Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-20579-z

CFCAI: improving collaborative filtering for solving cold start issues with clustering technique in the recommender systems

Navid Khaledian[1] · Amin Nazari[2] · Masoud Barkhan[3]

Received: 1 April 2024 / Revised: 6 November 2024 / Accepted: 23 December 2024

Abstract

Recommendation systems are crucial in managing data overload, enabling online platforms to provide users with personalized recommendations. However, these systems often encounter significant challenges, such as the cold start problem and data sparsity, which hinder recommendation accuracy. To address these issues effectively, this study proposes an innovative approach that leverages implicit knowledge of users and items, structured into three primary stages. First, we employ clustering techniques to segment the user base, which reduces data volume and mitigates sparsity, enhancing the system's ability to deliver accurate recommendations. In the second phase, Association Rule Mining (ARM) analyses users' implicit interaction records, allowing us to derive valuable association rules and better understand user preferences. Finally, in the third stage, the system leverages these insights to suggest optimal items to each user, enhancing personalization. To validate the proposed technique, we conducted experiments using the Million Songs Dataset (MSD) within the LibRec 2.0.0 framework, offering a comprehensive analysis of its effectiveness. Comparative evaluations against recent state-of-the-art recommendation techniques, including GBPR, EALS, and User-Time K-NN, reveal that our approach consistently outperforms alternative methods in terms of Precision, Recall, and F-measure metrics, with performance improvements ranging from 0.5% to 5%. These findings underscore the approach's robustness in handling cold start and data sparsity challenges and its scalability potential for large-scale recommendation applications. This work presents a significant advancement in recommendation system methodologies, demonstrating the feasibility of combining clustering and ARM to enhance collaborative filtering techniques in diverse, sparse environments.

Keywords Recommender systems · Collaborative filtering · Data sparsity · Association rules mining · Clustering · Cold start

Extended author information available on the last page of the article

1 Introduction

Because of the Internet's explosive expansion, service and product providers must personalize the customer experience. Information retrieval specialists created the recommender systems (RSs) subclass to address this problem. RSs weed out irrelevant material and provide users with content recommendations based on their interests. RSs are used in various sectors to improve user experience and promote user involvement. To address the issue of too much information in our daily lives, RSs have been widely adopted [1, 2]. For instance, E-commerce recommendation systems [3] and tourism recommendation systems [4]. Online businesses and activities, including e-commerce, online news aggregators, and online video/music [5] sharing, are also eligible to utilize the recommendation system. This is due to the abundance of information available on the Internet that could be helpful to customers or prospective buyers, including online news, books, articles, music, movies, etc. [6]. RSs are typically categorized according to the items they recommend. Generally, three techniques are applied: Content-Based Filtering (CBF), Hybrid Filtering (HF) [7], and Collaborative Filtering (CF) [8]. CF suggests products that other users with similar tastes find appealing. Google[1] displays smart ads to users with recommender systems, e-commerce websites such as Amazon[2] manage data overhead with RSs, and Facebook[3] provides the correct name using a suitable recommendation algorithm when users upload photos and tag them [9]. Therefore, a recommender system offers users an alternative to find products they might not have found independently. CF, a recommender system technology that relies on explicit rating feedback from neighbours who share similar interests, is one of the more effective methods [10]. Based on user's previous interactions, the CF recommendation approach forecasts their preferences. CF is one of the most popular and effective techniques in a recommender system since it produces the greatest results and accurate recommendations while having an implicit methodology. This is why many online businesses and commercial systems use it to make client recommendations. Generally, CF recommendation approaches can be classified into two categories based on the type of input: explicit feedback-based rating-oriented CF algorithms [11] and implicit feedback-based ranking-oriented CF algorithms [12].

The explicit feedback (Fig. 3) directly obtains the user's preference data, e.g., user evaluation and selection. The CF recommendation approach suffers from the sparsity problem and the cold start problem [10, 13] because obtaining explicit feedback from new users takes a lot of work. Unlike detailed feedback, implicit feedback (Fig. 4) expresses users' preferences indirectly. Still, it makes it much easier to capture in many applications, e.g., purchasing behaviour and browsing history/time on e-commerce websites or the sightseeing route and viewing times in the tourist recommendation system. The implicit feedback can be used to facilitate the performance of CF-based approaches. Although the advantages of implicit feedback are apparent, it suffers from the One-Class Collaborative Filtering (OCCF) problem [14], where positive feedback is given, and all negative feedback is missing. Collaborative filtering systems face various challenges; Data sparsity and Cold start are two fundamental problems in RSs [6, 15–17]. When user ratings on items are incredibly sparse compared to the enormous number of users and items in the user-item matrix, CF results in poor recommendations (data sparsity problem). Data sparsity, which occurs when specific expected values are

[1] www.google.com

[2] www.amazon.com

[3] www.facebook.com

missing in the dataset, is a prevalent issue in large-scale data analysis. This situation arises in recommender systems when only some active users rate items. As a result, the accuracy of the recommendation is lowered. Several methods can mitigate this issue, including model-based collaborative techniques, demographic filtering, and singular value decomposition [15]. Implicit data is used to profile a user's item preferences without a user rating. Through observations of users' behaviour, implicit data can provide additional evidence and information about users' preferences. Table 1 illustrates a simple example: the sparsity of rated items matrix by users.

The cold-start problem is one of the general research issues in recommendation systems [13, 18–20]. The cold start problem occurs when insufficient data is available for the recommender system to make inferences. When a user is considered cold start, it means they have rated very few items, or none at all, and the system cannot generate practical recommendations for them. When a new user joins the system, when new items (or products) are added to the database or when launching a new system, the catalogue may contain many items but little user interaction and presence, making it difficult to provide reliable recommendations; it usually happens [15, 21, 22].

This article uses an Association Rules Mining (ARM) algorithm to solve these problems by identifying potential rules between users' interests. Extracting rules from real datasets can be time-consuming because ARM is an NP-hard problem [23]. Then, to reduce the data size, we propose a clustering solution. The ARM algorithm receives the transactions from each cluster once they are clustered. As a result, clustering overcomes data sparsity by reducing the amount of data. In addition to developing association rules between users' interests, improving the quality of recommendations for the target user is possible. Furthermore, the proposed method performs better when handling cold starts by utilizing implicit data (demographic information) about users and items.

Thus, the contributions and innovations of this research would be as follows:

- Clustering to Reduce Data Sparsity: This research introduces a clustering approach that enhances recommendation quality by grouping users and items, effectively addressing data volume and sparsity issues.
- Utilization of Implicit Interaction Records: The study leverages implicit interaction data to derive valuable association rules, thereby gaining deeper insights into user preferences and enhancing the personalization of recommendations.
- Enhanced Algorithm for Optimal Item Suggestions: The proposed algorithm utilizes the analyzed data to suggest optimal items, effectively tackling the challenges posed by cold start scenarios and data sparsity.
- Addressing the Sparsity Issue: By reducing the dimensionality of the rating matrix through clustering techniques, we improve the ability to identify customer similarities and interests, leading to more relevant recommendations.

