IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2018, accepted November 25, 2018, date of publication November 28, 2018,
date of current version December 27, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2883742

Evaluating Collaborative Filtering Recommender Algorithms: A Survey

MAHDI JALILI[1], (Senior Member, IEEE), SAJAD AHMADIAN[2], MALIHEH IZADI[3], PARHAM MORADI[4], AND MOSTAFA SALEHI[5]

[1]School of Engineering, RMIT University, Melbourne, VIC 3001, Australia
[2]Department of Computer Engineering, University of Zanjan, Zanjan 4537138791, Iran
[3]Department of Computer Engineering, Sharif University of Technology, Tehran 1136511155, Iran
[4]Department of Computer Engineering, University of Kurdistan, Sanandaj 66177-15175, Iran
[5]Faculty of New Sciences, University of Tehran, Tehran 1417466191, Iran

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

This work was supported by the Australian Research Council under Project DP17010230.

ABSTRACT Due to the explosion of available information on the Internet, the need for effective means of accessing and processing them has become vital for everyone. Recommender systems have been developed to help users to find what they may be interested in and business owners to sell their products more efficiently. They have found much attention in both academia and industry. A recommender algorithm takes into account user–item interactions, i.e., rating (or purchase) history of users on items, and their contextual information, if available. It then provides a list of potential items for each target user, such that the user is likely to positively rate (or purchase) them. In this paper, we review evaluation metrics used to assess performance of recommendation algorithms. We also survey a number of classical and modern recommendation algorithms and compare their performance in terms of different evaluation metrics on five benchmark datasets. Our experiments show that there is no golden recommendation algorithm showing the best performance in all evaluation metrics. We also find large variability across the datasets. This indicates that one should carefully consider the evaluation criteria in choosing a recommendation algorithm for a particular application.

INDEX TERMS Recommender systems, collaborative filtering, evaluation metrics, precision, ranking, diversity.

I. INTRODUCTION

Nowadays, there are many commercial and non-commercial websites on the Internet offering increasing volume of diverse products to users. Recommender systems (RSs) have become an inevitable part of the world-wide-web, due to the emergence of e-commerce, wide and fast growing range of choices for customers, diversity of preferences between users, lack of precise knowledge of their needs, and lack of keyword terms to express and use search engines to meet these demands. Recommender algorithms predict the utility of an item to a target user, and suggest the best items regarding the user's preferences using their past ratings to available items in the system. An item may represent a movie, a music track, a book, an application, a restaurant, a place for vacation, a webpage or any other thing used by users. Personalized RSs utilize user-related information to customize the generated recommendations regarding their preferences.

As more RS algorithms are developed, it is necessary to put efforts on developing proper evaluation methods. Evaluating performance of RSs is one of the challenging tasks within these systems [1]. There is no a single metric that can efficiently measure the performance and often one has to use a number of them at the same time. For example, accuracy and diversity of RSs that are not often in the same direction, have to be shown at the same time to have a fair understanding of the performance. Recommendation algorithms are proposed for different objectives, and thus different evaluation metrics are chosen to assess the performance of the algorithms from different aspects. A RS may have good performance based on some evaluation metrics, while having poor performance based on some other. Furthermore, performance of RSs depends on the nature of items considered in these systems. For example, a specific method may have good performance only for movie recommendation systems, while not performing well for other types of items such as music, book or news.

Although there are a number of reviews for RSs [2]–[5], to the best of our knowledge, there is no comprehensive work comparing different RS algorithms on available evaluation metrics and also various types of datasets. In this paper,

we provide a brief overview of a number of memory- and model-based collaborative filtering RSs and assess their performance on five benchmark datasets including Movielens, Epinions, LastFM, BookCrossing, and Jester, which have been frequently used as benchmarks in RSs research. The evaluation metrics are categorized into three classes: accuracy, rank-based, coverage and novelty/diversity metrics. In addition, a combination of several evaluation metrics is considered as a unified metric to compare the performance of recommendation methods. Our experiments on the performance of the algorithms on the datasets show that there is no golden algorithm that outperforms others in all of the evaluation metrics. The best performers are indeed spread across the evaluation metrics, which indicates the significance of the evaluation metric in choosing a recommendation algorithm for a particular application. We also find large variability across the datasets.

