An efficient hybrid recommendation model based on collaborative filtering recommender systems

Mohammed Fadhel Aljunid | Manjaiah Doddaghatta Huchaiah

Department of Computer Science, Mangalore University, Mangalore, India

Correspondence
Mohammed Fadhel Aljunid, Department of Computer Science, Mangalore University, Mangalore, India.
Email: ngm505@yahoo.com

Abstract
In recent years, collaborative filtering (CF) techniques have become one of the most popularly used techniques for providing personalized services to users. CF techniques collect users’ previous information about items such as books, music, movies, ideas, and so on. Memory-based models are generally referred to as similarity-based CF models, which are one of the most widely agreeable approaches for providing service recommendations. The memory-based approach includes user-based CF (UCF) and item-based CF (ICF) algorithms. The UCF model recommends items by finding similar users, while the ICF model recommends items by finding similar items based on the user-item rating matrix. However, consequent to the ingrained sparsity of the user-item rating matrix, a large number of ratings are missing. This results in the availability of only a few ratings to make predictions for the unknown ratings. The result is the poor prediction quality of the CF model. A model to find the best algorithm is provided here, which gives the most accurate recommendation based on different similarity metrics. Here a hybrid recommendation model, namely \( \Gamma \)UCF, is proposed. The \( \Gamma \)UCF model integrates the UCF and ICF models with the \( \Gamma \) linear regression model to model the sparsity and scalability issue of the user-item rating matrix. Detailed experimentation on two different real-world datasets shows that the proposed model demonstrates substantial performance when compared with the existing methods.

1 INTRODUCTION
In recent times, recommender systems (RS) have become one of the important areas that commercial companies and researchers direct towards due to their usefulness in our daily lives. The RS is a subclass of information filtering systems that seeks to predict the “rating” or “preference” a user would give to an item. They are primarily used in commercial applications. In other words, RS mainly refers to the application of collaborative intelligence to make recommendations. RS can effectively solve the problem of information overload. The recommendation system provides users with a sorted personalized item recommendation list based on the user’s historical preferences and constraints. A more accurate recommendation system can enhance and improve user experience [1, 2]. Recommendation results can usually be generated based on user preferences, product characteristics, user-commodity transactions, and other environmental factors (such as time, season, location, etc.). The recommended items may include music [3], movies [4, 5], news items [6], books [7], research articles [8], and so on. The literature related to RS divides them into three categories of algorithm filtering, namely collaborative filtering-based (CF) RSs, content-based RSs, and hybrid-based RSs [9, 10]. CF-based RS use only the interactive information between users and items to generate recommendations. Content-based RS utilise a series of discrete, pre-tagged characteristics of an item to recommend additional items with similar properties. The hybrid recommendation models use interactive information, metadata of users, and items. In this work, we focus on CF algorithms which are considered the most popular successful algorithms in RS [11]. The content-based RS is also a very old recommendation algorithm. But even today, the CF algorithm is the most widely used recommendation system occupying an extremely important position in the field of recommendation [12], and it was even once synonymous with recommendation systems. The

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. CAAI Transactions on Intelligence Technology published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology and Chongqing University of Technology.
basic idea of the CF algorithm is to make recommendations based on all the interactive behaviours of the user/item using collective wisdom.

There are two main types of CF algorithms, namely memory-based CF models and model-based CF algorithms. The memory-based model uses the historical behaviour data to find a possible association between a user and an item to make recommendations [13]. Memory-based CF can be divided into two sub-categories, namely, user-based CF (UCF) and item-based CF (ICF). Several memory-based CF models have been proposed either UCF or ICF [14–16]. A UCF model mainly finds the items that similar users like and predict the target user's rating of the corresponding items. An ICF model finds the similarity between items. Only when the target user's rating of certain items is found, it can predict items with high similarity and recommend several similar items with the highest scores to the user. A model-based CF model is also called a learning-based model [13]. It defines a parameter model to describe the relationship between the users and items and then obtains the model parameters through the optimization process. The model-based CF model is scalable and can handle higher sparsity than the memory-based method, but when users or items without any ratings enter the system it is also miscalculation. Different machine learning algorithms are based on the model-based CF method such as the matrix factorization algorithm [17], neural networks [5], Bayesian classifiers [18], the clustering algorithm [19], genetic algorithms [20], and the regression model [21] among others.

In order to further improve the CF algorithms and meet the needs of the development of the recommendation system, we must first analyse the problems that exist in the implementation of CF and then make targeted improvements. Through a research of CF algorithms and recommendation systems, we found the main problems in the realization of CF algorithms and they are listed as follows.

- **Cold start problem:** The cold start problem is also known as the first-rater or new-item problem. From a certain perspective, it can be regarded as an extreme case of the sparse problem. Because the traditional CF recommendation is based on the calculation of similar users/items to get the recommendation of the target user, when a new item appears for the first time, the simple CF cannot predict its score and recommend items because no user has evaluated it. Moreover, due to the early emergence of new items, there are fewer user reviews and the accuracy of recommendations is also relatively poor. Similarly, the recommendation system for new users is also very poor. An extreme case of the cold start problem is when a CF recommendation system runs for the first time, and every user faces the cold start problem on every project [22, 23].

- **Sparsity problem:** In the sparsity problem, the realization of the CF algorithm first needs to use the user-item rating matrix to represent the user information. Although this is simple in theory, many e-commerce recommendation systems need to process a large amount of data information, and in these systems, the total amount of goods purchased by general users accounts for about 1% of the total amount of goods on the website, so the rating matrix (user-item matrix) is very sparse. In the case of large and sparse data, on the one hand it is difficult to find the nearest neighbour user set, and on the other hand the cost of similarity calculation is very high. At the same time, because the data is very sparse when the nearest neighbour user set of the target user is formed, moving forward causes loss of information which leads to the reduction of the recommendation effect [15, 22].

- **Scalability problem:** In the CF recommendation algorithm, the global numerical algorithm can use the latest information from time to time to generate a relatively accurate user interest prediction or recommendation for users. However, in the face of an increasing number of users the amount of data increases sharply, and the algorithm's scalability issues (i.e., the problem of adapting to the expanding scale of the system) have become an important factor restricting the implementation of the recommendation system. Although compared with model-based algorithms, global numerical algorithms save the training time for building models, the amount of calculation used to identify the nearest neighbour algorithm increases greatly with the increase in users and items for millions. The usual algorithms will encounter serious scalability bottlenecks. If this problem is not solved well, it directly affects a recommendation system based on the CF technology to provide users with a real-time solution to the recommendation problem. The better the real-time performance of the recommendation system and the higher the accuracy, the more the system will be accepted by users [15, 22].