Table 1 Sparse user-item rating matrix

	Item 1	Item 2	Item 3	Item 4
User 1	3	-	-	1
User 2	2	1	-	-
User 3	-	-	-	4
User 4	4	-	-	-

- Cold Start and Data Dispersion Resilience: The system employs demographic information and user profiles to make informed suggestions for new users or unique items, utilizing association rules to predict user preferences efficiently.
- Performance Evaluation: We conducted comprehensive evaluations on a song dataset, demonstrating performance improvements of 0.5% to 5% in F-Measure, Precision, and Recall metrics compared to recent recommendation approaches.
- Advancement of Collaborative Filtering Techniques: This research contributes to advancing collaborative filtering methodologies and developing more effective recommendation systems capable of operating in challenging environments.

The rest of the paper is as follows. In Section 2, we will review the research literature. In Section 3, the proposed method and its components are explained in detail, and in Section 4, the proposed method is evaluated on a known dataset of songs. Section 5 describes conclusions and future works.

2 Related work

A recommendation system allows users to discover products they may not have encountered independently [1]. Figure 1 shows an RS schema.

CF is a technology within recommender systems that depends on explicit rating feedback from like-minded neighbours with similar interests, making it among the more successful methods [10]. By predicting user preferences based on their past interactions, the CF recommendation approach proves to be both popular and effective. Despite its implicit methodology, CF stands out as one of the most widely used and accurate techniques in recommender systems, yielding superior results [23]. Figure 2 shows a collaborative filtering process.

From a perspective standpoint, CF recommendation approaches can typically be divided into two groups depending on the input type they utilize: CF algorithms centred on explicit feedback and ratings (Fig. 3) [11] and CF algorithms centred on implicit feedback and rankings (Fig. 4) [12].

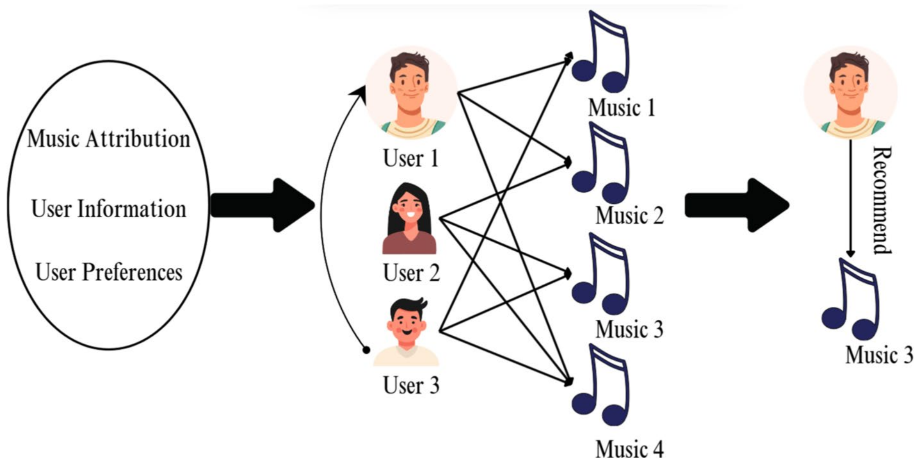


Fig. 1 Recommender system

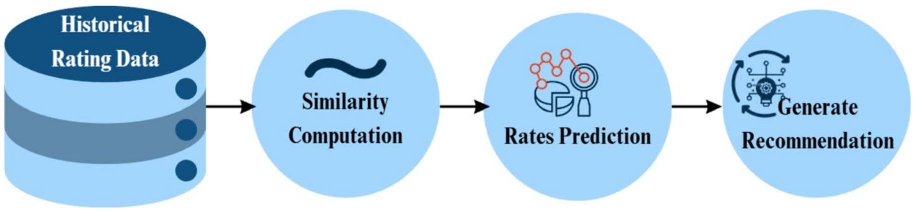
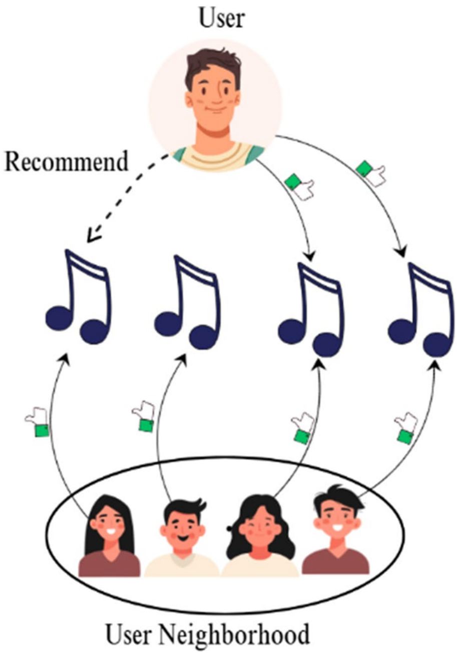


Fig. 2 A simple view of collaborative filtering architecture

Fig. 3 Explicit collaborative filtering



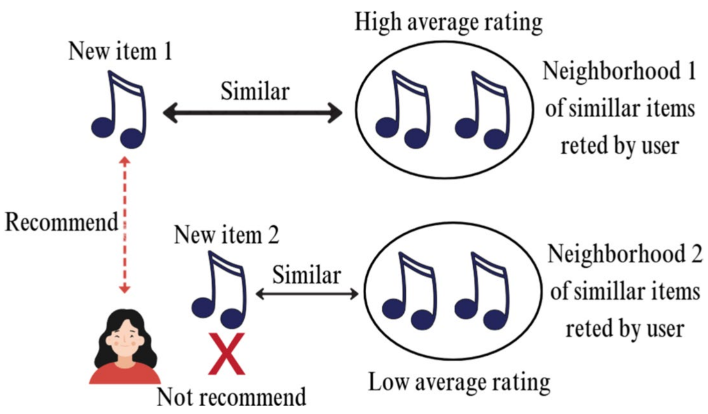


Fig. 4 Implicit collaborative filtering

Functionally, Collaborative filter methods are divided into memory-based and model-based methods [24–26]. Memory-based methods include two main phases: similarity calculation and prediction. The degree of proximity or similarity of user items is calculated in the similarity phase. In the prediction phase, the scores of the most similar neighbours to the target user are used to predict his possible rank of unobserved items [18, 23, 26]. Therefore, improving the similarity calculation function and the prediction algorithm are two critical issues in memory-based methods.

It should be noted that the main problem of memory-based methods is their high calculation cost. Lima G. et al. presented a method to reduce the computations in memory-based collaborative filtering by reducing the data dimensions to decrease the computations [27]. This method starts with representing the ranking matrix in the low-dimensional space. Then, the KNN algorithm identifies and predicts the closest users to the target user. Reducing data dimensions is a solution used in many research. Zare H. et al. [28] introduced a meta-path method to measure the similarity matrix that considers hidden user relationships. Each meta path has a special meaning in the two-part graph; by aggregating the extracted data between the paths, the similarity between users can be identified with high accuracy based on the meaning of each path.

