IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2020, accepted October 15, 2020, date of publication November 3, 2020,
date of current version November 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035703

Improved Collaborative Filtering Recommendation Through Similarity Prediction

NIMA JOORABLOO[1], MAHDI JALILI[1], (Senior Member, IEEE), AND YONGLI REN[2]

[1]School of Engineering, RMIT University, Melbourne, VIC 3001, Australia
[2]School of Science, RMIT University, Melbourne, VIC 3001, Australia

Corresponding author: Mahdi Jalili (mahdi.jalili@rmit.edu.au)

This work was supported by the Australian Research Council under Project LP180101309.

ABSTRACT Collaborative Filtering (CF) approaches have been widely used in various applications of recommender systems. These methods are based on estimating the similarity between users/items by analyzing the ratings provided by users. The existing methods are often domain-specific and have not considered the time of the ratings being made in the calculation of the similarity. However, users' preferences vary over time, and so their similarity. In this paper, a novel method is proposed by re-ranking the users/items neighborhood set considering their future similarity trend. The trend of similarity is predicted, and depending on increased/decreased trend, we update the final nearest neighbor sets that are used in CF formulation. This method can be applied on a broad range of CF methods that are based on similarities between users and/or items. We apply the proposed approach on a set of CF algorithms over two benchmark datasets and show that the proposed approach significantly improves the performance of the original CF recommenders. As the proposed method only re-ranks the neighborhood set, it can be applied to any existing non-temporal similarity-based CF recommenders to improve their performance.

INDEX TERMS Collaborative filtering, recommendation system, sequential pattern, similarity measure, time, prediction.

I. INTRODUCTION

Providing personalized user experience is a critical issue for product/service providers on the Web. Recommendation systems (RSs) have been developed to deal with this problem by filtering unnecessary information and providing personalized content and service delivery to users. Global industry firms have applied RSs to predict the potential preferences of customers and recommend relevant products/services to them. This approach has improved the user experience and made a huge impact on their commercial success [1]. The existing research papers in the RSs field have mainly focused on movie recommendations [2], [3]. There are also some research works in other domains, such as e-commerce [4], books [5], documents [6], music [7], television programs [8], applications in markets [9], e-learning [10], social network [11], and Web search [12]. The approaches for recommendation can be classified as collaborative filtering (CF), content-based (CB), and hybrid [13] methods. CF is a widely used approach in RSs, which is based on users' past behavior and the way they have rated items [14]. The main idea for user-based CF is if two users have similar behavior in the past, e.g. have rated some items similarly, they will likely prefer similar items in the future. A user-based CFs consists of three main steps to recommend items which are shown in Fig. 1. These are: *i*) generating the similarity matrix which contains similarity between all users, *ii)* selecting top-N users with the highest similarity as a neighborhood set of a target user, and *iii)* recommending items to the target user from the list of those that highly liked by their neighbors. On the other hand, an item-based CF works based on the similarity between items calculated based on users' ratings of those items.

CF RSs rely on explicit data such as user ratings, or implicit data which is captured from the behavior of users such as viewing or purchasing an item. Along with using rating information, there is other valuable information that is not often considered in classical RSs, including time, device type, and location. Considering such extra information often improves the quality of recommendations [15]–[18]. Using sequences

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Luca Bernardi.

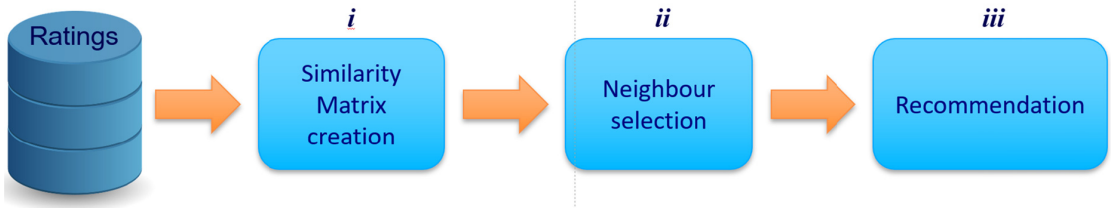

FIGURE 1. The main steps to recommend items by a user-based collaborative filtering recommender system.

of ratings and time information can be useful in improving the accuracy of recommendations. Often, users' behaviors and preferences vary over time [19]. Therefore, the time factor can play an important role in providing effective personalized recommendations and consequently improving the accuracy of predictions [17], [20]. Although several studies have been proposed to use temporal information to enhance the performance of the recommender [21], [22], there are still some gaps. First, the existing methods are often domain-specific and do not work across all domains [23]. Time information has not been considered in many of the domain-specific algorithms and proposing a universal method to add temporal information to them may improve their performance. Second, some algorithms have used a time-decay function to decrease the effect of old ratings by decaying their influence [24]–[26]. The issue is that, if users' preferences stay consistent over time for a particular type of items, old ratings related to that type may help to improve the accuracy of prediction; however, such information is lost in the conventional time decay-based algorithms [27]. Third, creating a high-performance and scalable recommendation system is not an easy task in the current era of Web. Typically, very specialized systems are developed to deal with the problem of high-quality recommendations on large datasets [28]. Conventional CF algorithms have the lowest computational complexity, which makes them suitable for large-scale systems, but they are not highly accurate.

To address the above issues, in this manuscript, a novel method is proposed to add valuable time information to similarity-based RSs. The proposed method can be applied to all similarity-based RSs available in the literature to improve the performance of their recommendations. Fig. 1 shows a schematic process of a user-based recommender system. Based on this, we can boost the recommendation performance by either improving the neighbor scoring function in the second step, or by the way this function is used in the recommendation process in the third step, or both [29]. Our proposed method only makes a change in the second step – neighborhood selection – while leaving other steps unchanged. Our algorithm tracks the trend of similarity changes between two users/items to predict whether their similarity will increase or decrease in the future. The algorithm then re-ranks neighbors for each target user/item using the predicted similarity trends so as to get a better neighborhood ranking, which leads to achieving better performance than the original RS. Our experiments on benchmark datasets show that the proposed method significantly improves the performance of many similarity-based CF recommenders.

The rest of the paper is as follows. Related studies are reviewed in Section II. The proposed method is introduced in section III, and section IV shows the experimental results by comparing the performance of some algorithms after and before applying the proposed method. Finally, we conclude our work in Section V.

II. RELATED WORKS

Temporal information of the ratings is useful metadata, which can help us to track changes in users' behavior and preferences over time [20]. Lee *et al*. performed an empirical study to show the important effect of temporal information on the performance of recommendations [30]. Several methods have been proposed in the literature to take into account the rating time for recommendations. Some of these methods use a time-decay function to decrease the effect of old ratings in the recommendation process. Ding and Li assigned a weight to each user's rating based on an exponential time-decay function [26]. However, not all recent data are more important than old ones, and if the users' preference stays consistent over time for a particular type of item, neglecting old ratings related to that type may negatively impact the recommendation accuracy [27].