In this work, we address these issues of CF RSs by applying a hybrid approach of the UCF algorithm and ICF algorithm. The bottleneck in traditional CF algorithms is the search for neighbours amongst a large user population of potential neighbours. A hybrid model avoids this bottleneck by modelling the relationships between items and the relationships between users. Because the relationships between users or items are relatively static, a hybrid of the user and item algorithms may be able to provide a higher quality compared to the UCF and ICF model, with less online computation. Here, we present a new recommendation model named ‘An Efficient Hybrid Recommendation Model Based on Collaborative Filtering Recommender System’ that blends UCF and ICF models with a linear regression model (ΓUICF). The ΓUICF model is utilised to reduce the dimension of the user-item rating matrix. The major different user to user and item to item similarity computations are computed and then the prediction for the unobserved target rating is made by adapting UCF and ICF models in an offline step. The Γ parameter linear regression model learns for each user and item and fuses the two predictions in the online step in a weighted model. The experiments were conducted on two different real-world datasets, namely 100k MovieLens and 1M MovieLens datasets. The proposed model was observed to perform significantly when compared to the existing models.
The structures of this work is organised as follows:

The related works are given in Section 2, a methodology is presented in Section 3, the detailed experimental work is given in Section 4, and finally, the conclusion and future work are given in Section 5.

2 | RELATED WORKS

Presently, the collaborative filtering (CF) method is attracting a large number of research works, resulting in a huge variety of CF recommendation models. These models are frequently not deterministic. CF is a recommendation model based on similar information, that is, it finds an item or the user’s ‘neighbour’ based on the user’s behaviour with the item. The ‘neighbour’ here generally refers to the group, and its basic assumption is that similar users may use similar preferences and similar items may be preferred by similar people [24]. It mainly includes UCF and ICF. In both models, a very small portion of the information from the user-item rating matrix is taken into consideration to predict the unobserved ratings. This leads to the effect of prediction accuracy. To address these research problems, authors have been recently seeking to combine UCF and ICF models.

Tso-Sutter KH et al. [25] proposed a personalised recommendation model that combines items, users, and tags. The tags represent the information about the items given by the users. Firstly, they developed a generic model allowing tags to be integrated into CF models. Secondly, they presented a hybrid technique to project the 3-dimensional correlations between items, users, and tags.

Wang, Jing and Jian Yin. [26] developed a blend model that integrates UCF and ICF models for efficient recommendation diversity. The proposed model predicted a varied recommendation list of items for the target users. Item ranking means the users’ rating frequency for every item in the system.

Yamashita et al. [27] proposed an adaptive blend of UCF and ICF models to efficiently figure out the accuracy of recommendation systems. The proposed model uses a weight to integrate the UCF and ICF model. Firstly, they evaluated the correlation between the weight parameter and prediction accuracy. Secondly, they presented an estimation of the appropriate weight value based on absolute collected ratings.

Miyahara and Pazzani [28] developed a hybrid model that blends UCF and ICF models to efficiently evaluate the performance of predictions. This model includes filling out a pseudo score using UCF and ICF based on the simple Bayes. Then, the Bayesian CF recommender returns the hybrid results.

Kant S and Mahara T. [29] proposed a biclustering (BiUCF) model that combines UCF and ICF models to project the dimension reduction and simultaneously group the users and items of the rating data matrix. Biclustering computed the similarity between users or items on the basis of the cluster they belong to. Then the prediction for the unobserved rating is obtained by utilizing UCF and ICF models.

Slokom, Manel, and Raooua Ayachi [30] proposed a hybrid model that combines the prediction results of the possibilistic UCF and ICT model using a possibilistic information affinity measure to represent user-item preferences.

The hybrid models that we mentioned gave good results and addressed the issue of sparsity of the user-item rating matrix. However, these models have shortcomings when dealing with real-world applications and result in imprecise user preferences; hence, the cold start problem and the sparsity problem remain in CF models. To this end, we present a new hybrid model that combines UCF and ICF models with a linear regression model to solve the issue of sparsity and scalability of the user-item rating matrix and the performance of predictions that we get is significant.

3 | SYSTEM OVERVIEW

In this section, firstly, we presented the impact of some properties of the user-item rating matrix on collaborative filtering (CF) models in subsection 3.1. Then in subsection 3.2, we computed the similarities of both user based (UCF) and item based (ICF) models via two different metrics similarity namely, Cosine Similarity (COS) and Pearson Correlation Co-efficient (PCC). After finding similarities, we provided the prediction of both UCF and ICF models in subsection 3.3. UCF and ICF models have limitations related to sparsity and scalability. More precisely, the user-item rating matrix is sparse, and the user to user or item to item similarity matrix is quite dense. This is because even a few often purchased items can lead to dense user-to-user or item-to-item similarities. Hence, the accuracy of the prediction of each model is poor and imprecise. Therefore, to solve this issue, we propose a new recommendation model which is a fusion of the strengths of UCF and ICF models and a parameter linear regression model to minimise the prediction error. The proposed model is provided in detail in subsection 3.4.

3.1 | Problem formulation

The CF model aims to suggest new items or to predict the utility of a specific item for a specific user based on the user’s previous interest and the behaviours of many other users.

We suppose that the user-item ratings matrix is an incomplete \( m \times n \) matrix \( R_{m \times n} = \{r_{ui}\} \) containing a set of \( m \) users \( \{u_1, u_2, u_3, \ldots, u_m\} \) and set of \( n \) items \( I = \{i_1, i_2, i_3, \ldots, i_n\} \) (see Figure 1). Each user \( u_i \) has a rated list of items \( I_{uis}\) about which the user has expressed his/her behaviour. Behaviours can be explicit feedback given by the user as rating score within a specific integer range between 1 and 5, or implicit feedback derived from the user’s purchase record such as clicks, browsing history, timing logs, and so on. Note that \( I_u \subset I \) and \( I_u \) can be a null set. There exists a distinguished list \( u_i \in \mathbb{U} \) called active user for whom the task of a CF model is to find an item likeliness that can be of two forms.
1. **Prediction** of unknown ratings is a numerical value \( \hat{r}_{ui} \), representing the predicted similarity of an item \( i_j \in I_u \) for the active user \( u \). This predicted numerical value is on a scale of 1 to 5 based on the opinion values provided by \( u \).

2. **Recommendation of the Top-k items or Top-k users** is a list of \( k \) items \( i_j \), \( I_i \subseteq I \), that the active user will like the most. Note that the recommended list of items must be on items not already rated (purchased) by the active user, that is, \( I_i \cap I_u = \emptyset \). This formulation of a CF model almost always refers to the Top-K recommendation.

Note: The Top-k problem term almost frequently indicates the process of finding the Top-k items instead of the Top-k users. Nevertheless, Top-k users’ formulation is also beneficial to the merchant to determine the best users to target with a marketing effort.

### 3.2 Computed similarity of UCF and ICF models

Generally, the similarity between the users or items based on the CF model is computed by treating them as vectors in the user space or item space and computing their similarity. Various metrics are proposed such as the COS measure [15], PCCs [31], and Jaccard [32].