In model-based methods, learning algorithms are used to build the model. These algorithms include various supervised and unsupervised machine-learning techniques. Supervised methods include neural networks and deep neural networks, which have recently received much attention due to the availability of appropriate hardware resources. Clustering methods and exploring association rules are also non-supervised methods used to build the model. After learning the model, it is used for recommendations in the following steps. The main advantage of these methods is that they can know and be used repeatedly to provide recommendations [29]. Tanwar A. et al. [30] use a deep neural network-based approach that uses user and item vectors to encapsulate user-item data and train on high-dimensional nonlinear data to provide more accurate recommendations. User-user networks are employed to bring greater collaboration and synergy to our model. In our approach, they combine user-user networks with deep neural networks, resulting in higher prediction accuracy and better runtime than other state-of-the-art methods. Bobadilla J et al. [31] applied the variational concept not to generate augmented sparse data but to create expanded samples in the latent space encoded in the dense inner layers of the proposed neural network. This approach attempts to combine the potential of variational stochastics with the augmentation concept. Augmented samples are generated in the dense latent space of the neural network model. According to their claims, the variation process avoided the sparse scenario.

Behera, G. and Nain, N. also [32] proposed a deep collaborative recommender system that embeds object metadata to combat nonlinearity and sparse data. The model consists of two stages. The first stage uses a neural network to capture the nonlinear properties of the data by embedding vectors. These vectors are concatenated and provided as input to the second stage of the model. The model output provides a partially observable estimate. Input data and parameters are simultaneously optimized and updated to minimize errors. Chen J et al. proposed an improved time-weighted collaborative filtering algorithm [33] that considers storage times and expiration characteristics. First, inspired by the half-life of information, they introduced a rounding-forgetting function to mine the user's interest rule. In this way, the information's retention periods can be considered. The new rating matrix contains more user interests and will continue to be used to implement recommendations. Second, a difference equation is applied to simulate the evolutionary process of changing user interests, and these equations are proven to be robust for analyzing user interest clusters. Finally, the ratings are predicted based on the newly found matrix, and a recommendation list is created based on the predicted ratings.

Yousefian Jazi et al. [5] proposed a music recommender system that suggests music based on keystrokes and mouse click patterns. Their proposed system directly maps these patterns to the user's favourite music without labelling their current emotions. Pichl, M. and Zangerle, E [34] introduced a multi-context-aware user model that utilizes both situational contexts derived from playlist names and playlist archetypes sharing acoustic characteristics to model the type of music jointly listened to in specific situational contexts. Both the situational context and musical preferences are represented as cluster assignments. For the computation of recommendations, we use Factorization Machines that use the proposed user model as input to exploit interaction effects between contexts. Through extensive offline experiments, we demonstrate that including situational context enhances the accuracy of music recommender systems, acoustic features, and, thus, a user's musical taste. A neural content-aware collaborative filtering was presented by Magron P. et al. [35], a unified framework that mitigates some limitations and develops the prior neural collaborative filtering into its content-aware equivalent. This model uses Deep learning to model user-song embedding interactions and content information extraction from low-level acoustic data. The deep content feature extractor produces two versions (strict and relaxed) of their model: one that predicts the item embedding directly and another that acts as a regularization prior. According to experimental findings, the suggested approach achieves state-of-the-art performance for both warm- and cold-start music recommendation tasks. Using the context of the group as well as the individual interests of the users, Adrián Valera et al. [36] designed a group recommender system to suggest music to groups of users. The cold-start problem represents a common research challenge in recommendation systems [19, 20]. Using the Linked Open Data (LOD) knowledge base, Natarajan S. et al. [20] resolved the cold start issue by collecting information about the new entity. Any computation techniques employed in previous systems and analyses showed that feature-dependent data help develop the effectiveness of recommender systems. They proposed a matrix analysis model of linked open data based on the similarity measure of linked available data. Tey F.J. et al. [21] used indirect connections between friends and "friends of friends" to solve the cold start problem for new users. If a friend does not have relevant data, the system will find the friend's friends to make the most accurate recommendations.

Gopal B. et al. proposed an MF model [37] that integrates user-side information to handle a few observations about an item, leading to sparsity issues of CF. They integrated

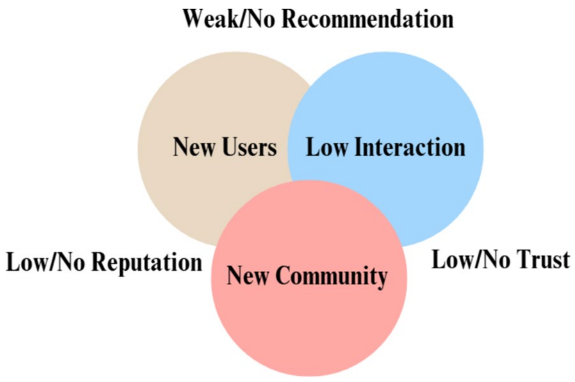


Fig. 5 Cold start problem

user-side information regarding vectors and bias to overcome the sparsity problem of collaborative filtering.

Taushif A. et al. in a paper [38], implemented a system to overcome cold start and sparsity issues by 1) Generating a user-item similarity matrix and prediction matrix by performing collaborative filtering using memory-based CF approaches including KNNBasic, KNNBaseline, KNNWithMeans, SVD, and SVD + +. 2) Generating a user-item similarity matrix and prediction matrix by performing collaborative filtering using a model-based CF approach, Co-Clustering. The results reveal that the CF implemented using the K-NNBaseline approach decreased the error rate when applied to movie datasets using cross-validation (Fig. 5).

Guan. J et al. worked on a new robust hybrid similarity model [39], the Wasserstein distance-based CF (WCF) model, to relieve the cold-start CF problem in sparse data. They measured the similarity via the Wasserstein distance [40], which may improve and circumvent the Bhattacharyya coefficient and KL divergence drawbacks.

Table 2 comprehensively compares various recommender systems mentioned in Section 2, detailing their reference numbers, titles, general methods, advantages, and weaknesses to highlight their unique contributions and limitations in addressing challenges like cold-start and data sparsity.

3 Proposed method

This section introduces the proposed framework, a collaborative filtering system, in three general phases, including pre-processing phases with clustering, user interest prediction using ARM, and recommending the items to the user. An overview of the proposed method is shown in Algorithm 1.