Zimdars *et al*. first sorted data based on time and then used a decision-tree learning model for recommendation [31]. Ricci and Nguyen considered users' long-term preferences using their past interactions and let users explicitly define a set of stable preferences [32]. The temporal factorization model was used to model the historical data in [33] by Koren to predict ratings of movies in the Netflix dataset. They recognized that users' bias and preference change over time, and proposed an RS by incorporating temporal information into an item–item neighbor modelling. Tang *et al*. improved the performance of the CF recommender system using movie production years and scaling down candidate sets [34]. The Recommendation accuracy was improved in [35] by considering purchase time and also launch time of items. Karahodza *et al*. proposed a user-based CF algorithm over movie datasets by considering temporal contextual information, which led to an increase in accuracy [36]. The weight function proposed in their study is based on changes in the group user's preferences over time. In [37], Xia *et al*. redefined the item-based similarity in a different way based on time-decay and proposed a top-N item-based RS which recommended items dynamically.

Lathia *et al*. proposed a time-based method to update neighborhood sizes automatically instead of setting a fixed size [38]. They formalized the problem as a time-dependent, iterative prediction problem and performed a temporal analysis of the Netflix dataset. They showed that due to the dynamic nature of the data, a certain algorithm with accurate predictions on the Netflix dataset did not show the same behavior with growing data. Chen *et al*. showed that users' interests and the popularity of topics shift very fast in online social platforms. They used an online ranking technique to propose a topic recommender system to provide the right topics (hashtags) at the right time for Twitter users [39]. Zheng *et al*. used opinion-mining technology and SVD++ to propose a tourism destination recommender system that considered user sentiment, and employed a temporal dynamics to represent changes of user preference in destinations over time [40]. A taxi recommender system for determining the next cruising location was proposed in [41], which considered the distance between the current location and the recommended location, waiting time for the next passengers, expected fare for the trip and the most likely location to pick up passengers from drivers' points of view.

III. RECOMMENDATION BASED ON IMPROVED SIMILARITY ESTIMATION THROUGH SIMILARITY PREDICTION

In this section, a novel domain-independent recommendation method is proposed based on predicting the similarity trend between users/items. The proposed method can be applied on any existing similarity-based CF recommenders. Although this method works for both user-based and item-based CFs, here we explain the methodology for user-based CF, and one needs a similar adoption to use it for item-based CF. The proposed method works based on rating sequences and predicting the trend of user-user similarity in the future. To this end, users' ratings are first transferred into a formal sequential model, and then a time-window is applied on the sequences to calculate the similarity between users in different time-windows. Having this, a time-series of user-user similarity values are obtained instead of single similarity value. The next step is to obtain the similarity trend using these time series, and then to determine whether a particular similarity value will increase or decrease in the future. After that, we re-rank users' neighborhood sets which are used in the recommendation process using the predicted trends. In the following, we provide details of the above steps.

A. SEQUENTIAL PATTERN REPRESENTATION

To effectively consider the sequence of ratings, a formal representation model for sequential patterns is introduced. This model is based on the order of items rated by the users. Let us consider I as the set of items, U as the set of users, and R as the set of ratings in the system. $S_u = <x_1, x_2, \ldots, x_l>$ represents the sequence of rating for target user u where each x_i in this sequence is denoted as (i, r). $i \in I$ is an item rated by target user u, and r is its corresponding rating value. Items of a sequence are sorted based on time of ratings in ascending order. As an example, for a simple user-based CF, suppose that the history of ratings for two users u and v is like Table 1. The sequential pattern for users u and v are represented by $= < (i_3, 5), (i_5, 2), (i_{30}, 4), (i_1, 1), (i_{13}, 1), (i_{41}, 2), (i_5, 3), (i_9, 3), (i_2, 4) >$ and $S_v = < (i_{22}, 1), (i_{22}, 1), (i_8, 4), (i_6, 4), (i_1, 2), (i_9, 3), (i_2, 5) >$, respectively.

TABLE 1. Example rating history for users *u* and *v*.

User	Item	Rating	Time
u	i_3	5	t_1
	i_5	2	t_2
	i_{30}	4	t_3
	i_1	1	t_4
	i_{13}	1	t_5
	i_{41}	2	t_6
	i_7	3	t_7
	i_9	3	t_8
	i_2	4	t_9
v	i_7	1	t_1
	i_8	4	t_2
	i_6	4	t_3
	i_1	2	t_4
	i_9	3	t_5
	i_2	5	t_6

B. PREDICTING SIMILARITY TREND

Algorithms based on user similarity, first extract the neighborhood list for each target user by finding the most similar users, i.e., a subset of users is chosen with the highest similarities as the nearest neighbors for the target user.

In its simplest way, a similarity value between user u and v is calculated by calculating the similarity between their ratings history. The existing CF algorithms have proposed various ways to calculate the similarity between users, and Cosine measure [42] is one of the simplest one which is function as follows:

$$Sim\,(u, v) = \frac{\sum_{i\in(I_u\cap I_v)} \left(r_{u,i}\right)\left(r_{v,i}\right)}{\sqrt{\sum_{i\in(I_u\cap I_v)} \left(r_{u,i}\right)^2}\sqrt{\sum_{i\in(I_u\cap I_v)} \left(r_{v,i}\right)^2}}, \quad (1)$$

where $r_{u,i}$ is the rating given by user u to item i and I_u is the sequence of items in S_u. $I_u \cap I_v$ is the set of items which are rated by both users u and v. The user-user similarity matrix is used to select the set of neighbors for each target user, which are those with the highest similarity values to the target user.

In the proposed method, a time-series of user-user similarity values are obtained instead of single similarity value. This time-series is obtained by calculating the similarity of S_u partialized with respect to S_v. To this end, we define a dynamic time-window with an initial size of w. Then, we calculate the similarity between u and v based on the items in S_v and the first w items in S_u. We iteratively increase the size of the time-window and calculate the similarity between u and v until all items in S_u are covered. In fact, we expand the size of the time-window step-by-step to involve more

recent items of u in calculating similarity. As a result, we have a time-series of similarity values with length of l for user u and v, which is denoted by:

$$ST\,(u,v) = \{Sim\,(u,v)_1, Sim\,(u,v)_2, \ldots., Sim\,(u,v)_l\}, \quad (2)$$

where l is the number of iterations to cover all items in S_u and calculated as follow:

$$l = \left\lceil \frac{|I_u|}{w} \right\rceil \quad (3)$$

and $Sim\,(u,v)_p$ is the similarity value between u and v in the p-th iteration and calculated as follow:

$$ST\,(u,v)_{p\in\{1,\ldots,l\}} = \frac{\sum_{i\in(I_u^{(1:d)}\cap I_v)}\left(r_{u,i}\right)\left(r_{v,i}\right)}{\sqrt{\sum_{i\in(I_u^{(1:d)}\cap I_v)}\left(r_{u,i}\right)^2}\sqrt{\sum_{i\in(I_u^{(1:d)}\cap I_v)}\left(r_{v,i}\right)^2}} \quad (4)$$

In above, d is the size of the time-window in p-th iteration which is obtained by Eq. 5, and $I_u^{(1:d)}$ is the subset of early items in S_u with length of d.

$$d = \begin{cases} p * w, & if\ p < l \\ lenght\,(I_u)\,, & if\ p == l \end{cases} \quad (5)$$

$Sim\,(u,v)_l$, the last value in $ST\,(u,v)$, is generated based on the total available ratings of u and v, which is indeed the similarity value calculated by the original Cosine algorithm. For some of the similarity measures like Cosine and Jaccard, we can calculate the similarity values accumulatively, and it does not need to calculate them several times from scratch. In fact, calculations for each iteration can be used to calculate the next value in the time-series, which helps us to prevent a significant increase in the algorithm complexity. It is obvious that by selecting smaller time-windows, the complexity of the proposed algorithm may significantly increase. A pseudo-code for the proposed process is shown in Algorithm 1 where *simFunction* is the similarity function of RS.