II. RECOMMENDER SYSTEMS

The task of RSs is either *prediction* or *recommendation*. Prediction means deciding whether or not a user would like a particular item by predicting *how much* they would rate a new item. On the other hand, recommendation refers to the task of recommending a *list* of items that the user would probably like [6]. Real-world example of items for recommendations include movies [7], [8], music [9], [10], documents and books [11], [12], news [13], [14], jokes [15], e-learning material [16], [17] and e-commerce applications [18], [19]. RSs utilize various explicit or implicit information sources to generate their recommendations, including ratings, social information, demographic and contextual information such as time, location, contents, features and tags [20]–[24]. Regarding to how recommendations are made, RSs algorithms are categorized into three groups: *content-based*, *collaborative filtering* and *hybrid* recommendation algorithms [3], [25], [26].

A. CONTENT-BASED RECOMMENDATION ALGORITHMS

The items recommended using these algorithms have similar contents or features to those positively rated by the target user [27]–[30]. For example, in recommending a movie, the features may include genre, actors and director. When a user indicates their preference in action movies, a content-based algorithm recommends the best-matching action movies to them. Moreover, content-based recommenders can exploit the information in users' profiles [31] such as demographic information, age, gender, nationality, education or occupation [32], [33] to improve the quality of recommendation.

B. COLLABORATIVE FILTERING RECOMMENDATION ALGORITHMS

This approach recommends the items based on the similarity between the users and/or the items [4], [26]. The term ''collaborative'' (first used by the creators of Tapestry recommender [34]) refers to the need of collaboration of different users for ''filtering'' abundant data and generating recommendations. For example, if users u and p have similarly rated items in the past, their history of preferences will impact the recommendations each one gets, more than preferences of other non-similar users to them. The algorithms in this category are divided into two subgroups of memory-based and model-based Collaborative Filtering (CF) recommenders [35], [36]. This approach is the most widely used RSs algorithm in many industrial applications [3], [37].

C. HYBRID RECOMMENDATION ALGORITHMS

Different combinations of content-based and CF algorithms exist that exploit users' and items' information as well as the ratings and similarity of different users and items [38]–[41]. Generally, there are four different ways to combine the content-based and CF recommenders [25], including (1) separately implement content-based and CF algorithms, then combine their results [33], [42], (2) use some of content-based features to boost a CF algorithm [33], [43], [44], (3) or alternatively, use some of the characteristics of a CF algorithm to boost a content-based recommender [43], (4) and finally, unify content-based and CF characteristics into one recommender [33], [45]. Several studies have shown that hybrid RS algorithms provide better recommendations than separate content-based or CF algorithms [33], [43], [46].

In spite of the growing research in the field of RSs over the past years, there are still several challenges for both content-based and CF methods. Generally, comparing content-based and CF approaches, the latter is not limited to the content or features since they use the users' rating history for prediction and they can recommend diverse items. However *new user problem* is still an issue, likewise *new item problem* and *sparsity* [25], [33], [43], [47]. The first two problems are usually referred to as ''cold-start problem'' [43], [45]. Combining content-based and CF algorithms – a hybrid approach – has been used as a solution for the cold start problem in different studies. Other studies have used various methods to find the best item for recommending to a new user and learn their preferences gradually; e.g., a simple way is to recommend popular items or items based on users' demographic information such as age, gender, location, nationality, education or occupation (assuming new users have to fill out a profile before using the system) [25], [48]. The quality of RSs can be enhanced by various approaches; for example, by better knowing users and items [43], [49]–[51] through incorporating features of their profiles such as demographic information or keywords [31], [33], [52], [53] or using data mining techniques [31], [33], [52], [54], utilizing additional contextual information such as time and location [5], [55]–[57], getting feedbacks on recommendations and improving the performance according to them [13], [58], [59], using multi-criteria ratings, or by adding the features of complex networks into recommendation process [60].

III. COLLABORATIVE FILTERING RECOMMENDATION ALGORITHMS

Collaborative Filtering (CF) RSs consider item- or user-based similarities and extract the list of recommendations based