Table 2 A comprehensive comparison of various recommender systems

Ref. Num	Method	Advantages	Weaknesses
[27]	Collaborative Filtering	Enhances memory recall; improved accuracy in recommendations	Limited to specific user contexts
[32]		Addresses cold-start problems effectively	Potential overfitting with small datasets
[35]		Mitigates cold-start issues effectively	Requires extensive training data
[20]		Uses Linked Open Data to enhance recommendation quality	Complexity of integrating external data
[21]		Enhances recommendation accuracy through indirect relations	Requires careful modelling of relationships
[38]		Addresses cold start and sparsity issues effectively	struggles with real-time updates
[5]	Content-Based	Considers user emotions for tailored recommendations	not generalize well across different users
[34]		Adapts to different user contexts for improved relevance	Complexity in context modelling
[36]		Tailors recommendations for groups, enhancing user experience	Difficulty in accurately assessing group preferences
[31]	Deep Learning	Captures uncertainty in predictions; improved accuracy	Requires large datasets for practical training
[28]	Hybrid Model	Utilizes predictive networks for better recommendations	Complexity in model design
[30]		Combines user-user interactions with deep learning	High computational cost
[39]		Mitigates cold-start and sparsity issues using hybrid similarity	Potential computational overhead
[38]	KNN, Matrix Factorization	Cold Start and Sparsity Mitigation SVD and Co-Clustering also show good performance in reducing error rates (RMSE, MAE)	struggle with large-scale datasets due to the high computational costs involved in similarity calculations Cold Start Dependency
[37]	Matrix Factorization	Improves recommendations by integrating user info	Dependence on the quality of user data

Algorithm 1 The proposed method

```
Inputs:   Available listening background of users.
Outputs:  Recommend similar items from each cluster based on the threshold
Begin
    Phase I: Pre-processing
1:      Cluster items according to features
2:      Calculate the ratio of user opening counts for a specified cluster of items.
3:      Group the profile of user's tastes in items and cluster items as input for ARM.
    Phase II: Prediction
1:      Run the association algorithm and create rules according to users' history.
2:      Compute the similarity between users' interests in clustered items based on the overlap of the opened group
        items
3:      According to the user's previous preference list, find rules for their antecedent side (left side of the rule).
4:      Predict user preferences on clusters via extracting the consequent side (right side) of the rules; then, the
        resultant side of the regulations is refined, and just the rules with the higher preference level value between
        users (highest opening counts) are selected
    Phase III: Recommendation
1:      Calculate the similarities between pairs of songs using the information about the song's features, which are the
        artist, year, title, release, song popularity, artist_familiarity, duration, and tags to acquire similarity degrees
        between items.
2:      Compute the threshold value for specifying items similar to user preferences. Then, match the user's profile
        (history of the user) with item features to recommend items similar to those they have liked.
3:      Items with a higher similarity value than the calculated threshold are selected for recommendation to the active
        user.
End
```

3.1 Phase I: data pre-processing with clustering

At this stage, the items are clustered based on their demographic characteristics, and then an item performance rate from each cluster is calculated for each user. Then, based on the obtained execution rate, the user is assigned to a cluster or a set of clusters. To make the method more understandable, we continue the methodology with the sample dataset of the songs. The overview of the first phase is shown in Fig. 6.

Figure 7 shows the tree structure of the clusters so that the songs are placed in separate clusters based on tag and duration attributes.

The song tags are on the first level (Level 1). Each tag contains five categories for song duration. The song's duration can be concise, short, medium, long, or very long, which exists in the second level (Level 2) of the clustering structure. That is, each song is placed in a cluster based on the tag and duration of that song. With this clustering, the items with the most similarity in tag and duration are placed in a cluster.

After clustering, an item's performance rate from each cluster is calculated for each user, and the user is assigned to item clusters with the highest item's performance rate for that cluster. In the example, with this method, we combined the user profile with the

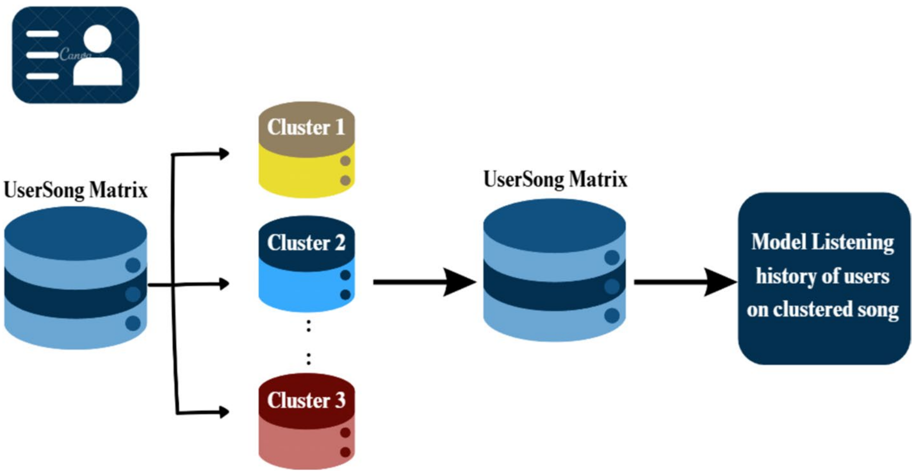


Fig. 6 Data pre-processing with clustering for the song's dataset

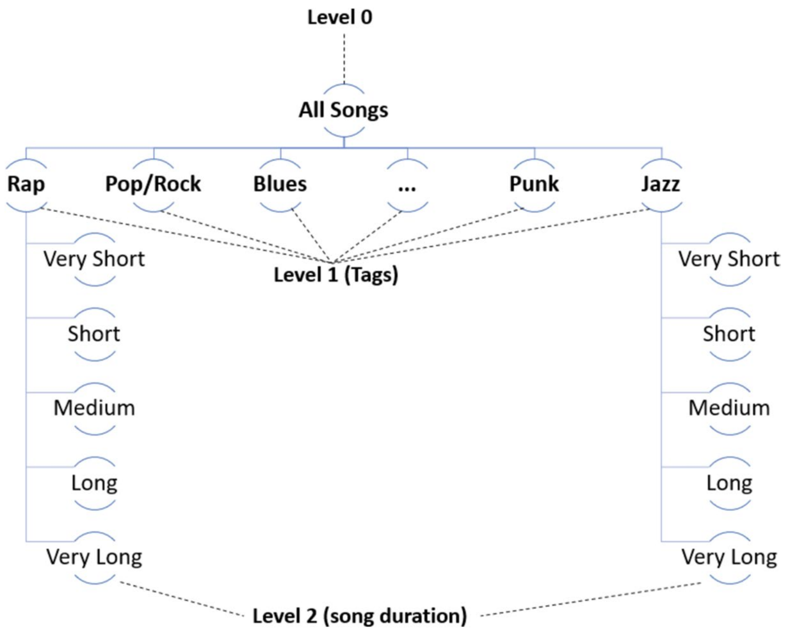


Fig. 7 Overview of songs' clustering

song cluster, and every user placed in each cluster was most interested in the songs of that cluster. On the other hand, different levels of belonging to clusters are considered for each user. Therefore, a user may belong to a cluster with a high membership level and others with a medium or low membership level. The membership level amount for each user can be calculated using Eq. (1).