In this work, the COS measure and PCC are adopted, as provided in [27, 33], and we have explained them as follows:

#### 3.2.1 Cosine similarity

COS finds the correlation between two variables, which measures the similarity between two feature vectors in terms of the angle between them. The COS formula of the UCF model is defined in Equation (1).

\[
\hat{r}_{ui} = \frac{\sum_{i \in \text{ratings }} u \cdot r_{ui} \cdot v_{ij}} {\sqrt{\sum_{i \in \text{ratings }} u \cdot r_{ui}^2} \sqrt{\sum_{i \in \text{ratings }} v_{ij}^2}} \tag{1}
\]

where the dot product \( u \cdot v \) of the user vector is obtained and then divided by the product of the Euclidean length of the vector. \( I_{uv} \) represents the set of items rated by both users \( u \) and \( v \), and \( r_{ui} \) and \( r_{vi} \) represent the rating of user \( u \) or \( v \) for the item \( i \).

Likewise, the formulation COS of the ICF model is shown in Equation (2).

\[
\hat{r}_{ij} = \frac{\sum_{i \in \text{ratings }} u \cdot r_{ij} \cdot v_{ij}} {\sqrt{\sum_{i \in \text{ratings }} u \cdot r_{ij}^2} \sqrt{\sum_{i \in \text{ratings }} v_{ij}^2}} \tag{2}
\]

where \( s(i,j) \) represents the similarity of items \( i \) and \( j \), \( \cup_{ij} \) represents the set of common users who rated items \( i \) and \( j \) across all the users who rated such items.

#### 3.2.2 Pearson correlation coefficient

PCC is a measure of vector similarity. The output range is between −1 and +1, where zero means no correlation, negative means the negative correlation, and a positive value means a positive correlation.

The UCF model formula based on the PCC is given in Equation (3).

\[
PCC_{user}(u, v) = \frac{\sum_{i \in \text{ratings }} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)} {\sqrt{\sum_{i \in \text{ratings }} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in \text{ratings }} (r_{vi} - \bar{r}_v)^2}} \tag{3}
\]

where \( s(u, v) \) represents the similarity of users \( u \) and \( v \), and \( \bar{r}_u \) and \( \bar{r}_v \) donate the mean rating of user \( u \) or \( v \) across all items.
Likewise, the formulation of the ICF model based on PCC becomes

$$PCC_{item}(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_{i})(r_{uj} - \bar{r}_{j})}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_{i})^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_{j})^2}}$$

(4)

where $s(i,j)$ represents the similarity of items $i$ and $j$, and $\bar{r}_{i}$ and $\bar{r}_{j}$ donate the mean rating of item $i$ or $j$ across all the users who rated such items.

### 3.3 Rating prediction of UCF and ICF models

First user similarity and item similarity are computed. The next step of the recommendation way includes predicting an estimation of the unobserved rating for the items that the target user has not. Through a list of rating estimates for unrated items, the recommender model will simply sort by decreasing value and recommend the Top k-Items of the list to the target user. The prediction formula of UCF and ICF models are shown in Equations (5) and (6), respectively.

$$\hat{r}_{u,i} = \mu_u + \frac{\sum_{v \in N_i(u)} \text{sim}_{uv} \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i(u)} \text{sim}_{uv}}$$

(5)

$$\hat{r}_{u,i} = \mu_u + \frac{\sum_{j \in N_i(i)} \text{sim}_{ij} \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_i(i)} \text{sim}_{ij}}$$

(6)

where $\mu_u$ represents the mean rating of user $u$ across all the items rated. $N_i(u)$ represents the number of neighbours who have item $i$ in common with user $u$, of which $v$ is a particular neighbour user and $\text{sim}_{uv}$ is the similarity between the user $u$ and one of his or her neighbours $v$.

Since all the previous models need to be calculated for each prediction, the time cost is large and not all the scored items belong to the category similar to the item. Therefore, the TopK technique has adopted to filter out the items $K$ that have the highest similarity with the item among all the scored items and only weigh the average of the items represented by $K$ (see Equation (7) and (8)).

$$\hat{r}_{UTopK}^{U,i} = \mu_u + \frac{\sum_{v \in N_i(u)} \text{sim}_{uv} \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i(u)} \text{sim}_{uv}}$$

(7)

### 3.4 Hybrid proposed model (TUICF)

The largest e-commerce sites are operating on a scale that emphasises the direct implementation of the CF algorithm. In memory-based CF models such as UCF or ICF similarity computation, the nearest neighbour formation process step turns out to be the performance bottleneck, which in turn, can make the entire process inappropriate for generating real-time recommendations. Moreover, the user-item rating matrix experiences high sparsity. Hence, in some cases, the density of the matrix is lower than 1%. This is because users rated only some of the items that they have viewed from a large pool of items in the user-item rating matrix. Therefore, the UCF or ICF model needs to predict the unobserved ratings from the few available ratings. According to Sarwar et al. [15], the similarities computed using the cosine or correlation measure may be imprecise or misleading in the sense that two rating vectors may be distant (in Euclidean sense), yet they may have very high similarity. In this case, using the raw ratings of a similar item may result in poor prediction.

One of the mechanisms ensuring high sparsity and scalability is the employment of model-based algorithms. Model-based algorithms have the power to contribute to RS to operate on a highly sparse and scalable dataset. Sarwar et al. [15] proposed a linear regression model to improve the performance of the ICF model. Their model used the results of the linear regression model instead of simply using the raw rating of similar items for prediction. Their model used approximated values based on the linear regression model (see Equation (9)).

$$\hat{r}_N = \Gamma \hat{r}_f + \beta + \epsilon$$

(9)

where $\Gamma$ and $\beta$ indicate the parameter determined by going over both rating vectors, and $\epsilon$ indicates the error in the regression model.

More precisely, the general and intuitive supposition is that there exists a relation between the overall item ratings and the single criteria ratings. Practically, the overall rating $r_0$ can subsequently be seen as being determined by a function $f$ of single criteria ratings (see Equation (10)).

$$r_0 = f(r_1, \ldots, r_k)$$

(10)

The prediction of $r_0$ for a given user $u$ and a target item $i$ can be accomplished in a multi-step process. Firstly, in an offline phase, the function $f$ has to be determined. One option
could be to define $f$ based on domain expertise or by averaging the criteria ratings. A more promising method, however, is to apply statistical or machine learning algorithms to automatically detect the hidden relationship between the overall rating and the criteria ratings [34].

For example [34], to approximate a function $f$ for each item in the system using multiple linear regression algorithms. Hence, the overall rating $r_i$ can be observed to be dependent on a linear aggregation of the criteria ratings, where every criterion is assigned a parameter $\Gamma_i$, that is $r_i$ (see Equation (11)).

$$r_i = \Gamma_1 r_1 + \Gamma_2 r_2 + \cdots + \Gamma_n r_n + \epsilon$$  \hspace{1cm} (11)

As shown in Equation (11), in the online step, the first criteria rating $r_1$ to $r_n$ for $i$ have to be estimated. Afterwards, the overall rating can be computed using $f$. For computing $r_i$ to $r_n$, any standard CF model can employ them one by one for each criterion. For example, in the movie domain, we were able to observe the prediction of the rating for the aspect of the action to specify its own CF issue, which we compute based on the ratings for this aspect of the neighbours of the target user.