on them. To formalize this, let's denote "U" as the set of users in the system, displayed by $U = \{u_1, u_2, \ldots, u_m\}$ and "I" as the set of items, represented by $I = \{I_1, I_2, \ldots, I_n\}$. r_{ui} represents the rating given by user u to item i, and usually has a numerical scale, such as the five-star rating scale for Movielens[1] (1 means very bad and 5 very good) and the ten-star rating scale for IMDB.[2] In some cases, the scale is continuous, e.g., r_{ui} is in the range $[-10, 10]$ for Jester Joke recommender. There are explicit or implicit ways for gathering the users' ratings, as indicator of their preference toward the items [61]–[63]. The numerical ratings along binary (like/dislike) and unary ones (e.g., the only choice of *Like* in Facebook), which are entered directly by a user, are considered as explicit ratings [4]. Implicit feedback gathered from the users can be used to improve the quality of recommendations [4], [64]. The data file containing ratings (called user-item interaction matrix) is used for learning the users' preferences and habits, predicting new ratings, recommending items to users, and finally evaluating the system. For evaluation purposes, ones needs to divide the rating matrix into two parts: training set for learning and test set for evaluating the performance. Memory-based and model-based CF algorithms use these matrices differently, which are discussed in the following sections.

A. MEMORY-BASED COLLABORATIVE FILTERING

As mentioned, memory-based algorithms predict new ratings based on the available data (which is loaded into the memory), using similarity of other users or items to the target user/item [4], [25]. The set of similar users to a target user (or similar items to a target item) is called their neighbor set and is used to extract users/items with similar history of ratings. The underlying assumption is that if two users have similar history of ratings for common items, they will likely have similar preferences for the rest of items. As for the two items rated similarly by several users, they would probably be rated in the same manner by the rest of users. However, there are always individuals with unique tastes and preferences which would not help this case, but generally speaking, this assumption has proved to be useful. After forming the neighborhood, a new rating for a target user-item pair is predicted as a function of the neighbors' ratings for that particular item and the degree of their similarities to the target user. Based on using the target users/items neighbors, these algorithms are divided into two categories of user-based and item-based CF.

There are various similarity measures for extracting the set of neighbors, such as cosine–based similarity adapted from information retrieval [65], adjusted cosine, Pearson correlation coefficient, constrained Pearson correlation, Euclidean and mean squared differences [3]. In addition to the similarity, in some cases, one can use dissimilarity values as well [66], e.g., when the sparsity of available data is high and the relevance becomes more important than the correlation [67]–[69]. Next, we review cosine-based similarity and Pearson correlation; however as Pearson correlation coefficient has shown to be the more effective technique [25], we chose Pearson as our similarity measure in the implementations. Cosine-based similarity between two users u and u' is calculated using Eq. (1) as follows [6], [64]:

$$similarity(\mathrm{u}, \mathrm{u}') = \cos(\mathrm{u}, \mathrm{u}') = \frac{\sum\limits_{i \in I_{uu'}} r_{ui} \times r_{u'i}}{\sqrt{\sum\limits_{i \in I_{uu'}} r_{ui}^2} \sqrt{\sum\limits_{i \in I_{uu'}} r_{u'i}^2}} \quad (1)$$

where $I_{uu'}$ is the set of items rated by both users, and r_{ui} is the rating that user u has given to item i. Pearson correlation coefficient for users u and u' is obtained using Eq. (2) [70], [71]:

$$\begin{aligned} similarity(\mathrm{u}, \mathrm{u}') &= PC_{uu'} \\ &= \frac{\sum\limits_{i \in I_{uu'}} (r_{ui} - \overline{r}_u) \times (r_{u'i} - \overline{r}_{u'})}{\sqrt{\sum\limits_{i \in I_{uu'}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum\limits_{i \in I_{uu'}} (r_{u'i} - \overline{r}_{u'})^2}} \end{aligned} \quad (2)$$

where $\overline{r_u}$ is the average rating of user u.

As the similarity values are computed, one can use k *Nearest Neighbor*(KNN) algorithm to find the k-most similar users to the target user. Eqs. (1) and (2) are also applicable for calculating the similarities between items [6], [72]. Usually, in real recommenders the similarities are pre-calculated and used whenever there is a need to generate recommendations. This enables to make the recommendation on a real-time fashion. However, the similarity values must be recalculated once in a while, due to the change in the users and items networks in the system [25].