Table 3 Levelling of users belonging to clusters

The level of users belonging to a cluster	%10—%30	%30—%60	%60- %100
Cluster membership level	Low	Medium	High

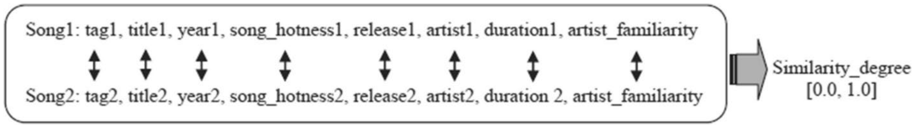

Song1: tag1, title1, year1, song_hotness1, release1, artist1, duration1, artist_familiarity
Song2: tag2, title2, year2, song_hotness2, release2, artist2, duration 2, artist_familiarity
Similarity_degree
[0.0, 1.0]

Fig. 8 Calculation of the degree of similarity between two songs in the used data set

$$U_l = \frac{S_p}{max_s} \tag{1}$$

U_l is the level of the user belonging to a cluster; S_p is the number of songs the user has played in that cluster; max_s is the maximum number of songs in the user's profile.

The levelling of users who are part of clusters is described in Table 3. Ultimately, the number of clusters and the level to which a user belongs are determined based on their profile.

3.2 Phase II: association rules mining

In the second phase, the clusters generated in the first phase are used as input to discover association rules between clusters. The objective of this phase is to identify clusters that are similar to the target user's preferences. The FP-growth algorithm [41], a well-known and efficient association rule mining (ARM) method, is employed to generate these rules. The association rule generator extracts rules from the clusters and then selects rules that align with the active user's listening history. Finally, the system predicts the user's song preferences within the identified clusters.

3.3 Phase III: recommending the best items to the user

In the third phase, the system aims to provide personalized recommendations to the user. The outputs from the second phase, a series of clusters closely related to the user's cluster based on association rules, are utilized. A set of items from these similar clusters is selected to align with the user's profile. The recommendation process involves calculating similarities between items based on their features. A subset of items similar to those already in the user's profile is identified. Finally, similar items from each cluster are recommended, subject to a defined threshold.

As illustrated in Fig. 8, the similarity between the items is calculated based on their feature vectors, such as (tag, publication year, singer name, duration, title, etc.) after determining the clusters. These features should match the most in the two songs so that the two songs are the most similar to each other. Figure 8 calculates the degree of similarity between two songs in the used data set.

As shown in Fig. 8, by adjusting the feature vectors obtained from each song, a degree of similarity between each pair of desired songs can be found, a numerical similarity in the range of [0,1], and a higher degree of similarity would indicate higher similarity. For example, if 6 out of 8 features match, the degree of similarity of the two songs is 6/8 = 0.75. After determining the similarity, the items with the most significant similarity are suggested to the target user according to the following relationship.

$$R(u) = \frac{\sum_{S_i \in P_u} Sim\left(S, S_i\right)}{\left|P_u\right|} \tag{2}$$

In Eq. (2), S are the songs inside the pre-built clusters, and S_i are songs the target user u has already listened to. Also, $\left|P_u\right|$ is the number of songs the target listener has heard. Therefore, the degree of similarity between the songs is obtained with the user's profile (songs he has already listened to). $Sim\left(s, s_i\right)$ is the calculated similarity between two items, based on matching the characteristics of the items as shown in Fig. 9. As illustrated in Fig. 9, a comparison of the metrics of precision, recall, and F-measure is presented by various methods. Figure 9a illustrates the comparison of the precision metric, Fig. 9b illustrates the comparison of the recall metric, and Fig. 9c illustrates the comparison of the F-measure metric. Moreover, any song from the clusters most similar to one of the user's profile songs will be suitable to recommend to the active user.

4 Experiments

This section presents the experiment's results on the Million Songs Dataset (MSD) [42] to improve recommendation performance. Our primary objective is to compare the behaviour of the proposed technique, implemented in Java using the LibRec 2.0.0 framework, on the recommended music list against traditional Collaborative Filtering (CF) under sparse data conditions. Additionally, we present the experimental results of several recent music recommendation techniques implemented in Java using LibRec 2.0.0 for comparative analysis. LibRec 2.0.0 provides a robust foundation for developing recommendation systems, incorporating many well-known methods. By leveraging this framework, we efficiently implemented and evaluated our proposed technique and other state-of-the-art approaches. This dataset comprises various components, including song, song text, song tag, song duration, similarity between songs, song singer, etc. It is organized into multiple subsections. We conducted our experiments and obtained our findings using a portion of this dataset. Table 4 displays the general attributes of the data set.

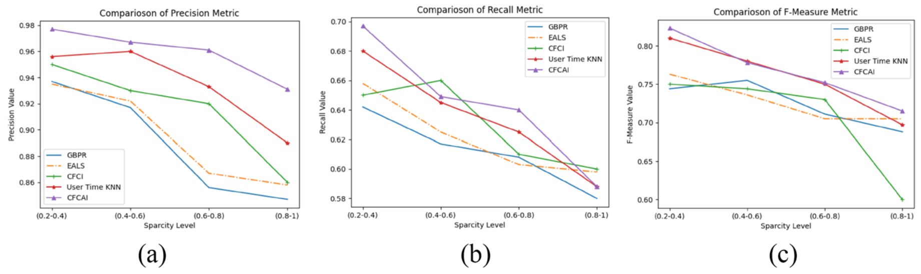


Fig. 9 Comparison of the metrics of Precision, Recall and F-Measure by different methods

Table 4 Characteristics of the dataset

Dataset	Number of items	Number of users	Ratings
MSD	20,277	1,508	10,192,175

To evaluate the approaches, we conducted tests on a standard laptop equipped with 32 GB of RAM and an Intel$^{(R)}$ Core$^{(TM)}$ i9-12900H Processor (24M Cache, up to 5.00 GHz). The operating system used was Windows 10 Enterprise N 64-bit.

4.1 Evaluation criteria

All experiments were conducted based on the Top-N strategy and Precision, Recall, and F-measure evaluation criteria [43–45]. We determine the accuracy of our recommendations by using the following formula: True Positives (TP) are items that users find appealing and are recommended to them. At the same time, False Positives (FP) are not attractive to users but are still recommended. False Negatives (FN) are items that users would enjoy but are not suggested, and True Negatives (TN) are not attractive to users and are not suggested.

The Recall metric tells us about the recommender's ability to recommend all relevant items. At the same time, precision determines the recommender system's ability to indicate only relevant items within a combination of irrelevant and appropriate items. In other words, precision calculates the ratio of suggested pertinent items, whereas Recall calculates the percentage of relevant recommended items. The basis for these measurements is the quantity of relevant or irrelevant elements, recommended or not recommended.

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP+FP} \tag{3}$$

Also, the recall criterion is calculated as follows:

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP+FN} \tag{4}$$

Generally, the two introduced criteria work divergently. This means that reducing Recall increases accuracy. As a result, the F-measure was introduced to find the appropriate state between the above two criteria.

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

The F-measure aims to determine the best possible trade-off between the Precision and Recall values. This is done by merging the two values into a single measure and giving each of them the same weight [38, 40, 41].