Algorithm 1 Obtaining Time-Series of Similarities Between User u and v

1. **Input:** S_u, S_v, w
2. **Output:** $ST\,(u,v)$
3. $ST\,(u,v) \leftarrow []$
4. $p \leftarrow 1$
5. $d \leftarrow 0$
6. $l \leftarrow ceil(lenght\,(I_u)\,/w)$
7. while $(p <= l)$
8. if $(p == l)$
9. $d \leftarrow lenght\,(I_u)$
10. else
11. $d \leftarrow p * w$
12. end if
13. $Sim_p\,(u,v) \leftarrow simFunction(I_u^{(1:d)}, I_v)$
14. Insert $Sim_p\,(u,v)$ into $ST\,(u,v)$
15. $p \leftarrow p+1$
16. end while

Fig. 2 depicts the process of generating $ST\,(u,v)$ over different time-windows for the example of Table 1. In this example, the initial size of the time-window is set to 3, and Cosine correlation is used as a similarity function. As a result, we have three similarity values, one for each time window: $Sim_1\,(u,v) = 0$, $Sim_2\,(u,v) = 0.11$, and $Sim_3\,(u,v) = 0.56$. these values are similarity values for user u partialized to user v. Using these values, a time-series of similarity is created as $ST\,(u,v) = <0, 0.11, 0.56>$.

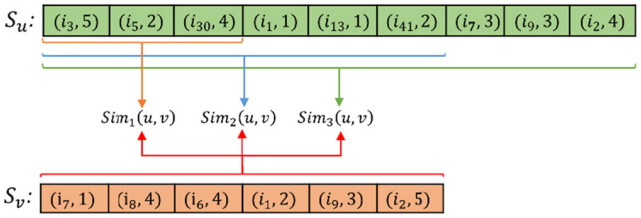

FIGURE 2. Process of generating trend of similarity $ST\,(u,v)$ for user u partialized to user v over different time-windows.

C. OBTAINING SIMILARITY TREND BETWEEN USERS

Time series forecasting is to predict the future trend using historical data [43]. In this paper, we aim at predicting whether the trend of similarity between users will increase or decrease in the near future. This is indeed one step ahead prediction. However, we often do not have many data points in our time-series to use sophisticated prediction methods. Therefore, this limits the choice of prediction methods. Many sophisticated methods require enough data points to create a prediction model with reasonable accuracy, which is not the case here. Often under limited data constraint, testing, and validation processes are hard if not impossible. Any model with more than one or two parameters often produces poor forecasts due to the overfitting problem and the estimation error [44]. Linear Regression is one of the most widely known techniques in learning predictive modelling [45]. In this technique, the dependent variable is continuous, the independent variable(s) can be continuous or discrete, and the nature of the regression line is linear. In the general problem of linear regression, assume that we want to model some observed data point D in a linear function. D is denoted as follow:

$$D = \{(X_1, Y_1)\,, (X_2, Y_2)\,, \ldots, (X_n, Y_n)\}, \quad (6)$$

where the X_i's are independent and defined on an interval λ and the Y_i's are generated by the regression model,

$$Y_i = \beta_0 + \int_\lambda \beta_1 X_i + \varepsilon_i \quad (7)$$

Here, β_0 is the intercept which is a constant number, β_1 is a square-integrable function on λ denoting the slop function, and ε is error. The best-fit line can be easily obtained by Least Square method, which works by minimizing the sum of the squares of the vertical deviations from each data point to the line. Here, we predict the trend of similarity by calculating β_1

which is the slope of the fitted line. For each user pair u and v, first we define data points D as follows:

$$D(u,v) = \{(w_1, ST(u,v)_1), (w_2, ST(u,v)_2), \ldots, (w_l, ST(u,v)_l)\}, \quad (8)$$

where $ST(u,v)_p$ is the obtained similarity by considering the p-th time-window and w_p is its position in sequence $ST(u,v)$ with length l. We denote the slop of fitted line to $D(u,v)$ data points by *SimTrend(u,v),* which indeed indicates the future trend of the similarity between u and v. *SimTrend(u,v)* shows whether the similarity trend is upward or downward. The downward trend shows that the similarity will likely decrease in the future, while the upward trend indicates increased similarity in the future. We do this process for all pairs of users in the system. As a result, we have *SimTrend,* which is a user-user matrix of similarity trends in the system.

In the example shown in Table 1, we use the classical user-based CF with similarity values obtained by Cosine correlation. Fig. 3 shows the proposed method to obtain the future trend of similarity between users u and v. In this example, *SimTrend(u,v)* is positive and takes an upward trend. Therefore, based on our proposed method, we predict that their future similarity, $Sim_f(u,v)$, will increase.

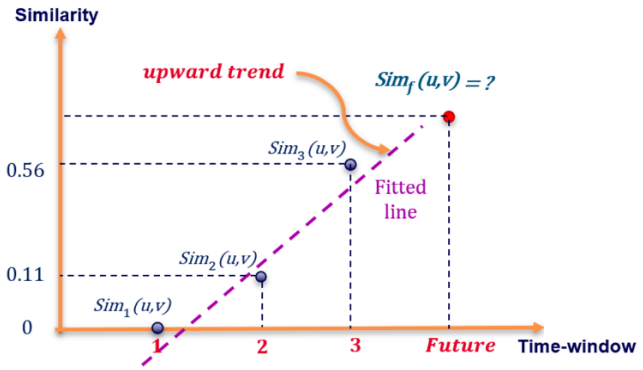

FIGURE 3. Obtaining The similarity trend for users u and v over a time-window with length three.

D. OBTAINING THE NEAREST NEIGHBORHOOD SET

The aim of this step is to re-rank users in the nearest neighborhood set using the similarity trend obtained in the previous step. In a similarity-based RS, the neighborhood list for each target user is extracted by finding its most similar users. Thus, a subset of users with the highest similarities is chosen as the nearest neighbors, which are then used to predict ratings for items not yet rated by the target user. Our idea here is that users with upward similarity could be better neighbors than those with downward trends. Thus, we strengthen (weaken) upward (downward) similarities.

With these assumptions, the similarity between two users in terms of trend and value can be categorized in one of following classes: *i)* upward and positive (*UP*), *ii)* upward and negative (*UD*), *iii)* downward and positive (*DP*) and *iv)* downward and negative (*DN*). Our hypothesis is that the most similar users to a target user, in order of preference are *UP, UD, DP* and then *DN*. Each item in these lists has a membership degree which is stored in *WUP, WUD, WDP,* and *WDN*, respectively. The membership degree for users u and v is equal to their original similarity value, the one that is stored in $Sim(u,v)_l$.