In this work, we propose a blend model-based algorithm (TUICF). We learn a linear regression function for UCF and ICF models and fuse the two predictions to result in a weighted model. We used the $\Gamma$ parameter (weights) as the impact coefficient for all the users and items corresponding to the UCF and ICF model, respectively. The similarity calculation model is still correlation-based but the calculation is performed on the user’s and item’s latent space.

More precisely, in the UCF linear regression model, we try to explore for every user which aspects of an item (movie) are of specific significance across all the items (movies) the user has scored, while in the ICF linear regression model, we try to explore how strongly the particular aspects of an item (movie) affect the overall item (movie) ratings across all users.

Generally, we try to evaluate two kinds of dependencies in the user-item rating matrix. The main idea here is that we just need a small fraction of similar users and similar items to compute predictions that lead us to propose an alternate model-based algorithm. Although this process is effective, the user-item rating matrix can be very sparse because the users scored only some of the items that they have viewed from a large pool of items in the user-item rating matrix. Therefore, we can learn from the idea of Kant S, Mahara T. [29], and Jannach D et al. [34] that both models can complement each other and can possibly produce more accurate predictions compared to the individual model and the formula of the proposed hybrid model as follows:

$$\widehat{r}_{ui}^{TUICF} = \Gamma_u \ast \widehat{r}_{ui}^{TopkU} + \Gamma_i \ast \widehat{r}_{ui}^{TopkI}$$  \hspace{1cm} (12)

Similar to the idea in the TopK method, the formula can be improved to Topk, a linear fusion function, and the degree of fusion is used as the weights of the prediction function (see Equations (12) and (13)).

$$\widehat{r}_{ui}^{TUICF} = \Gamma_u \ast \widehat{r}_{ui}^{TopkU} + (1 - \Gamma_i) \ast \widehat{r}_{ui}^{TopkI}$$  \hspace{1cm} (13)

Among them, $\Gamma \in [0, 1]$ is degraded into two models at the two endpoints, and the coefficient $\Gamma$ represents the degree of fusion where $\widehat{r}_{ui}^{TopkU}$ and $\widehat{r}_{ui}^{TopkI}$ are the prediction results of the UCF and ICF based CF models, respectively, $\Gamma_u$ and $\Gamma_i$ are the parameters of the regression model which controls the influence of the UCF and ICF models.

The proposed model (TUICF) utilised the $\Gamma_u$ and $\Gamma_i$ parameter regression model for each user and item and fused the UCF and ICF models in the online phase to minimize the error prediction and infer a user’s preference for the item. The intuition behind the fusing UCF and ICF models is that both kinds of information can be valuable. In other words, the user will be interested in watching movies that are similar to the movies the user previously liked and will tend to avoid movies that are similar to the movies the user previously disliked. This method does not require identification of the vicinity of similar users when requesting a recommendation. As a result, it tends to make recommendations much faster.

**Algorithm 1 ΓUICF Algorithm**

1. **Input**: $R_{m \times n}$ user-item ratings matrix (MovieLens dataset)
   2. **Output**: Prediction of unknown ratings $\hat{r}_{ui}$ for $R_{m \times n}$ user-item matrix using RMSE Eq.: 15 and MAE Eq.: 14 evaluation metric
   3. **for** user $u \in R_{m \times n}$ do
   4. **for** all users $u \in R_{m \times n}$ do
   5. **if** CosSim then
   6. Calculate $\hat{r}_{ui}^{BCF}$ baseline CF model (Eq.:18)
   7. **else** FCC
   8. Calculate $\hat{r}_{ui}^{BCF+UCF}$ UCF + baseline CF model (Eq.:19)
   9. **else** FCC
   10. Calculate $\hat{r}_{ui}^{TopkUCF}$ Topk UCF model (Eq.:7)
   11. **end if**
   12. **end for**
   13. **end if**
   14. **end for**
   15. **for** all items $i \in R_{m \times n}$ do
   16. **if** CosSim then
   17. Calculate $\hat{r}_{ui}^{Topk1CF}$ Topk ICF model (Eq.:7)
   18. **end if**
   19. **for** all users $u \in R_{m \times n}$ do
   20. **end for**
21: \textbf{else} PCC  
22: \textbf{Calculate} PCC \((i, j)\) (Eq.:3)  
23: \textbf{Pred} \(\hat{\Gamma}_{ui}\) Topk ICF model (Eq.:7)  
24: \textbf{end if}  
25: \textbf{end for}  
26: \textbf{Initialization:} \(\Gamma_u, \Gamma_i;\)  
27: \textbf{for each user} \(u\) do  
28: \textbf{for each item} \(i\) has rated by user \(u\) do  
29: \textbf{Pred} \(\hat{\Gamma}_{ui}^{UICF} = \Gamma_u \ast \hat{\Gamma}_{ui}^{TopkUCF} + (1-\Gamma_i) \ast \hat{\Gamma}_{ui}^{TopkICF}\) (Eq.:12)  
30: \textbf{Pred} \(\hat{\Gamma}_{ui}^{UICF(norm)} = \Gamma_u \ast \hat{\Gamma}_{ui}^{TopkUCF norm} + (1-\Gamma_i) \ast \hat{\Gamma}_{ui}^{TopkICF norm}\) (Eq.:12)  
31: \textbf{end for}  
32: \textbf{end for}  

Algorithm 1 depicted a pseudo-code description of the \(\Gamma\) UICF model which takes as the input of ratings \(\hat{R}_{m \times n}\) the user-item sparse matrix and outputs a prediction of unknown ratings based on the proposed model. The bulk of the first loop (Line 1) computed the baseline CF model through the mean user, the mean item, and all mean (Line 1). The bulk of the second loop (Line 4) takes the UCF model where the loop consists of the condition. If the measured similarity is a cosine similarity, then the similarity between the users (Line 4) is measured and then the basis of the UCF model (Line 7), the basis of the UCF model (Line 7), and the basis of the Topk UCF model (Line 9) are computed. Otherwise, the UCF model is normalized using the PCC (Line 12).  

Similarly, the bulk of the third loop (Line 14) takes the ICF model, where the loop consists of the condition. If the measured similarity is a cosine similarity, then the similarity between the items (Line 17) is measured and then the basis of the ICF model (Line 18), the basis of the ICF model (Line 18), and the basis of the Topk UCF model (Line 20) are computed. Otherwise, the ICF model is normalized using the PCC (Line 23).  

We initialised \(\Gamma_u\) and \(\Gamma_i\) (Line 26). The bulk of the loop through each user and each item has been rated by a user \(u\) (Line 18), and the unknown ratings have been predicted by weighting all the users and items corresponding to the Topk UCF and TopkICF models (Line 29).  

Line 30 predicted the unknown ratings by weighing all the users and items corresponding to the normalised TopkUCF and TopkICF models.  

4 | EXPERIMENTAL STUDY  

In this section, we present our experimentation which includes the experimental setup in subsection 4.1, a dataset description in subsection 4.2, and the evaluation metrics is presented in subsection 4.3. The settings of the proposed model is given in subsection 4.4, and finally, the experimental results are demonstrated in subsection 4.6.  