4.2 Practical results

By conducting tests on the dataset (MSD), the performance of the proposed algorithm was evaluated in comparison with other methods using the introduced evaluation criteria. The results of these performances are shown below. The best performance for each criterion is highlighted and bolded. After pre-processing the data in MSD, we conducted four evaluation experiments by constructing datasets with different Spartan levels to check the accuracy of quality recommendations in the proposed recommendation system against other methods. Users and items were selected based on the Spartan level in each evaluation experiment. The reason for conducting experiments at different spartan levels is that the proposed method should be evaluated for datasets with more changes in spartan levels. Hence, each data set was randomly combined into a training set containing 80% of the preference list per user and a test set containing the remaining preferences from the user. The reliability coefficient of the data is calculated based on Eq. (6), which can create different Spartan levels for different users and items.

$$sparsity\ measure = 1 - \frac{nr}{Nusers * Nitems} \tag{6}$$

To show the effectiveness of the proposed method, our method has been compared with several famous and efficient methods, including GBPR [46] and EALS [47], then a recent K-NN-based method [48] called User-Time K-NN and a method proposed by Najafabadi et al. [49] that performed better than Basic CF (CFCI) and CFMT [23]. The results for different sparsity levels are compared in Table 5. This is expected since a higher sparsity in the dataset leads to a worse outcome in the collaborative filtering methods. Therefore, the experimental results show that our method performs better at different and even higher sparsity levels than other methods. The results have been improved at the sparsity level, with a factor of more than 0.8. The experimental results in Table 5 show that our proposed solution performs better in offering recommendations to the user. The effectiveness of our method's recommendations is due to using a clustering strategy to determine the degree of users' preferences that characterize certain music features (tags and duration) and using association rules to find similar interest patterns amongst users.

The visualized experiments in Fig. 9 illustrate that the proposed approach performs better in most cases.

Table 5 Comparing the results of different methods with the proposed method on the MSD dataset

Sparsity Level		Methods					
	Metrics	GBPR	EALS	CFCI	User-Time K-NN	CFMT	CFCAI
(0.2–0.4)	Precision	0.937	0.935	0.95	0.956	0.96	**0.977**
	Recall	0.642	0.658	0.65	0.680	0.64	**0.697**
	F-measure	0.744	0.763	0.75	0.810	0.76	**0.823**
(0.4–0.6)	Precision	0.917	0.922	0.930	0.960	0.93	**0.967**
	Recall	0.617	0.625	**0.660**	0.645	0.64	0.649
	F-measure	0.755	0.736	0.744	**0.780**	0.75	0.778
(0.6–0.8)	Precision	0.856	0.867	0.92	0.933	0.95	**0.961**
	Recall	0.608	0.603	0.610	0.625	0.62	**0.640**
	F-measure	0.711	0.705	0.730	0.750	0.75	**0.752**
(0.8–1)	Precision	0.847	0.858	0.860	0.890	0.89	**0.931**
	Recall	0.580	0.598	**0.600**	0.588	0.584	0.588
	F-measure	0.688	0.705	0.680	0.697	0.71	**0.715**

Table 6 Comparing the results of different methods in cold start mode

Dataset		Methods					
	Metrics	GBPR	EALS	CFCI	User-Time K-NN	CFMT	CFCAI
MSD	Precision	0.747	0.808	0.79	0.796	0.813	**0.826**
	Recall	0.570	0.598	0.592	0.601	0.614	**0.627**
	F-measure	0.688	0.665	0.67	0.691	0.701	**0.719**

Table 7 Comparing based on accuracy in recommending the number of specific items in the MSD dataset

View of Recommend	Methods						
	Metrics	GBP	EALS	CFCI	User-Time K-NN	CFMT	CFCAI
Top_5	Precision	0.927	0.935	0.936	0.935	0.942	**0.977**
	Recall	0.638	0.618	0.652	0.671	0.684	**0.698**
	F-measure	0.744	0.763	0.772	0.791	0.792	**0.798**
Top_10	Precision	0.907	0.911	0.935	0.95	0.951	**0.968**
	Recall	0.610	0.614	0.628	0.666	0.7	**0.689**
	F-measure	0.735	0.736	0.722	**0.747**	0.725	0.735
Top_20	Precision	0.832	0.888	0.910	0.93	0.923	**0.942**
	Recall	0.602	0.588	0.590	0.615	0.601	**0.671**
	F-measure	0.711	**0.735**	0.720	0.711	0.705	0.728
Top_30	Precision	0.827	0.832	0.866	0.906	0.914	**0.932**
	Recall	0.568	0.528	0.554	0.570	0.609	**0.612**
	F-measure	0.614	0.712	0.600	0.695	0.699	**0.718**

4.3 Comparison of different methods in cold start mode

New items and users cause an essential challenge in recommender systems. As aforementioned, an item can only be recommended due to a cold start issue if many users have rated it. The first category of these problems occurs in collaborative filtering systems, where a new user enters the system, and the system does not know the user's preferences. In addition, a new user must have rated a suitable number of items so that the recommender algorithm can provide reliable and accurate recommendations. Therefore, by exploiting and using the implicit information of users and items, our proposed method can be resistant against cold start users, and the results of Table 6 show a comparison of the mentioned methods on cold start users. The proposed method produced a better response than the other methods. The proposed method can solve this critical problem by gaining better results and using the implicit information of users and items. Table 6. presents the result of target metrics on cold-start evaluation.

4.4 Comparison of different methods in different recommendations

In this section, we compare the proposed method with different methods based on accuracy in recommending the number of specific items. As shown in Table 7 and Fig. 10, the proposed method provided better results for a few and numerous recommended items in most cases. As

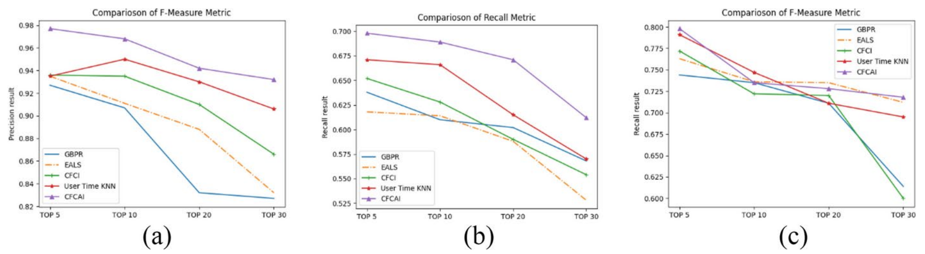


Fig. 10 Comparison of the metrics of Precision, Recall and F-Measure on Top-N items

illustrated in Fig. 10, a comparison of the metrics Precision, Recall, and F-Measure is presented on Top-N items. Figure 10a and c present the comparison of the Precision, Recall, and F-Measure metrics, respectively. Other methods also produce relatively good results, but their performance is usually similar to the proposed method. The proposed method uses association rules to discover user similarities and can get the ranks accurately. Therefore, when recommending different ranges of items, the accuracy of the proposed method is more stable.