In the next step, we obtain the final ranking list of neighbors of target user u, denoted by $neighbor_u$. To this end, first we sort the items in each class based on their importance weight and obtain $neighbor_u$ by concatenating lists from higher priority to lower ones as follow:

$$neighbor_u = \text{Concatenate}(UP_u, UD_u, DP_u, DN_u), \quad (9)$$

where UP_u, UD_u, DP_u, DN_u are classes for user u. The main difference between classical methods and our method is the way we rank neighbors. While classical methods rank users based on the original similarity value, our proposed method ranks them regarding their future similarity trend and also their similarity values.

E. RECOMMENDATION

The last step is to use the updated neighborhood to provide recommendation lists to target users. In the proposed user-based RS, first the neighborhood list for each target user u is selected from *Top_P* items in $neighbour_u$ and denoted by K_u. In the next step, they are used to predict ratings for items not yet been rated by u. The user-based CF obtains the predicted rating of item j for the target user u as:

$$P_{u,j} = \bar{r}_u + \frac{\sum_{e \in K_{u,j}} Sim(u,e)_l * (r_{e,j} - \bar{r}_u)}{\sum_{u \in N_{e,j}} Sim(u,e)_l}, \quad (10)$$

where $K_{u,j}$ is the set of users in K_u who have rated item j, $\bar{r}_u$ is the average ratings made by user u and $Sim(u,e)_l$ is the last similarity value in $ST(u,e)$.

Note that we do not change anything with the recommendation process but only re-rank the similarity values based on the predicted trend of the similarity. This results in choosing more effective neighbors for target users leading to improved performance of recommendations. The same approach can be applied to item-based CF. To this aim, the similarity between items is calculated in different time-window, and then similarity trends between items are used to update neighborhood sets. In the last step, after predicting the rates of the unseen items for the active user, the algorithm selects those of the *Top_N* items to be recommended.

IV. EXPERIMENTAL RESULTS

In this section, we study the performance of some classical and state-of-the-art CF recommendation algorithms using the proposed update rule for the similarity. To evaluate the recommendation results, we split the ratings into a training set, and a test set. Time-unaware RSs mostly split data randomly,

however in time-aware RSs the order of ratings is important [46]. To perform the experiments in this work, first all ratings are sorted according to their timestamps in an ascending order. Then, 20% of the most recent ratings are selected for testing and the rest are taken as the training set. Similar approach has already been used in previous time-aware RSs, e.g. [18], [19], [47]–[49].

A. BENCHMARK ALGORITHMS

We consider the following benchmark algorithms:

1) Cosine (COS) [42]: Cosine is one of the most popular similarity measures for CF RSs to evaluate how much two users are correlated. The Cosine similarity between is calculated by (1).
2) Pearson Correlation Coefficient (PCC): Pearson is an improved version of Cosine measure by adding an average of ratings to calculations. Pearson Correlation reflects the similarity between users more accurate than Cosine and is one of the most popular measures used in classical similarity-based CF RSs [50].

$$PCC\ (u,v) = \frac{\sum_{i\in I_{u,v}} \left(r_{u,i}-\bar{r_u}\right)\left(r_{v,i}-\bar{r_v}\right)}{\sqrt{\sum_{i\in I_{u,v}} \left(r_{u,i}-\bar{r_u}\right)^2}\sqrt{\sum_{i\in I_{u,v}} \left(r_{v,i}-\bar{r_u}\right)^2}}, \tag{11}$$

where $\bar{r_u}$ is the average rating of user u for co-rated items represented by set I.

3) Spearman Rank Correlation (SRC): is a statistical measure of the strength of a monotonic relationship between paired data [51]. The following formula is used to calculate the Spearman Rank Correlation:

$$SRC\ (u,v) = 1 - \frac{6 * \sum_{i\in I_{u,v}} d_{u,v}^2}{n\left(n^2-1\right)}, \tag{12}$$

where $d_{u,v}$ is the difference between the ranks of corresponding variables, and n is the number of items in common itemset I.

4) Jaccard: that takes into account the number of common preferences between two users. Jaccard does not consider the rating values but the number of items being rated. Two users are more similar if they have more commonly rated items [52].

$$J\ (u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}, \tag{13}$$

5) Jaccard-Pearson Correlation Coefficient (JPCC): this measure calculates the combination of Jaccard and Parson Correlation Coefficient to get better similarity value. JPCC is obtained as follow:

$$JPCC\ (u,v) = Jac\ (u,v) * PCC\ (u,v), \tag{14}$$

6) Constrained Pearson Correlation Coefficient (CPCC): it is the same as PCC, but instead of the average ratings co-rated by both users, it uses the median value for rating scale [53].

$$CPCC\ (u,v) = \frac{\sum_{i\in I_{u,v}} \left(r_{u,i}-r_{med}\right)\left(r_{v,i}-r_{med}\right)}{\sqrt{\sum_{i\in I_{u,v}} \left(r_{u,i}-r_{med}\right)^2}\sqrt{\sum_{i\in I_{u,v}} \left(r_{v,i}-r_{med}\right)^2}}, \tag{15}$$

where r_{med} is the median value for the rating scale.

7) Jaccard-Constrained Pearson Correlation Coefficient (JCPCC): obtains similarity between two users using the combination of Jaccard and Constrained Pearson Correlation Coefficient, which is calculated as follows:

$$JCPCC\ (u,v) = Jac\ (u,v) * CPCC\ (u,v), \tag{16}$$

8) Jaccard-Spearman Rank Correlation (JSRC): obtains similarity between two users using the combination of Jaccard and Spearman Rank Correlation which is calculated as follows:

$$JSRC\ (u,v) = J\ (u,v) * SRC\ (u,v), \tag{17}$$

9) Jaccard-Cosine (JCOS): calculates the similarity between users with combining Jaccard Cosine Correlation which is calculated as follows:

$$JCOS\ (u,v) = J\ (u,v) * COS\ (u,v), \tag{18}$$

10) Weighted Pearson Correlation Coefficient (WPCC): the size of the common item set is not taken into account in the classical Pearson correlation coefficient. To resolve this issue, Weighted Pearson Correlation Coefficient has been proposed [54]. If two users have fewer than F commonly rated items, a weight $w_{u,v}$ is applied to their correlation. If there are more than F co-rated items, $w_{u,v}$ is set to one. $WPCC$ for users u and v is calculated as follows:

$$WPCC\ (u,v) = \frac{\sum_{i\in I_{u,v}} w_{u,v} * \left(r_{u,i}-\bar{r_u}\right)\left(r_{v,i}-\bar{r_v}\right)}{\sqrt{\sum_{i\in I_{u,v}} w_{u,v} * \left(r_{u,i}-\bar{r_u}\right)^2}\sqrt{\sum_{i\in I_{u,v}} w_{u,v} * \left(r_{v,i}-\bar{r_u}\right)^2}}, \tag{19}$$

where $w_{u,v}$ is obtained as follows:

$$w_{u,v} = \begin{cases} \frac{n}{F}, & \textit{if correlated items is less than } F \\ 1, & \textit{otherwise}, \end{cases} \tag{20}$$

where n is the number of co-rated items.