4.1 | Experimental setup  

We implemented our experimentation of the models on Ubuntu 16.4 operating system running on Intel® Core™ i5-2400 CPU 3.10 GHz 4 processors and a hard disk of 500 GB. We used the Python language version 3.6.  

4.2 | Dataset description  

Originally, we used the NetflixPrize dataset, which is a dataset on movie ratings. The standard NetflixPrize dataset contains 480,189 users, 17,770 items, and a total of about 100 million ratings. The dataset also includes the time of scoring, and the name and year corresponding to each movie id. For CF methods, the scale matrix generated by this dataset reaches 480,189 \(\times\) 17,770, and the total items are more than 8 billion. The basic statistics on this matrix have already taken tens of seconds. Fine-grained calculations are slower. Therefore, we switched to a smaller dataset, MovieLens (http://grouplens.org/datasets/movielens), which is of the same data format. It provides datasets of different sizes including versions of multiple sizes such as 100K, 1M, 10M, and 20M (both refer to rating numbers).  

Two types of MovieLens datasets are used in our experiments. The first dataset in MovieLens is called 100K MovieLens; there are 100,000 ratings with 943 user and 1682 movies. Another is 1M MovieLens; it includes 6040 users and 3952 movies with 1,000,209 ratings. In both datasets, each user has rated at least 20 movies. The sparsity of the user-item matrix is 93.701% in 100k and 95.810% in 1M. We choose these two datasets because they are the most used datasets by researchers and developers in a CF domain.  

4.3 | Evaluation metrics  

In this work, the mean absolute error (MAE) and the root mean squared error (RMSE) are used as the evaluation metrics to evaluate the prediction accuracy of estimating the ratings of specific user-item combinations. The lower the value of the MAE and RMSE, the better the prediction effect. Their definitions are presented Equation (14) and Equation (15).  

\[
MAE = \frac{\sum_{i=1}^{N}|r_{ui} - \hat{r}_{ui}|}{N} \quad (14)
\]

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N}(r_{ui} - \hat{r}_{ui})^2}{N}} \quad (15)
\]

where \(\hat{r}_{ui}\) and \(r_{ui}\) are the predicted and actual scores of user \(u\) for item \(i\).  

Because of the cross-validation technique, the average MAE and RMSE of each dataset is selected as the final RMSE value of the prediction method.
\[ \text{MAE} = \frac{1}{k} \sum_{i=1}^{k} \text{MAE}_i \quad \text{and} \quad \text{RMSE} = \frac{1}{k} \sum_{i=1}^{k} \text{RMSE}_i \]  

where \( k \) is the number of cross-validation groups.

### 4.4 Model setting

Before the experiments, we need to set the parameters for our model. Originally, for all the data we used a set of the training set and test set, and the scale ratio of the training set and test set was 4:1, that is, the scale of the training set is 80,000 and the scale of the test set is 20,000. The 5-fold cross-validation method was adopted in each dataset (see section 4.3 and Equation 16). The TopK approach was adopted on the training set and test set, where the values of \( k \) scale from 5 to 200 are shown in Table 1 and Figure 2a and 2b. Moreover, in Table 2 and Figure 3a, 3b represents the degree of fusion, where the value of \( \Gamma \) is between [0, 1].

For the 100k MovieLens dataset, the optimal \( k \) value for Topk UCF is \( (k = 30) \), Topk ICF is \( (k = 20) \), Topk user normalised CF is \( (k = 40) \), Topk item normalized CF is \( (k = 20) \), and the blend parameter linear regression is \( \Gamma = 0.70 \), as shown in Table 1 and Table 2, respectively.

For the 1M MovieLens dataset, the optimal \( k \) value for Topk UCF is \( (k = 30) \), Topk ICF is \( (k = 20) \), Topk user normalised CF is \( (k = 40) \), Topk Item normalised CF model is \( (k = 20) \), and the blend parameter linear regression is \( \Gamma = 0.80 \).

### 4.5 Analysis and discussion

In this section, we analysed and discussed the basic models of the CF model such as the baseline CF model (BCF), baseline UCF and ICF model, UCF and ICF model with BCF, BCF with bias, Topk UCF and Topk ICF model, normalised UCF and ICF model and normalised ICF model. According to their results, we proposed our blend model.

#### 4.5.1 Baselines CF model

The CF baseline model is a simple prediction based on these statistics. Its prediction formula is shown in Equation (17).

\[ \hat{r}_{ui} = \mu + u_b + i_b \]  

Among them, \( \mu \) is the population mean, and \( u_b \) and \( i_b \) are the deviations of the user \( u \) and item \( i \) mean from the population mean (see Equation (18)).

\[ \hat{r}_{ui}^{BCF} = r_{user \ u} + r_{item \ i} - \mu \]  

This model is the effect of the baseline CF, and its RMSE value is 0.9694 and 0.9346 for the 100k and 1M MovieLens dataset, respectively.

#### 4.5.2 Baseline UCF and ICF models

The similarity is measured by the Cosine similarity metric for UCF and ICF baseline models, respectively. After the user similarity and item similarity are obtained, the similarity matrix is used for prediction. The prediction formula of UCF and ICF baseline models is shown in Equations (198) and (2019), respectively. In these models, the implicit assumption is that all users score out of the same statistical distribution.

\[ \hat{r}_{ui}^{UCFbaseline} = \frac{\sum_{v \in N(i)} r_{uv} \cdot r_{vi}}{\sum_{v \in N(i)} r_{uv}} \]  

\[ \hat{r}_{ui}^{ICFbaseline} = \frac{\sum_{v \in N(u)} s_{ij} \cdot r_{vj}}{\sum_{v \in N(u)} s_{ij}} \]

It is not possible to calculate a weighted average of all the data here because it does not have scored items and averaging is meaningless. Hence, it will increase the value of the denominator, leading to serious deviations in prediction. Therefore, the cross-validation result of the UCF model is 1.0149 and is lower than the result based on the ICF model, which is 1.0174 on the 100k MovieLens dataset and also exceeds 1. While the RMSE result of the UCF model is 0.9726, which is higher than the result based on the ICF model, i.e. 1.0025 on the 1M MovieLens dataset.

#### 4.5.3 UCF and ICF with baseline CF

The significance of the baseline CF model is important to use. We combined UCF and ICF with the baseline CF to predict the basis of each model (see Section 3 Equations (5) and (6)). Hence, the RMSE error of the UCF and ICF model based on the baseline CF model has been decreased to 0.9548 and 0.9362, respectively, which is a great improvement over the baseline on the 100k MovieLens dataset.

Similarly, for the 1M MovieLens dataset, the RMSE error of the UCF and ICF model based on the baseline CF model has been decreased to 0.9225 and 0.8960, respectively. It can be concluded that in practical applications, the ICF based on the baseline CF model performs better, roughly the difference between commodities, but not as good as the difference in taste between people.

