and User-Adapted Interaction, 1-58.