11) Proximity-Impact-Popularity (PIP): This similarity measure considers three different aspects which are proximity (the simple arithmetic difference between two ratings), impact (how strongly an item is preferred or disliked by buyers) and popularity (giving bigger value to similarity for ratings that are further from the average rating of a co-rated item) [55]. Finally, PIP similarity is obtained by merging them as follows:

$$PIP\ (u,v) = proximity\ (u,v) * impact\ (u,v) * popularity\ (u,v) \tag{21}$$

12) New heuristic similarity model (NHSM): This similarity measure considers the fact that different users have different preference scale [56]. NHSM is a combination of JPSS and User Rating Preference URP) similarity measures, which is based on mean and variance of the users' rating. NHSM values range is between 0 to 1 and calculated as follows:

$$NHSM\ (u, v) = JPSS\ (u, v) * URP\ (u, v) \tag{22}$$

13) Adjust Cosine (ACOS): This similarity measure considers the difference in the rating scale used by each user [55]. ACOS similarity subtracts the average rating of user u for all the items rated by user u and is obtained as follows:

$$ACOS\ (u, v) = \frac{\sum_{i \in I_{u,v}} \left(r_{u,i} - \bar{r_u}\right)\left(r_{v,i} - \bar{r_v}\right)}{\sqrt{\sum_{i \in I_{u,v}} \left(r_{u,i} - \bar{r_u}\right)^2} \sqrt{\sum_{i \in I_{u,v}} \left(r_{v,i} - \bar{r_u}\right)^2}}, \tag{23}$$

where $\bar{r_u}$ is the average rating of all items rated by user u.

14) Mean Squared Difference (MSD): considers the absolute ratings to calculate the similarity [56]. MSD is obtained as follow:

$$MSD\ (u, v) = 1 - \frac{\sum_{i \in I_{u,v}} \left(r_{v,i} - r_{u,i}\right)^2}{|I|} \tag{24}$$

15) Multi-level Collaborative Filtering (MLCF): divides the similarity calculated by PCC into different levels and improves the accuracy of the recommendations by adding constraints to each level [57]. In this algorithm, predefined thresholds (t_1, t_2, t_3, t_4) represent the constraints on the number of co-rated items for each level. The similarity between users in this method is calculated as follow:

$$MLCF\ (u.v) = \begin{cases} PCC\ (u, v) + x, & if\ \frac{|I|}{T} \geq t_1 \\ and\ PCC\ (u, v) \geq y & \\ PCC\ (u, v) + x, & if\ \frac{|I|}{T} < t_1 \\ and\ \frac{|I|}{T} \geq t_2\ and\ PCC\ (u, v) \geq y & \\ PCC\ (u, v) + x, & if\ \frac{|I|}{T} < t_2 \\ and\ \frac{|I|}{T} \geq t_3\ and\ PCC\ (u, v) \geq y & \\ PCC\ (u, v) + x, & if\ \frac{|I|}{T} < t_3 \\ and\ \frac{|I|}{T} \geq t_4\ and\ PCC\ (u, v) \geq y & \\ 0, & otherwise, \end{cases} \tag{25}$$

where T is the total number of co-rated items, x and y are real positive numbers which are different at each level.

16) Jaccard Mean Squared Difference (JMSD): This combines MSD and Jaccard to obtain a new similarity measure. It is defined as follows [58]:

$$JMSD\ (u, v) = J\ (u, v) * MSD\ (u, v) \tag{26}$$

17) Sigmoid function based on Pearson Correlation Coefficient (SPCC): is a modified form of PCC [59]. This measure is an exponential version of PCC, and users with a smaller number of co-rated items have a weaker similarity. $SPCC$ is obtained as follow:

$$SPCC\ (u, v) = PCC\ (u, v) * \frac{1}{1 + \exp(-\frac{|I|}{2})} \tag{27}$$

18) Item-based CF (ICF): is a form of collaborative filtering that works based on the similarity between items [60]. Similarities between items are calculated using one of several similarity measures in the literature like Cosine. Then these similarity values are used to predict ratings for unseen items of a target user.

The details of the experiments are presented in the following sections.

B. DATASET

Movielens and Goodreads are two well-known benchmark datasets used in the experiments to verify the effectiveness of the proposed method [61], [62]. Movielens dataset contains 1682 movies, 943 users, and 100,000 different ratings. Goodreads dataset contains ratings of nearly 47,000 users over around 2 million books. We randomly chose 800 anonymous users for our experiment, who rated more than 40 books. The total number of books rated by these users is about 83,000 titles, and the total number of ratings is 126,881. The rating values are integer numbers in the range of 1 (bad) to 5 (excellent). While rating density in Movielens dataset is 6.30%, Goodreads dataset is heavily sparse with a rating density of only 0.18%. Applying the proposed method on these datasets allow us to evaluate the performance on dataset with different densities.

C. EVALUATION METRICS

1) PRECISION

Precision metric evaluates the accuracy of recommendations. Precision of recommending N items to user u is denoted by $P_u(N)$ and defined as the percentage of relevant items in the recommendation list of user u. Precision of a system that recommends N items to its users is shown by with $P(N)$ and is calculated as:

$$P\ (N) = \frac{\sum_{u \in TestSet} P_u\ (N)}{|u_{TestSet}|}, \tag{28}$$

where $TestSet$ is the part of the dataset that we assume as a test set.

2) RECALL

Recall for a target user u is the number of relevant recommended items to the total number of relevant items and is

TABLE 2. Comparing the performance of original and modified version of algorithms using proposed method over Movielens. Note that in the proposed algorithm we re-rank users' neighborhood sets using the predicted similarity trends.

	Precision			*Recall*			*F1*		
	Original	Proposed	Improvement %	Original	Proposed	Improvement %	Original	Proposed	Improvement %
PCC	0.3080	**0.4302**	39.68	0.4320	**0.5587**	29.33	0.3401	**0.4306**	26.61
CPCC	0.4977	**0.4993**	0.32	0.6697	**0.6739**	0.63	0.4925	**0.4946**	0.43
WPCC	0.5339	**0.5349**	0.19	0.7276	**0.7279**	0.04	0.5253	**0.5258**	0.10
JMSD	0.4853	**0.4915**	1.28	0.6353	**0.6465**	1.76	0.4808	**0.4871**	1.31
MSD	0.3655	**0.4477**	22.49	0.4887	**0.5785**	18.38	0.3810	**0.4452**	16.85
COS	0.3006	**0.4087**	35.96	0.4243	**0.5249**	23.71	0.3361	**0.4136**	23.06
SRC	0.2273	**0.2593**	14.08	0.3802	**0.4047**	6.44	0.3097	**0.3257**	5.17
JSRC	0.2291	**0.2863**	24.97	0.3768	**0.4191**	11.23	0.3088	**0.3335**	8.00
JCPCC	0.5098	**0.5140**	0.82	0.6674	**0.6732**	0.87	0.4987	**0.5030**	0.86
JCOS	0.4936	**0.4957**	0.43	0.6464	**0.6499**	0.54	0.4853	**0.4856**	0.06
JPCC	0.5035	**0.5090**	1.09	0.6588	**0.6671**	1.26	0.4908	**0.4970**	1.26
SPCC	0.4467	**0.4716**	5.57	0.6083	**0.6337**	4.18	0.4554	**0.4733**	3.93
NHSM	0.5202	**0.5213**	0.21	0.6969	**0.6973**	0.06	0.5134	**0.5152**	0.35
PIP	0.5366	**0.5392**	0.48	0.7261	**0.7299**	0.52	0.5270	**0.5296**	0.49
ACOS	0.3164	**0.4344**	37.29	0.4401	**0.5607**	27.40	0.3461	**0.4334**	25.22
MLCF	0.3091	**0.4294**	38.92	0.4328	**0.5584**	29.02	0.7363	**0.9091**	23.47
ICF	0.1129	**0.1835**	62.53	0.0946	**0.1538**	62.58	0.1029	**0.1673**	62.59