#### 4.5.4 Topk UCF and Topk ICF models

The TopK technique has been adopted for UCF and ICF models based on the baseline CF model to filter out the items \( K \) that have the highest similarity with the item among all the scored items and to only weigh the average of the items \( K \). If the number of items scored by the user is less than \( K \), then all the scored items are used directly. This is also because some users may only tend to rate their favourite products. At run time, when item-item CF is
TABLE 1 Root mean squared error (RMSE) performance with different K values for UCF and ICF models on the 100k and 1M MovieLens dataset

| 100 MovieLens | 1M MovieLens |
|---------------|--------------|
| k             | ICF TrainSet | ICF TestSet | UCF TrainSet | UCF TestSet | k             | ICF TrainSet | ICF TestSet | UCF TrainSet | UCF TestSet |
| 5             | 0.5747       | 0.9543      | 0.6101       | 0.9874      | 5             | 0.5211       | 0.8975      | 0.5717       | 0.9544      |
| 10            | 0.6845       | 0.9278      | 0.7242       | 0.9573      | 10            | 0.6401       | 0.8746      | 0.6936       | 0.9244      |
| 15            | 0.7322       | 0.9223      | 0.7699       | 0.9489      | 15            | 0.6953       | 0.8606      | 0.7447       | 0.9145      |
| 18            | 0.7500       | 0.9215      | 0.7866       | 0.9468      | 18            | 0.7135       | 0.8685      | 0.7636       | 0.9118      |
| 20            | 0.7596       | 0.9213      | 0.7951       | 0.9458      | 20            | 0.7242       | 0.8681      | 0.7734       | 0.9106      |
| 25            | 0.7774       | 0.9217      | 0.8113       | 0.9449      | 25            | 0.7447       | 0.8686      | 0.7921       | 0.9085      |
| 30            | 0.7902       | 0.9225      | 0.8225       | 0.9445      | 30            | 0.7595       | 0.8692      | 0.8054       | 0.9076      |
| 40            | 0.8073       | 0.9242      | 0.8373       | 0.9447      | 40            | 0.7796       | 0.8712      | 0.8230       | 0.9067      |
| 50            | 0.8182       | 0.9258      | 0.8465       | 0.9453      | 50            | 0.7928       | 0.8733      | 0.8544       | 0.9067      |
| 100           | 0.8422       | 0.9314      | 0.8660       | 0.9491      | 100           | 0.8233       | 0.8812      | 0.8605       | 0.9093      |
| 200           | 0.8546       | 0.9345      | 0.8747       | 0.9526      | 200           | 0.8415       | 0.8892      | 0.8766       | 0.9135      |

TABLE 2 Root mean squared error (RMSE) performance with different \( \Gamma \) values on the train and test set for the 100k and 1M MovieLens dataset in the blend model

| \( \Gamma \) Parameter | 100k MovieLens | 1M MovieLens |
|-------------------------|----------------|--------------|
|                         | TrainSet       | TestSet      | TrainSet       | TestSet |
| 0.00                    | 0.8192         | 0.9388       | 0.8233         | 0.9048  |
| 0.10                    | 0.8077         | 0.9309       | 0.8076         | 0.8948  |
| 0.30                    | 0.7880         | 0.9187       | 0.7796         | 0.8787  |
| 0.50                    | 0.7730         | 0.9115       | 0.7567         | 0.8681  |
| 0.60                    | 0.7674         | 0.9097       | 0.7473         | 0.8649  |
| 0.65                    | 0.7651         | 0.9094       | 0.7432         | 0.8639  |
| 0.70                    | 0.7630         | 0.9093       | 0.7394         | 0.8632  |
| 0.75                    | 0.7613         | 0.9096       | 0.7359         | 0.8628  |
| 0.80                    | 0.7600         | 0.9101       | 0.7328         | \textbf{0.8628} |
| 0.90                    | 0.7582         | 0.9123       | 0.7279         | 0.8639  |
| 1.00                    | 0.7578         | 0.9157       | 0.7244         | 0.8665  |

**FIGURE 2** Root mean squared error (RMSE) performance of each K value for item to item and user to user CF models on the 100k and 1M MovieLens dataset (a) RMSE performance of each K value on the 100k MovieLens dataset (b) RMSE performance of each K value on the 1M MovieLens dataset

used and when selecting \( K = 10 \), the resulting RMSE is \textbf{0.9278}, and when selecting \( K = 40 \), the resulting RMSE is \textbf{0.9242}. It is found that different \( K \) values have certain effects. When \( K \) is very small, it cannot cover all the items that are similar to it, and when \( K \) is very large, the selected item may already no longer be very similar to it. We tried to adjust \( K \) with different \( K \) values as observed in Table 1 and Figures 2a, 2b. It is found that the RMSE of the training set was gradually increased, instead of gradually decreasing in the general sense.

The main reason is that the training data incorporates the mean and similarity, and the original information is inevitably
used in the prediction, so the values of RMSE on the training set are not very representative. Moreover, the test set shown in Table 1 and Figures 2a, 2b is the corresponding optimal $K$ value. When $K$ increases, its negative effect is not so great. This is probably because with more data, the marginal data has a reduced impact on the final predicted value due to its weight reduction, similar to the law of diminishing marginal effects in economics. Obviously, different sizes of input should have different sizes of $K$ values to match. A larger scale data should require a larger scale $K$ value.

Here, the method we adopted is to directly set a $K$ value set, so it is not very adaptable to different scales. In practical applications, a better way to obtain a better $K$ value is through learning. Hence, for the 100K MovieLens dataset, the optimal values of $\hat{k}$ of UCF and ICF models are $K = 20$ and $\hat{k} = 30$, and the RMSE is 0.9067 and 0.8681 for UCF and ICF models, respectively. For the 1M MovieLens dataset, the optimal value of $\hat{k}$ of UCF and ICF models is $K = 20$ and $\hat{k} = 40$ and RMSE is 0.9445 and 0.9213 for UCF and ICF models, respectively.

4.5.5 Normalize UCF and ICF models

In this work, the PCC metric has been adopted to normalise the UCF and ICF models. The normalised similarity matrix of users and items are calculated as shown in Equations (3) and (4), respectively.

At the same time, the function used for the prediction can be kept unchanged and only the similarity matrix adopted therein can be replaced by the normalised similarity matrix. However, based on the TopK ICF model, the cosine similarity has been replaced by the normalised similarity matrix (PCC), but it is observed that the RMSE based on ($k = 20$) has increased from $0.9213$ to $0.9253$ in the 100k MovieLens dataset, while the RMSE of the ICF model based on ($k = 20$) has decreased from $0.8686$ to $0.8585$ in the 1M MovieLens dataset.

This is due to some properties of the 100k MovieLens dataset itself. For some items with few scores, after normalisation they are erased. These items already have very little information. For example, in a niche movie, three or four audiences with similar tastes have scored a high score of 5 points at the same time. After normalisation, the high score information of the movie was erased at once but a loss of information occurred.