1) USER-BASED COLLABORATIVE FILTERING

User-based CF is among the most successful and widely implemented techniques in RSs [6], [25]. It recommends items to a target user based on opinions of other similar users to them [70]. After forming the neighborhood, the new rating for the target user-item pair is estimated considering the weights of different neighbors. That is, the higher is the similarity of a user with the target user, the more impact their rating has on the estimation of the target user's rating. The new rating for user u and item i is predicted as $\tilde{r_{ui}}$ using:

$$\tilde{r}_{ui} = \frac{\sum\limits_{u' \in NS_u} (r_{u'i}) \times (\mathrm{similarity}(\mathrm{u}, \mathrm{u}'))}{\sum\limits_{u' \in NS_u} |(\mathrm{similarity}(\mathrm{u}, \mathrm{u}'))|} \quad (3)$$

where NS_u is the neighbor set of target user u with k members. However, Eq. (3) neglects the fact that different people may differently use the rating scale. Eq. (4) considers this issue through adjusting the formula by first subtracting the average rating of each neighbor user, $\overline{r_{u'}}$, then multiplying the result

[1] www.movielens.com
[2] www.imdb.com

by his/her similarity with the target user, and finally adding the target user's average rating [4], [25].

$$\tilde{r}_{ui} = \bar{r}_u + \frac{\sum_{u' \in NS_u} (r_{u'i} - \bar{r}_{u'}) \times (\text{similarity}(u, u'))}{\sum_{u' \in NS_u} |(\text{similarity}(u, u'))|} \quad (4)$$

User-based CF is a popular method for being relatively fast with reasonable accuracy in the prediction task. However, it has its own drawbacks, such as dealing with sparsity and scalability issues. As the number of items grows, even an active user does not visit a high percentage of total items, and thus the data will be extremely sparse. A possible solution for the sparsity problem is to use semi-intelligent content-boosted agents to increase the density of rating matrix [73], [74]. Another solution is to use latent semantic indexing to detect the similarity between users and items in a reduced dimensional space [75], [76]. As for the scalability, the computation complexity increases as the number of users and items in the system grow [6]. In the worst case, for a user-based CF with m users and n items, the phase of calculating similarities has a time complexity in order of $O(m^2n)$, and the prediction phase takes the order of $O(km)$. The user-based CF provides low coverage of available items in the system, and the final recommendations are more biased to popular items.

2) ITEM-BASED COLLABORATIVE FILTERING

In order to alleviate the scalability issue of user-based CF, item-based CF algorithm was proposed [6]. The underlying assumption here is that two items which are being rated similarly by several users, would probably be rated in the same manner by the rest of users. Unlike the user-based CF, the item-based algorithm analyzes the rating matrix to identify the relationships between *items*. Prediction is computed by taking a weighted average of the target user's ratings on similar items. Item-based CF is preferred to user-based whenever the number of items in the system is far less than the number of users. Fig. 1, retrievd from Sarwar *et al.* [6], depicts the similarity computation of item-based CF.

When estimating an unknown rating value, first Eq. (3) is modified to calculate the Pearson correlation between items rather than users. Then, the new ratings are calculated using Eq. (5):

$$\tilde{r}_{ui} = \frac{\sum_{i' \in NS_i} (r_{ui'}) \times (\text{similarity}(i, i'))}{\sum_{i' \in NS_i} |(\text{similarity}(i, i'))|} \quad (5)$$

where NS_i is the neighborhood set of item i. According to [6] and [72], item-based CF is scalable for larger data sets since the item-neighborhood is rather static, and therefore, the similarities can be efficiently pre-computed. Thus, the online (prediction phase) performance is better than user-based CF. Regarding the computational complexity of item-based CF with m users and n items, the similarity calculation phase needs a time complexity in order of $O(m^2n)$ and prediction phase takes $O(km)$. However, in a real system with large number of users and items, the actual complexity is far less, since the number of items not being rated by a user is much less than the total number of items.

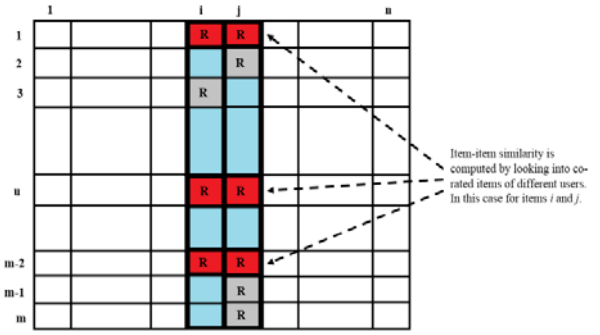

FIGURE 1. Calculating similarity values between items for item-based CF method. The purpose is to calculate similarity value between items *i* and *j* based on their common ratings provided by users. In this matrix, the rows represent the users and the columns represent the items. The common ratings for items *i* and *j* are provided by users 1, *u*, and *m*-2. The figure and concept is adopted from [6].