denoted by $Recall_u$. Recall for a system with M users is defined as follow:

$$Recall = \frac{\sum_{u \in testSet} Recall_u}{M} \tag{29}$$

3) F1 SCORE

Since recall and precision metrics are inversely correlated and are also dependent on the number of recommended items, researchers have often used the combination of them to evaluate RSs:

$$F1 = \frac{2 * P(N) * Recall}{P(N) + Recall} \tag{30}$$

D. RESULTS

As mentioned, to perform the experiments in this work, first, the ratings are sorted based on the time, and then the first 80% of the ratings are selected as the training set, and the remaining data is used as the test set. To generate the result for the proposed method, the initial size of time-window is set to 20. With this setting, the average number of iterations to compute the time-series for each pair of users in Movielens and Goodreads dataset are 4.24 and 6.33, respectively. As it mentioned earlier, it does not need to calculate the similarity measures, the obtained value in each iteration can be used to calculate the value for the next iteration. This approach prevents the time complexity of the algorithm blowing out. Our experiments show that the runtime of the proposed algorithm is at most three times more than the original algorithm.

The results of the experiments for Movielens and Goodreads dataset are reported in Tables 2 and Table 3, respectively. As one can see from these results, the proposed method significantly improved the classical algorithms and outperformed the original methods over these datasets, especially in Movielens. For example, some algorithms, such as MLCF, PCC, ICF, experienced more than 30% improvements. Although classical algorithms such as COS, PCC, and ICF have the worse results in terms of precision, by applying the proposed method, their precision improved about 36%, 40% and 63%, respectively. Moreover, these algorithms experienced 24%, 29% and 63% improvement, respectively in terms of recall. From Table 2 we can see that all modified version of algorithms outperformed their original ones in terms of F1 metric. For example, the performance of MLCF and ACOS in terms of F1 was improved by 25% and 23%, respectively.

Table 3 compares the performance of the original algorithms and their improved versions over Goodreads dataset. The highest improvements in terms of precision are in MSD and ACOS algorithms. Besides, the results also show that the modified version of algorithms surpassed their original ones over recall and F1 metrics; JMSD and JPCC are two good examples that experienced about 10% improvement in terms of recall. Compared with original ICF and MSD,

TABLE 3. Comparing the performance of original and improved algorithms on Goodreads dataset.

	Precision			*Recall*			*F1*		
	Original	Proposed	Improvement %	Original	Proposed	Improvement %	Original	Proposed	Improvement %
PCC	0.1333	**0.1481**	11.1	0.0997	**0.1111**	11.43	0.1582	**0.1703**	7.65
CPCC	0.259	**0.262**	1.16	0.1646	**0.1673**	1.64	0.2396	**0.2427**	1.29
WPCC	0.2714	**0.273**	0.59	0.1766	**0.1779**	0.74	0.2489	**0.2512**	0.92
JMSD	0.2047	**0.2166**	5.81	0.1252	**0.1386**	10.7	0.2034	**0.2151**	5.75
MSD	0.1415	**0.1589**	12.3	0.1035	**0.1139**	10.05	0.1665	**0.1842**	10.63
COS	0.1254	**0.135**	7.66	0.0941	**0.1007**	7.01	0.1528	**0.1626**	6.41
SRC	0.1217	**0.1272**	4.52	0.0914	**0.0943**	3.17	0.1497	**0.1545**	3.21
JSRC	0.1262	**0.1292**	2.38	0.085	**0.0916**	7.76	0.1561	**0.1607**	2.95
JCPCC	0.2565	**0.2672**	4.17	0.1586	**0.1686**	6.31	0.236	**0.2469**	4.62
JCOS	0.2483	**0.2626**	5.76	0.1561	**0.1684**	7.88	0.2252	**0.2372**	5.33
JPCC	0.2198	**0.234**	6.46	0.1353	**0.1485**	9.76	0.213	**0.2221**	4.27
SPCC	0.2068	**0.2094**	1.26	0.1416	**0.1426**	0.71	0.213	**0.2161**	1.46
NHSM	0.2921	**0.2935**	0.48	**0.**1926	**0.1933**	0.36	0.2553	**0.2574**	0.82
PIP	0.2905	**0.2963**	2.00	0.1875	**0.1939**	3.41	0.2588	**0.265**	2.4
ACOS	0.1322	**0.1479**	11.88	0.0961	**0.1085**	12.9	0.1624	**0.1755**	8.07
MLCF	0.1777	**0.1936**	8.95	0.126	**0.1403**	11.35	0.3206	**0.3465**	8.08
ICF	0.1374	**0.1536**	11.79	0.0515	**0.0638**	23.88	0.0749	**0.0883**	17.89

our proposed model obtained 18% and 11% better performance over F1.

As one can see from these results, we have different levels of improvement for different similarity measures. Indeed, the quality of collaborative filtering recommendations is related to how the similarity metrics are defined, e.g. how complete they can describe the relationship between users. For example, PCC takes the average of ratings made by the user into consideration, but COS take each user as a vector without considering their average of ratings. On the other hand, the performance of similarity metrics and recommendations is highly dependent on the problem parameters such as number of users, number of items, sparsity level, etc., [63]–[65]. Mentioned parameters vary according to the size of the time-window, which affects the calculated similarity value in each iteration, and consequently causes various improvement levels at the time of using different similarity measures. Unlike many state-of-the-art recommendation methods providing better accuracy over classical CF algorithms, our proposed method does not significantly increase the computational complexity of the original algorithm and also don't change their similarity calculation and recommendation proposed methods is the update provided on the neighborhood selection step. This is an important issue for online RSs in particular, in which one has to deal with billions of uses, items, and ratings. process. The only difference between the original and in some cases, one has also to deal with lots of changes in a short period of time to keep the recommendation up-to-date for customers. Simple CF algorithms, e.g. classical user-based CF, has found the greatest success in real large-scale datasets, and the proposed method can be simply integrated with them without significantly increasing computational burden on them. Our experimental results showed that applying the proposed method on simple and basic algorithms can add valuable temporal information to them, and consequently lead to significant performance improvement. Furthermore, unlike many other temporal methods in the literature, the proposed method is not domain-specific and according to the experiment results, it can be applied to any similarity-based CF recommendation algorithm.

V. CONCLUSION

In this paper, a novel recommendation approach is proposed to effectively consider temporal information of ratings provided by the users in the recommendation process. To this end, first the ratings are converted to a sequential pattern based on the time of the rating. Then, a time-series of similarities between a pair of users over time is generated, which is used to predict the trend of similarity between them in the future.