4.5 Scalability of the proposed method

We inspected the scalability of the proposed method in terms of training time while using different percentages of the training dataset. Specifically, we changed the percentage of training data in the dataset from 0.1 to 1 in increments of 0.1. The results show that the training time increases linearly with the amount of training data. Hence, our approach can be applied to large-scale datasets. The proposed method also has fair scalability due to its low memory consumption. We also evaluated the scalability of the proposed method with different sets of training data. Here, we implemented the proposed method on the MSD dataset and measured the corresponding time cost of one iteration in the training process. Figure 11 demonstrates that the time consumption of our method is linear concerning the size of the training data. Our model can be considered linearly with the size of the training set. These experimental results and time complexity analysis show that our proposed method can provide suitable scalability for real applications.

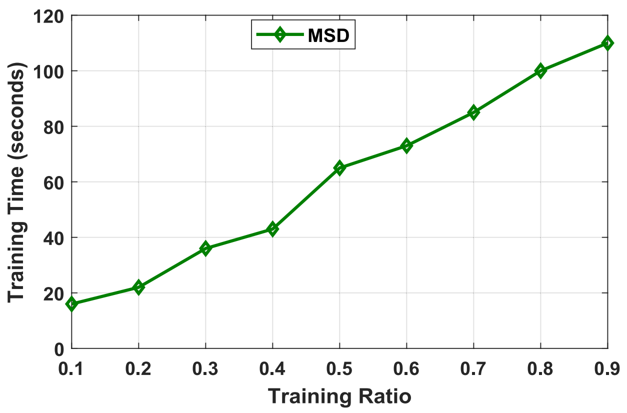


Fig. 11 Scalability of the proposed method on the MSD dataset

5 Conclusion and future direction

This paper introduced an RS framework designed to address the challenges posed by the explosive growth of the Internet and the overwhelming amount of information available to users. The proposed approach leverages CF techniques, explicitly focusing on mitigating the sparsity and cold start problems inherent in RSS. The study demonstrated that traditional CF methods face challenges, mainly when dealing with sparse data and new users/items. The proposed solution, incorporating Association Rules Mining (ARM) and a clustering strategy, aims to overcome these challenges. The clustering phase groups items based on demographic characteristics, and the ARM phase establishes rules between user clusters. This information is then utilized to enhance user recommendations, especially those facing the cold start problem. The experimental results, conducted on the Million Songs Dataset (MSD), revealed the effectiveness of the proposed method.

Regarding Precision, Recall, and F-measure, it outperformed other state-of-the-art techniques, including GBPR, EALS, CFCI, and User-Time K-NN. Additionally, the proposed method demonstrated resilience in handling cold start issues and showcased stability in recommending different numbers of items. Furthermore, the paper discussed the scalability of the proposed method, showing its linear relationship with training data size. This indicates its potential applicability to large-scale datasets, making it a promising solution for real-world scenarios.

In summary, the presented recommender system framework offers a robust approach to improving recommendation accuracy, addressing data sparsity, and handling cold start challenges. The positive experimental results underscore its viability for practical implementation in diverse domains where personalized content recommendations are essential. In the prospective evolution of the proposed approach:

- Numerous association rule mining algorithms exhibit expeditious execution but consume substantial memory resources. Enhancing the efficiency of rule mining algorithms can alleviate the runtime performance of RS.
- A noteworthy research challenge lies in the exploration of leveraging tree structures, graphs, and hyper-graphs to address the cold-start problem inherent in RS.
- Employing machine learning, pervasive across various scientific domains, allows gathering information preceding user activities within RSs, facilitating the expeditious creation of a database for prompt recommendation generation.

Acknowledgements We gratefully acknowledge Fatemeh Poormohammad for their assistance in data analysis, programming, and manuscript editing.

Author contributions Navid Khaledian: Supervisor, Methodology, Conceptualization, Investigation, Writing- Original draft preparation. Amin Nazari: Conceptualization, Methodology. Masoud Barkhan: Software, Visualization. Fatemeh Poormohammad: Data curation, Writing- Reviewing and Editing, Validation.

Funding No funding was obtained for this study.

Data availability The dataset used and analyzed during the current study is available from the corresponding author upon reasonable request.

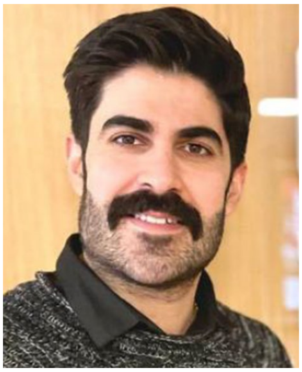
Declarations

Conflict of interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Zhou W, Haq AU, Qiu L, Akbar J (2024) Multi-view social recommendation via matrix factorization with sub-linear convergence rate. Expert Syst Appl 237:121687
2. Zangerle E, Bauer C (2022) Evaluating recommender systems: survey and framework. ACM Comput Surv 55(8):1–38
3. Islek I, Oguducu SG (2022) A hierarchical recommendation system for E-commerce using online user reviews. Electron Commer Res Appl 52:101131
4. Esmaeili L, Mardani S, Golpayegani SAH, Madar ZZ (2020) A novel tourism recommender system in the context of social commerce. Expert Syst Appl 149:113301
5. YousefianJazi S, Kaedi M, Fatemi A (2021) An emotion-aware music recommender system: bridging the user's interaction and music recommendation. Multimed Tools Appl 80(9):13559–13574
6. Yuan H, Hernandez AA (2023) User Cold Start Problem in Recommendation Systems: A Systematic Review. IEEE Access 11:136958–136977
7. Forouzandeh S, Rostami M, Berahmand K (2022) A Hybrid Method for Recommendation Systems based on Tourism with an Evolutionary Algorithm and Topsis Model. Fuzzy Inf Eng 14(1):26–50
8. Lee J et al (2019) l -Injection: Toward Effective Collaborative Filtering Using Uninteresting Items. IEEE Trans Knowl Data Eng 31(1):3–16
9. Parvin H, Moradi P, Esmaeili S (2019) TCFACO: Trust-aware collaborative filtering method based on ant colony optimization. Expert Syst Appl 118:152–168
10. Zhang Y et al (2020) Joint Personalized Markov Chains with Social Network Embedding for cold-start recommendation. Neurocomputing 386:208–220
11. Li G, Chen Y, Zhang Z, Zhong J, Chen Q (2018) Social personalized ranking with both the explicit and implicit influence of user trust and of item ratings. Eng Appl Artif Intell 67:283–295
12. Feng J, Wang K, Miao Q, Xi Y, Xia Z (2023) Personalized recommendation with hybrid feedback by refining implicit data. Expert Syst Appl 232:120855
13. Kawai M, Sato H, Shiohama T (2022) Topic model-based recommender systems and their applications to cold-start problems. Expert Syst Appl 202:117129
14. Alhijawi B, Al-Naymat G, Obeid N, Awajan A (2021) Novel predictive model to improve the accuracy of collaborative filtering recommender systems. Inf Syst 96:101670
15. Roy D, Dutta M (2022) A systematic review and research perspective on recommender systems. J Big Data 9(1):59
16. Ahmadian S, Joorabloo N, Jalili M, Ahmadian M (2022) Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach. Expert Syst Appl 187:115849
17. Forouzandeh S, Berahmand K, Rostami M (2021) Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens. Multimed Tools Appl 80(5):7805–7832
18. Khaledian N et al (2023) TrustDL: Use of trust-based dictionary learning to facilitate recommendation in social networks. Expert Syst Appl 228:120487
19. Bi Y, et al (2020) DCDIR: A Deep Cross-Domain Recommendation System for Cold Start Users in Insurance Domain, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery: Virtual Event, China, p 1661–1664
20. Natarajan S, Vairavasundaram S, Natarajan S, Gandomi AH (2020) Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data. Expert Syst Appl 149:113248
21. Tey FJ, Wu T-Y, Lin C-L, Chen J-L (2021) Accuracy improvements for cold-start recommendation problem using indirect relations in social networks. J Big Data 8(1):98
22. Wang H (2023) PowerMat: context-aware recommender system without user item rating values that solves the cold-start problem. https://doi.org/10.1117/12.2673085
23. Khaledian N, Mardukhi F (2022) CFMT: a collaborative filtering approach based on the non-negative matrix factorization technique and trust relationships. J Ambient Intell Humaniz Comput 13(5):2667–2683
24. Behera G, Nain N (2023) Collaborative Filtering with Temporal Features for Movie Recommendation System. Procedia Comput Sci 218:1366–1373
25. Behera G, Nain N (2022) Trade-off between memory and model-based collaborative filtering recommender system. In: Proceedings of the International conference on paradigms of communication, computing and data sciences. Singapore: Springer Singapore. https://doi.org/10.1007/978-981-16-5747-4_12
26. Torkashvand A, Jameii SM, Reza A (2023) Deep learning-based collaborative filtering recommender systems: a comprehensive and systematic review. Neural Comput Appl 35(35):24783–24827
27. Lima GR, Mello CE, Lyra A, Zimbrao G (2020) Applying landmarks to enhance memory-based collaborative filtering. Inf Sci 513:412–428