3) RESOURCE ALLOCATION COLLABORATIVE FILTERING

Resource allocation CF uses the notion of link prediction [77] in RSs to improve its performance. There are a number of methods for link prediction in complex networks [78]. Javari *et. al.* [79] used resource allocation (RA) index [80], which has shown good estimation of the missing links in bipartite networks of users and items. They used the concept that more popular items should have less impact when calculating the similarities of users, since most users like popular items, and therefore, they are less worthy to be used for the recommendation purposes. RA is a degree-based local similarity measure, in which its value for two users depends on the degree of the common-rated items by users; that is the more popular the common-rated items, the less the value of their RA similarity index. Moreover, as the number of common-rated items grows, RA increases, and the calculated similarity is more trust-worthy. According to this algorithm, if each node distributes its resources equally to all of its neighbors, the RA index, as a degree indicator of the similarity between users u and u', is obtained using Eq. (6) as follows:

$$S_{RA}(u, u') = \sum_{i \in \Gamma(u) \cap \Gamma(u')} \frac{1}{d_i} \quad (6)$$

where d_i is degree of item i, that is the number of users who have rated i, and $\Gamma(u)$ is the set of items rated by u. Eq. (6) indicates that the RA values depend on both the number of common neighbors and their degree. As the RA scores are obtained, the similarity between the users is obtained as follows:

$$similarity_{CF-RA}(u, u') = PC_{uu'} \times S_{RA}(u, u') \quad (7)$$

The rest is the same as standard user-based CF algorithm.

4) USER OPINION SPREADING

User Opinion Spreading (UOS) method uses a combination of CF algorithm with the users' opinion spreading process to predict more accurate ratings [81]. In this method, the similarity value between two users is defined by measuring the amount of opinion that the users transfer to each other based on the primitive user-item rating matrix. To this end, Eq. (8) is used to normalize the ratings by transforming the ratings to the range $[-1, 1]$.

$$e_{i\alpha} = \frac{2 * \left(r_{i\alpha} - r_i^{\min}\right)}{r_i^{\max} - r_i^{\min}} - 1 \tag{8}$$

where $r_{i\alpha}$ is the rating of item α given by user i, r_i^{min} and r_i^{max} are the minimum and maximum ratings that have been given by that user, respectively.

Also, the following equation computes the amount of opinion that user i transmits to user j through an item α mutually rated by them:

$$E_{ij\alpha} = e_{i\alpha} * e_{j\alpha} \tag{9}$$

where $e_{j\alpha}$ is computed using Eq. (8). Then, the opinion-based similarity weights between users are computed using Eq. (10) as follows:

$$S_{ij} = \frac{1}{k_i} \sum_{\alpha=1}^{P} \frac{A_{i\alpha} A_{j\alpha} E_{ij\alpha}}{k_\alpha} \tag{10}$$

where $A_{i\alpha} = 1$ if user i has rated item α, otherwise $A_{i\alpha} = 0$. P is the number of all items and k_i and k_α denote the number of neighbors of user i and item α, respectively. Finally, the rating of user i on item α is computed as follows:

$$r_{i\alpha} = \overline{r}_i + \frac{\sum_{j=1}^{n} \left[\left(r_{j\alpha} - \overline{r}_j\right) * S_{ji}\right]}{\sum_{j=1}^{n} S_{ji}} \tag{11}$$

where n denotes the number of neighbors, $r_{j\alpha}$ is the rating of user j on item α, $\overline{r}_i$ is the average ratings given by user i, and S_{ji} is the opinion-based similarity value between users i and j that is computed using Eq. (10).

5) MULTI-LEVEL COLLABORATIVE FILTERING

A Multi-Level Collaborative Filtering (MLCF) is proposed in [82] to improve the accuracy of the classical CF. The main idea of this method is to use an improved similarity measure to enhance the ability of CF in accurately predicting the unseen items This similarity measure is computed using Eq. (12), as:

$$sim\,(a, b) = \begin{cases} sim_{PCC(a,b)} + x, & if\ \dfrac{|I_a \cap I_b|}{T} \geq t_1 \\ \quad and\ sim_{PCC(a,b)} \geq y & \\ sim_{PCC(a,b)} + x, & if\ \dfrac{|I_a \cap I_b|}{T} < t_1 \\ \quad and\ \dfrac{|I_a \cap I_b|}{T} \geq t_2\ and\ sim_{PCC(a,b)} \geq y & \\ sim_{PCC(a,b)} + x, & if\ \dfrac{|I_a \cap I_b|}{T} < t_2 \\ \quad and\ \dfrac{|I_a \cap I_b|}{T} \geq t_3\ and\ sim_{PCC(a,b)} \geq y & \\ sim_{PCC(a,b)} + x, & if\ \dfrac{|I_a \cap I_b|}{T} < t_3 \\ \quad and\ \dfrac{|I_a \cap I_b|}{T} \geq t_4\ and\ sim_{PCC(a,b)} \geq y & \\ 0, & otherwise \end{cases} \tag{12}$$