In a similarity-based RS, a subset of users/item with the highest similarities is first selected as the nearest neighbors for each user/item in the dataset, and then are used in the recommendation process. It is a fact that users' preferences vary over time, and so their similarity. Although several studies have been proposed to use temporal information in similarity-based RSs, these methods are often domain-specific. Some of them have used time-decay functions to

decrease the effect of old ratings, which is not effective approach if users' preferences stay consistent over time for a particular type of item. In this case, old ratings may help to improve the accuracy of prediction.

In this work, we proposed a universal and domain-independent method for adding temporal information to any non-temporal RS. To this end, we predict the future similarity trend between all users/items and then re-ranking their neighborhood sets by giving more (less) priority to users/items with upward (downward) trends. The updated neighborhood sets are used within the target RS to recommend items. As such, the proposed method can be applied to any CF recommendation algorithm that is based on user-user and/or item-item similarity estimation. As the proposed method only make some small modification to the neighborhood selection step of the recommendation process, it does not significantly increase the computational complexity of the original RS algorithm. Our experiments on two benchmark movies and book recommendation datasets showed that this simple trick could significantly improve accuracy of many existing and state-of-the-art CF algorithms. According to the experiment results, the proposed method in some cases could improve accuracy by more than 50%.

REFERENCES

[1] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May 2017.

[2] P. Winoto and T. Y. Tang, "The role of user mood in movie recommendations," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 6086–6092, Aug. 2010, doi: 10.1016/j.eswa.2010.02.117.

[3] V. Subramaniyaswamy, R. Logesh, M. Chandrashekhar, A. Challa, and V. Vijayakumar, "A personalised movie recommendation system based on collaborative filtering," *Int. J. High Perform. Comput. Netw.*, vol. 10, nos. 1–2, pp. 54–63, 2017.

[4] J. J. Castro-Schez, R. Miguel, D. Vallejo, and L. M. López-López, "A highly adaptive recommender system based on fuzzy logic for B2C e-commerce portals," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2441–2454, Mar. 2011.

[5] E. R. Núñez-Valdéz, J. M. C. Lovelle, O. S. Martínez, V. García-Díaz, P. O. de Pablos, and C. E. M. Marín, "Implicit feedback techniques on recommender systems applied to electronic books," *Comput. Hum. Behav.*, vol. 28, no. 4, pp. 1186–1193, Jul. 2012.

[6] C. Porcel, A. Tejeda-Lorente, M. A. Martínez, and E. Herrera-Viedma, "A hybrid recommender system for the selective dissemination of research resources in a technology transfer office," *Inf. Sci.*, vol. 184, no. 1, pp. 1–19, Feb. 2012.

[7] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, and X. He, "Using rich social media information for music recommendation via hypergraph model," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 7, no. 1, p. 22, Oct. 2011.

[8] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf. Sci.*, vol. 180, no. 22, pp. 4290–4311, Nov. 2010.

[9] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López, "Which app? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9367–9375, Aug. 2012.

[10] J. Bobadilla, F. Serradilla, and A. Hernando, "Collaborative filtering adapted to recommender systems of e-learning," *Knowl.-Based Syst.*, vol. 22, no. 4, pp. 261–265, May 2009.

[11] S. Ahmadian, N. Joorabloo, M. Jalili, Y. Ren, M. Meghdadi, and M. Afsharchi, "A social recommender system based on reliable implicit relationships," *Knowl.-Based Syst.*, vol. 192, Mar. 2020, Art. no. 105371.

[12] K. McNally, M. P. O'Mahony, M. Coyle, P. Briggs, and B. Smyth, "A case study of collaboration and reputation in social Web search," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 1, pp. 1–29, Oct. 2011.

[13] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating collaborative filtering recommender algorithms: A survey," *IEEE Access*, vol. 6, pp. 74003–74024, 2018.

[14] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Syst. Appl.*, vol. 69, pp. 29–39, Mar. 2017.

[15] G. Adomavicius, A. Tuzhilin, and R. Zheng, "REQUEST: A query language for customizing recommendations," *Inf. Syst. Res.*, vol. 22, no. 1, pp. 99–117, Mar. 2011.

[16] L. Baltrunas and F. Ricci, "Experimental evaluation of context-dependent collaborative filtering using item splitting," *User Model. User-Adapted Interact.*, vol. 24, nos. 1–2, pp. 7–34, Feb. 2014.

[17] Y. Ren, M. Tomko, F. D. Salim, J. Chan, C. L. A. Clarke, and M. Sanderson, "A location-query-browse graph for contextual recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 204–218, Feb. 2018.

[18] N. Joorabloo, M. Jalili, and Y. Ren, "A probabilistic graph-based method to improve recommender system accuracy," in *Proc. Int. Conf. Eng. Appl. Neural Netw.* Cham, Switzerland: Springer, 2019, pp. 151–163.

[19] S. Ahmadian, N. Joorabloo, M. Jalili, M. Meghdadi, M. Afsharchi, and Y. Ren, "A temporal clustering approach for social recommender systems," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2018, pp. 1139–1144.

[20] N. N. Liu, M. Zhao, E. Xiang, and Q. Yang, "Online evolutionary collaborative filtering," in *Proc. 4th ACM Conf. Recommender Syst. (RecSys)*, 2010, pp. 95–102.

[21] R. Rafeh and A. Bahrehmand, "An adaptive approach to dealing with unstable behaviour of users in collaborative filtering systems," *J. Inf. Sci.*, vol. 38, no. 3, pp. 205–221, Jun. 2012.

[22] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! Music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, 2011, pp. 165–172.

[23] L. Cao, "Non-IID recommender systems: A review and framework of recommendation paradigm shifting," *Engineering*, vol. 2, no. 2, pp. 212–224, Jun. 2016.

[24] Y. Ding, X. Li, and M. E. Orlowska, "Recency-based collaborative filtering," in *Proc. 17th Australas. Database Conf.*, vol. 49. Darlinghurst, NSW, Australia: Australian Computer Society, 2006, pp. 99–107.

[25] C. Xing, F. Gao, S. Zhan, and L. Zhou, "Collaborative filtering recommendation algorithm incorporated with user interest change," *Jisuanji Yanjiu yu Fazhan, Comput. Res. Develop.*, vol. 44, no. 2, pp. 296–301, 2007.

[26] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proc. 14th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2005, pp. 485–492.

[27] Y. Zhang and Y. Liu, "A collaborative filtering algorithm based on time period partition," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Informat.*, Apr. 2010, pp. 777–780.

[28] J. Suchal and P. Návrat, "Full text search engine as scalable k-nearest neighbor recommendation system," in *Proc. IFIP Int. Conf. Artif. Intell. Theory Pract.* Berlin, Germany: Springer, 2010, pp. 165–173.

[29] A. Bellogín, P. Castells, and I. Cantador, "Neighbor selection and weighting in user-based collaborative filtering: A performance prediction approach," *ACM Trans. Web*, vol. 8, no. 2, pp. 1–30, Mar. 2014.

[30] T. Q. Lee, Y. Park, and Y.-T. Park, "An empirical study on effectiveness of temporal information as implicit ratings," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 1315–1321, Mar. 2009.

[31] A. Zimdars, D. M. Chickering, and C. Meek, "Using temporal data for making recommendations," in *Proc. 17th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 580–588.