For the UCF model, since each user has scored at least 20 ratings in the 100k MovieLens dataset, such disadvantages should be avoided as much as possible. Therefore, we used the normalised similarity matrix for the UCF model. However, the effect was also unsatisfactory because the RMSE of the UCF model based on ($k = 30$) increased from $0.9445$ to $0.9550$, while the RMSE value of the UCF model based on ($k = 40$) decreased from $0.9059$ to $0.8925$ in the 1M MovieLens dataset.

Regarding this issue, based on several related works, we found that there are two main considerations. First, there is extensive discussion on the measurement of similarity. Among them, there is a method that still uses the similarity calculation between items, but at this time, the item is not normalised. Instead, the user is normalized [15]. Another solution is to use the default voting method [33] when the number of scores is insufficient. In this method, an analysis method similar to tf-idf is used to obtain the default weight value. But the formula is more complicated [33]. Secondly, due to the similarity measurement method, in actual use, the vector distance of many essentially similar objects may not be ideal under the Euclidean space [15], which leads to deviations in prediction.

4.6 Results

In the above models there are two fundamentally different methods, item-item CF and user-user CF. Although derived from the same data, they are of two different dimensions. We used the results of the linear regression model instead of simply using the raw rating of similar items for prediction (See Equation (12)).

Without considering the normalised similarity matrix, we merged the TopK UCF and the TopK ICF model with the best parameter linear regression model, where the TopK of UCF, ICF, and $\Gamma$ models based on the 100k MovieLens are $k = 30$, $k = 20$, and $\Gamma = 0.70$, respectively. Hence, the RMSE and
MAE values of the blended model ΓUICF are 0.9130 and 0.7174, respectively.

In case the 1M MovieLens dataset and the TopK of UCF, ICF, and Γ models are \( k = 40 \) and \( k = 20 \), with \( \Gamma = 0.80 \), the RMSE and MAE values after fusion are 0.8645 and 0.6751.

After considering the normalised similarity matrix, we merged the UCF normalised and ICF normalised models with the best parameter linear regression model. For the 100k MovieLens dataset, the optimal values of UCFNorm, ICF-Norm, and Γ models are \( k = 40 \), \( k = 20 \), and \( \Gamma = 0.70 \), respectively. After fusion, the RMSE and MAE values are to 0.9229 and 0.7219, respectively.

For the 1M MovieLens dataset, RMSE and MAE of the fusion model are 0.8506 and 0.6627.

The recommendation accuracy of the state-of-the-art models among the proposed model is compared by using RMSE and MAE evaluation metrics, where Table 3 observed the performance of our proposed model along with the benchmark CF models such as the baseline CF model, baseline UCF, baseline ICF, UCF + Baseline CF, ICF + Baseline CF, TopkUCF, TopkICF, NormUCF, and NormICF. Our model achieves better performance under both the RMSE and MAE metrics on two datasets, which means that our blend model achieves a higher prediction accuracy and proves that the combination of UCF and ICF models with linear regression is effective.

Additionally, in Table 4 and Figure 4a and 4b we observe the performance of our blend proposed model compared with existing hybrid models such as the BiUCF [29], RSVD [35], ItemRank [36], iExpand [37], LDA [37], and AISCF [38]. Our model achieves better performance under both the RMSE and MAE metrics on two datasets, which means that our blend model achieves a higher prediction accuracy and proves that the combination of UCF and ICF models with linear regression is effective.

### Table 3
Root mean squared error (RMSE) and mean absolute error (MAE) comparison of the proposed model with different baseline methods on the 100k and 1M MovieLens dataset

| Model          | 100k MovieLens | 1M MovieLens |
|----------------|----------------|--------------|
|                | RMSE  | MAE  | RMSE  | MAE  |
| BCF            | 0.9648| 0.7617| 0.9346| 0.7349|
| BICF           | 1.0112| 0.8075| 1.0025| 0.7968|
| BUCF           | 1.0194| 0.8109| 0.9726| 0.7777|
| ICF + BCF      | 0.9352| 0.7377| 0.8960| 0.7047|
| UCF + BCF      | 0.9499| 0.7514| 0.9225| 0.7263|
| ICF + Bias     | 0.9350| 0.7373| 0.8959| 0.7069|
| TopkICF(\( k = 20 \)) | 0.9217| 0.7215| 0.8686| 0.6794|
| TopkUCF(\( k = 30 \)) | 0.9381| 0.7379| 0.9059| 0.7058|
| NorICF(\( k = 20 \)) | 0.9259| 0.7202| 0.8585| 0.6797|
| NorUCF(\( k = 30 \)) | 0.9461| 0.7399| 0.8925| 0.6938|
| ΓUICF(Γ = 0.70) | 0.9130| 0.7174| 0.8645| 0.6751|
| ΓUICF(Norm) (Γ = 0.70) | 0.9229| 0.7219| 0.8506| 0.6627|

### Table 4
According to the results obtained by Kant S and Mahara T. [29], here is the root mean squared error (RMSE) and mean absolute error (MAE) performance comparison of different methods with the proposed model.

| Models          | 100k MovieLens | 1M MovieLens |
|-----------------|----------------|--------------|
|                 | RMSE  | MAE  | RMSE  | MAE  |
| BiUCF [29]      | 1.001 | 0.786| 0.9378| 0.7379|
| RSVD [35]       | 1.042 | 0.798| 0.9410| 0.7421|
| ItemRank [36]   | 1.006 | 0.796| 0.9451| 0.7458|
| iExpand [37]    | 1.008 | 0.795| 0.9403| 0.7414|
| LDA [37]        | 1.050 | 0.825| 0.9423| 0.7424|
| AISCF [38]      | 1.060 | 0.843| 0.9381| 0.7385|
| ΓUICF           | 0.9130| 0.7174| 0.8645| 0.6751|
| ΓUICF(Norm)     | 0.9229| 0.7219| 0.8506| 0.6627|

### Conclusion and Future Work
We presented a hybrid recommendation model, namely ΓUICF. The ΓUICF model combines UCF and ICF linear regression models to model unknown user-item preferences. The ΓUICF model was implemented on two different real-world datasets, namely 100k MovieLens and 1M MovieLens datasets. Our results demonstrate the accuracy of the proposed recommendation model which has performed significantly with minimum error compared to existing CF models, which shows the outperformance of our approach compared to the existing CF models. The proposed model solved the limitations of CF algorithms; however, there is some shortage of CF models when dealing with real-world applications. It does not effectively address the well-known
cold start problem, that is, when a new user or new item enters the system. There is difficulty in dealing with sparse data, hence it is hard to find users that have scored the same items. It suffers when new users or items that do not have any ratings enter the system. It tends to recommend popular items. In future work, we plan to propose a recommendation model using another technique called the model-based CF technique based on deep learning methods with some features of the users’ and items’ data base to address the problems mentioned and minimise the prediction error in a real-world RS.