28. Zare H, Nikooie Pour MA, Moradi P (2019) Enhanced recommender system using predictive network approach. Physica A: Stat Mech Appl 520:322–337
29 Kiran R, Kumar P, Bhasker B (2020) DNNRec: A novel deep learning based hybrid recommender system. Expert Syst Appl 144:113054
30 Tanwar A, Vishwakarma DK (2023) A deep neural network-based hybrid recommender system with user-user networks. Multimed Tools Appl 82(10):15613–15633
31. Bobadilla J, Ortega F, Gutiérrez A, González-Prieto Á (2023) Deep variational models for collaborative filtering-based recommender systems. Neural Comput Appl 35(10):7817–7831
32. Behera G, Nain N (2022) Handling data sparsity via item metadata embedding into deep collaborative recommender system. J King Saud Univ - Comput Inf Sci 34(10, Part B):9953–9963
33. Chen J, Wang B, Liji U, Ouyang Z (2019) Personal recommender system based on user interest community in the social network model. Physica A: Stat Mech Appl 526:20961
34. Pichl M, Zangerle E (2021) User models for multi-context-aware music recommendation. Multimed Tools Appl 80(15):22509–22531
35. Magron P, Févotte C (2022) Neural content-aware collaborative filtering for cold-start music recommendation. Data Min Knowl Disc 36(5):1971–2005
36 Valera A, Lozano Murciego Á, Moreno-García MN (2021) Context-Aware Music Recommender Systems for Groups: A Comparative Study. Information 12:506. https://doi.org/10.3390/info12120506
37. Behera G, Nain N, Soni RK (2024) Integrating user-side information into matrix factorization to address data sparsity of collaborative filtering. Multimedia Syst 30(2):64
38. Anwar T, Uma V, Hussain MI, Pantula M (2022) Collaborative filtering and kNN based recommendation to overcome cold start and sparsity issues: A comparative analysis. Multimed Tools Appl 81(25):35693–35711
39. Guan J, Bilian C, Yu S (2024) A hybrid similarity model for mitigating the cold-start problem of collaborative filtering in sparse data. Expert Syst Appl 249:123700
40. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein Generative Adversarial Networks. In: Doina P, Yee Whye T (eds) Proceedings of the 34th International Conference on Machine Learning. PMLR: Proceedings of Machine Learning Research. p 214-223
41. Grahne G, Zhu J (2005) Fast algorithms for frequent itemset mining using FP-trees. IEEE Trans Knowl Data Eng 17(10):1347–1362
42. Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P (2011) The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)
43 Shinde SK, Kulkarni U (2012) Hybrid personalized recommender system using centering-bunching based clustering algorithm. Expert Syst Appl 39(1):1381–1387
44 Zangerle E, Bauer C (2022) Evaluating Recommender Systems: Survey and Framework. ACM Comput Surv 55(8):170
45 Coscrato V, Bridge D (2023) Estimating and Evaluating the Uncertainty of Rating Predictions and Top-n Recommendations in Recommender Systems. ACM Trans Recomm Syst 1(2):7
46. Pan W, Chen L (2013) GBPR: group preference based Bayesian personalized ranking for one-class collaborative filtering. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. AAAI Press: Beijing, China, p 2691–2697
47. He X, Zhang H, Kan M-Y, Chua T-S (2016) Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. Association for Computing Machinery: Pisa, Italy, p 549–558
48. Sánchez-Moreno D, Zheng Y, Moreno-García MN (2020) Time-Aware Music Recommender Systems: Modeling the Evolution of Implicit User Preferences and User Listening Habits in A Collaborative Filtering Approach. Appl Sci 10(15):5324
49. Najafabadi MK, Mahrin MN, Chuprat S, Sarkan HM (2017) Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. Comput Human Behav 67:113–128



Navid Khaledian received his Ph.D. in Artificial Intelligence from the Islamic Azad University, Sanandaj branch, Iran, in 2023. He is currently a postdoctoral researcher at the University of Luxembourg. His research interests include Artificial Intelligence (AI), Distributed Systems, and Internet of Things (IoT), focusing on task scheduling, recommender systems, and data mining.

Amin Nazari received his BSc degree in Computer Software Engineering from Islamic Azad University in Hamedan in 2009, followed by an MSc degree in Computer Software Engineering from Arak University in 2015. Currently, he is a Ph.D. candidate in artificial intelligence at Bu-Ali Sina University in Hamedan. His research interests include Wireless Sensor Networks, Internet of Things, IoT fog networks, and recommender systems.

Masoud Barkhan received his M.Sc. in Software Engineering from the Islamic Azad University, Sanandaj, Iran, in 2017. His research interests include AI, Data mining and blockchain technology.

Authors and Affiliations

Navid Khaledian[1] · **Amin Nazari**[2] · **Masoud Barkhan**[3]

✉ Navid Khaledian
navid.khaledian@uni.lu

Amin Nazari
a.nazari@eng.basu.ac.ir

Masoud Barkhan
mbarkhan@mapc.ir

1 Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Esch-Sur-Alzette, Luxembourg

2 Department of Computer Engineering, Bu-Ali Sina University, Hamedan, Iran

3 Mahabad Petrochemical Company, Mahabad, Iran