[32] F. Ricci and Q. N. Nguyen, "Acquiring and revising preferences in a critique-based mobile recommender system," *IEEE Intell. Syst.*, vol. 22, no. 3, pp. 22–29, May 2007.

[33] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 447–456.

[34] T. Y. Tang, P. Winoto, and K. C. C. Chan, "Scaling down candidate sets based on the temporal feature of items for improved hybrid recommendations," in *Proc. IJCAI Workshop Intell. Techn. Web Personalization*. Berlin, Germany: Springer, 2003, pp. 169–186.

[35] T. Lee, Y. Park, and Y. Park, "A time-based approach to effective recommender systems using implicit feedback," *Expert Syst. Appl.*, vol. 34, no. 4, pp. 3055–3062, May 2008.

[36] B. Karahodza, H. Supic, and D. Donko, "An approach to design of time-aware recommender system based on changes in group user's preferences," in *Proc. 10th Int. Symp. Telecommun. (BIHTEL)*, Oct. 2014, pp. 1–4.
[37] C. Xia, X. Jiang, S. Liu, Z. Luo, and Z. Yu, "Dynamic item-based recommendation algorithm with time decay," in *Proc. 6th Int. Conf. Natural Comput.*, vol. 1, Aug. 2010, pp. 242–247.
[38] N. Lathia, S. Hailes, and L. Capra, "Temporal collaborative filtering with adaptive neighbourhoods," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2009, pp. 796–797.
[39] C. Chen, H. Yin, J. Yao, and B. Cui, "TeRec: A temporal recommender system over tweet stream," *Proc. VLDB Endowment*, vol. 6, no. 12, pp. 1254–1257, Aug. 2013.
[40] X. Zheng, Y. Luo, L. Sun, J. Zhang, and F. Chen, "A tourism destination recommender system using users' sentiment and temporal dynamics," *J. Intell. Inf. Syst.*, vol. 51, no. 3, pp. 557–578, Dec. 2018.
[41] R.-H. Hwang, Y.-L. Hsueh, and Y.-T. Chen, "An effective taxi recommender system based on a spatio-temporal factor analysis model," *Inf. Sci.*, vol. 314, pp. 28–40, Sep. 2015.
[42] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st ed. Cham, Switzerland: Springer, 2016.
[43] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *Int. J. Forecasting*, vol. 22, no. 3, pp. 443–473, Jan. 2006.
[44] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. Melbourne, VIC, Australia: OTexts, 2018.
[45] W. Huiwen and M. Jie, "Predictive modeling on multivariate linear regression," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 33, no. 4, p. 500, 2007.
[46] P. G. Campos, F. Díez, and I. Cantador, "Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols," *User Model. User-Adapted Interact.*, vol. 24, nos. 1–2, pp. 67–119, Feb. 2014.
[47] P. Sánchez and A. Bellogín, "Time and sequence awareness in similarity metrics for recommendation," *Inf. Process. Manage.*, vol. 57, no. 3, May 2020, Art. no. 102228.
[48] F. Rezaeimehr, P. Moradi, S. Ahmadian, N. N. Qader, and M. Jalili, "TCARS: Time- and community-aware recommendation system," *Future Gener. Comput. Syst.*, vol. 78, pp. 419–429, Jan. 2018.
[49] S. M. Daneshmand, A. Javari, S. E. Abtahi, and M. Jalili, "A time-aware recommender system based on dependency network of items," *Comput. J.*, vol. 58, no. 9, pp. 1955–1966, Sep. 2015.
[50] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, p. 3, Jul. 2014.
[51] M. Kendall and J. Gibbons, *Rank Correlation Methods* (A Charles Griffin Book), E. Arnold, Ed., 5th ed. London, U.K.: Oxford Univ. Press, 1990.
[52] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bull. Soc. Vaudoise Sci. Nat.*, vol. 37, pp. 547–579, Jan. 1901.
[53] H.-N. Kim, A. El-Saddik, and G.-S. Jo, "Collaborative error-reflected models for cold-start recommender systems," *Decis. Support Syst.*, vol. 51, no. 3, pp. 519–531, Jun. 2011.
[54] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*. New York, NY, USA: Association for Computing Machinery, 1999, pp. 230–237.
[55] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Inf. Sci.*, vol. 178, no. 1, pp. 37–51, Jan. 2008.
[56] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative filtering," *Knowl.-Based Syst.*, vol. 56, pp. 156–166, Jan. 2014.
[57] N. Polatidis and C. K. Georgiadis, "A multi-level collaborative filtering method that improves recommendations," *Expert Syst. Appl.*, vol. 48, pp. 100–110, Apr. 2016.
[58] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowl.-Based Syst.*, vol. 23, no. 6, pp. 520–528, Aug. 2010.
[59] M. Jamali and M. Ester, "TrustWalker: A random walk model for combining trust-based and item-based recommendation," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 397–406.
[60] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. WWW*, vol. 1, 2001, pp. 285–295.
[61] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016.
[62] M. Thelwall and K. Kousha, "Goodreads: A social network site for book readers," *J. Assoc. Inf. Sci. Technol.*, vol. 68, no. 4, pp. 972–983, Apr. 2017.
[63] K. Verbert, H. Drachsler, N. Manouselis, M. Wolpers, R. Vuorikari, and E. Duval, "Dataset-driven research for improving recommender systems for learning," in *Proc. 1st Int. Conf. Learn. Analytics Knowl. (LAK)*, 2011, pp. 44–53.
[64] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," 2012, *arXiv:1205.3193*. [Online]. Available: http://arxiv.org/abs/1205.3193
[65] M. Grčar, D. Mladenič, B. Fortuna, and M. Grobelnik, "Data sparsity issues in the collaborative filtering framework," in *Proc. Int. Workshop Knowl. Discovery Web*. Berlin, Germany: Springer, 2005, pp. 58–76.

NIMA JOORABLOO received the B.S. degree in computer hardware engineer from the Azad University of Tehran, Tehran, Iran, in 2007, and the M.S. degree in artificial intelligence from the Science and Research University of Tehran, Tehran, in 2010. He is currently pursuing the Ph.D. degree in electrical and biomedical engineering with RMIT University. His academic research interests include recommender system, social network analysis, heterogeneous network, and sequence mining.

MAHDI JALILI (Senior Member, IEEE) received the Ph.D. degree in computer and communications sciences from the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, in 2008. He is currently a Senior Lecturer with the School of Engineering, RMIT University, Melbourne, VIC, Australia. His current research interests include network science and engineering. He was previously an Australian Research Council DECRA Fellow and a RMIT Vice-Chancellor Research Fellow. He is an Associate Editor of *Complex Adaptive Systems Modeling*, *Complexity*, and *Mathematical Problems in Engineering*.

YONGLI REN received the Ph.D. degree in information technology from Deakin University, Geelong, VIC, Australia. He is currently a Lecturer with the Computer Science and Information Technology, School of Science, RMIT University, Melbourne, VIC, Australia. His research interests include data analytics, user modeling, personalization, and recommender systems. He has won the Alfred Deakin Medal for Doctoral Thesis 2013 at Deakin University, and the Best Paper Award at the IEEE/ACM ASONAM 2012 Conference.