**ORCID**

Mohammed Fadhel Aljunid  
https://orcid.org/0000-0001-9099-3664

**REFERENCES**

1. Ge, Y., et al.: Understanding echo chambers in e-commerce recommender systems. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp. 2261–2270. X‘ian, China (July, 2020)
2. Amato, F., et al.: SOS: a multimedia recommender system for online social networks. Future Generat. Comput. Syst. 93, 914–923 (2019)
3. Vall, A., et al.: Feature-combination hybrid recommender systems for automated music playlist continuation. User Model User-Adapted Interact. 29(2), 527–572 (2019)
4. Aljunid, M.F., Manjaiah, D.H.: Movie recommender system based on collaborative filtering using Apache spark. In: Balas, V., Sharma, N., Chakrabarti, A. (eds.) Data Management, Analytics and Innovation. (Advances in Intelligent Systems and Computing), vol. 839, pp. 283–295. Springer, Singapore (September 2018)
5. Aljunid, M.F., Doddaghatta Huchaiha, M., Huchaiha, M.D.: Multi-model deep learning approach for collaborative filtering recommendation system. CAAI Trans. Intell. Technol. 5(4), 268–275 (2020)
6. Cleger-Tamayo, S., Fernández-Luna, J.M., Huete, J.F.: Top-N news recommendations in digital newspapers. Knowl. Base Syst. 27, 180–189 (2012)
7. Kim, H.K., et al.: Commenders: a recommendation procedure for online book communities. Electron Commer. Res. Appl. 10(5), 501–509 (2011)
8. Son, J., Kim, S.B.: Academic paper recommender system using multi-level simultaneous citation networks. Decis. Support Syst. 105, 24–33 (2018)
9. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 1–34. Springer, Boston (2015)
10. Dara, S., Chowdary, C.R., Kumar, C.: A survey on group recommender systems. J. Intell. Inf. Syst. 54(2), 271–295 (2020)
11. Chen, R., et al.: A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks. IEEE Access. 6, 64301–64320 (2018)
12. Ajoudanian, S., Abadeh, M.N.: Recommending human resources to project leaders using a collaborative filtering-based recommender system: case study of ghHul. JET Softw. 13(5), 379–385 (2019)
13. Aljunid, M.F., Di, M.: An efficient deep learning approach for collaborative filtering recommender system. Procedia Computer Science. 171, 829–836 (2020)
14. Jin, R., et al.: Collaborative filtering with decoupled models for preferences and ratings. In: Proceedings of the twelfth international conference on Information and knowledge management, pp. 309–316. New Orleans (November, 2003)
15. Sarwar, B., et al.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web, pp. 285–295. Hong Kong (May, 2001)
16. Kamishima, T.: Nanotac collaborative filtering: recommendation based on order responses. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 583–588. Washington (August, 2003)
17. Mandal, S., Maiti, A.: Explicit feedback meet with implicit feedback in GPME: a generalized probabilistic matrix factorization model for recommendation. Applied Intelligence, 50, 1955–1978 (2020)
18. Sun, Z., et al.: CFPS: Collaborative filtering based source projects selection for cross-project defect prediction. Applied Soft Computing, 99, 106940 (2021)
19. Feng, L., Zhao, Q., Zhou, C.: Improving performances of Top-N recommendations with co-clustering method. Expert. Syst. Appl. 143, 113078 (2020)
20. Neysiani, B.S., et al.: Improve performance of association rule-based collaborative filtering recommendation systems using genetic algorithm. Int. J. Inf. Technol. 11, 48–55 (2019)
21. Koudadia, A., Nouali, O., Al-Shamri, M.Y.H.: A multi-criteria collaborative filtering recommender system using learning-to-rank and rank aggregation. Arabian J. Sci. Eng. 45(4), 2635–2645 (2020)
22. Bamlaz, Z., et al.: A review on deep learning for recommender systems: challenges and remedies. Artif. Intell. Rev. 52(1), 1–37 (2019)
23. Bobadillas, J., et al.: Recommender systems survey. Knowl. Base. Syst. 46, 109–132 (2013)
24. FadhelAljunid, M., Manjaiah, D.H.: A survey on recommendation systems for social media using big data analytics. Int. J. Latest Trends Eng. Technol. 48–58 (2017)
25. Tso-Sutter, K.H., Marinho, L.B., Schmidt-Thieme, L.: Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: Proceedings of the 2008 ACM symposium on applied computing, pp. 1995–1999. Fortaleza, Ceara Brazil (March, 2008)

26. Wang, J., Yin, J.: Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity. In: 6th international conference on biomedical engineering and informatics Hangzhou, pp. 661–665. China (December, 2013)

27. Yamashita, A., Kawamura, H., Suzuki, K.: Adaptive fusion method for user-based and item-based collaborative filtering. Adv. Complex Syst. 14(02), 133–149 (2011)

28. Miyahara, K., Pazzani, M.J.: Improvement of collaborative filtering with the simple Bayesian classifier. Inf. Process. Soc. of Japan. 43(11) (2002)

29. Kant, S., Mahara, T.: Merging user and item based collaborative filtering to alleviate data sparsity. Int. J. Syst. Assur. Eng. Manag. 9(1), 173–179 (2018)

30. Slokorn, M., Ayachi, R.: A hybrid user and item based collaborative filtering approach by possibilistic similarity fusion. In: Hatzilygeroudis, I., Palade, V., Prentzas, J. (eds.), Advances in Combining Intelligent Methods. (Intelligent Systems Reference Library), vol. 116, pp. 125–147. Springer, Cham (2017)

31. Liu, H., et al.: A new user similarity model to improve the accuracy of collaborative filtering. Knowl. Base. Syst. 56, 156–166 (2014)

32. Koutrika, G., Bercovitz, B., Garcia-Molina, H.F.: Expressing and combining flexible recommendations. In: Proceedings of the 35th SIGMOD international conference on management of data (SIGMOD’09), vol. 29. Providence, RI, USA (June, 2009)

33. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363 (2013)

34. Jannach, D., Karakaya, Z., Gedikli, F.: Accuracy improvements for multi-criteria recommender systems. In: Proceedings of the 13th ACM conference on electronic commerce, pp. 674–689. Vale (June, 2012)

35. Funk, S.: Netflix update: try this at home (December 2006) (2006). https://sifter.org/simon/journal/20061211.html. Accessed May 2020

36. Pucci, A., et al.: A random-walk based scoring algorithm for recommender engines. In: Nasraoui, O. et al. (eds.), Advances in Web Mining and Web Usage Analysis. (WebKDD) Lecture Notes in Computer Science, vol. 4811, pp. 2766–2771. Springer, Berlin (2006)

37. Liu, Q., et al.: Enhancing collaborative filtering by user interest expansion via personalized ranking. IEEE Trans. Syst. Man Cybern. B (Cybernetics). 42(1), 218–233 (2011)

38. Chen, M.-H., Teng, C.-H., Chang, P.-C.: Applying artificial immune systems to collaborative filtering for movie recommendation. Adv. Eng. Inf. 29(4), 830–839 (2015)

How to cite this article: Aljunid MF, Huchaiah MD. An efficient hybrid recommendation model based on collaborative filtering recommender systems. *CAAI Trans. Intell. Technol.* 2021;1–13. [https://doi.org/10.1049/cit2.12048](https://doi.org/10.1049/cit2.12048)