where I_a is a set of items that are rated by user a, T is the total number of co-rated items, x and y denote positive real numbers, and $sim_{PCC(a,b)}$ is the similarity value between users a and b, which is calculated using Eq. (2). t_1, t_2, t_3, and t_4 are natural numbers that constraint the number of co-rated items for each level. The rest is the same as the classical CF algorithm.

6) ITEM GLOBAL PROFILE EXPANSION

Item Global Profile Expansion (IGPE) is a recommendation method which is based on expanding the users' rating profiles by using the items that are similar to those that have been already rated by the user [83]. Therefore, this method needs to use profile-level information to find similar items. To this end, the cosine vector similarity is used to measure the similarity values between the items as Eq. (13):

$$\varsigma\,(i, j) = \frac{\sum_{u \in U} V_{ui} V_{uj}}{\sqrt{\sum_{u \in U} V_{ui}^2 \sum_{u \in U} V_{uj}^2}} \tag{13}$$

where $\varsigma\,(i, j)$ is the cosine similarity value between items i and j, U is the set of all users in the system, and V_{ui} is the rating of item i given by user u. After calculating the similarity values, the top-k similar items are selected to expand the users' rating profile. To this end, each selected item is assigned by a rating value that is computed as the average weighted rating of the other similar items. Finally, the unseen items are predicted using the classical CF algorithm.

7) USER LOCAL PROFILE EXPANSION

User Local Profile Expansion (ULPE) is a recommendation method to expand users' ratings based on the items that have been rated by the neighbors of the active user [83]. In other words, in this method only those of items are considered in the computations that have already been rated by the neighbors of the active user. Thus, the profile is expanded with the top-l rated items. If a given item was rated by more than one neighbor, the rating is averaged according to their similarity

values with the active user and imputed to the user-item ratings matrix. To obtain the similarity values between the active user and his/her neighbors, the cosine similarity measure is used as Eq. (1). Finally, the similarity values between the users are computed based on the expanded rating profiles, and the remaining unseen items are predicted using the pure CF algorithm.

8) MORE ON MEMORY-BASED COLLABORATIVE FILTERING ALGORITHMS

Due to the simplicity of memory-based algorithms, they have been implemented in many industrial recommenders. The KNN technique used in these algorithms is simple and produces reasonably accurate results, and the new data can be added incrementally to the algorithm. Memory-based CF algorithms do not have a learning step (no model is extracted from the data), but their prediction phase may need heavy computations. Despite their wide-spread use in recent years, memory-based CF algorithms have several drawbacks related to synonymy, sparsity and scalability. Indeed, the similarity-based methods cannot be pre-computed for real-time performance [84]. According to [64], in order to improve the neighborhood-based results, one can utilize several extensions such as default votes, inverse user frequency or case amplification. The first extension assumes a default vote for the pairs of user-item, for which there is no explicit rating. This is performed in order to increase the common items rated by two users (or common users who have rated two items). The idea behind using inverse user frequency is to decrease the effect of universally-liked items when calculating the similarities. The case amplification technique uses a weight transform which would put more emphasis on higher weights, while punishing low weights. Furthermore, as the number of users and items increases, the time complexity of the nearest neighbor algorithms linearly grows [85]. A solution can be to use clustering techniques on users or items, in order to reduce the search time [86]–[88]. Another approach is to use model-based techniques such as matrix factorization and dimensionality reduction to deal with these limitations. Several Model-based algorithms are reviewed in the following section.

B. MODEL-BASED COLLABORATIVE FILTERING

Model-based CF algorithms use different techniques on the training set, in order to find patterns in the data and learn a model for predicting new ratings [36], [89], [90]. One can name Slope one [91], latent factor models such as Matrix Factorization (MF) and Singular Value Decomposition (SVD) [36], [92]–[94], Bayesian classifiers [95], [96], clustering models [97]–[100], various probabilistic relational models [101], [102], probabilistic latent semantic analysis [89], [103], [104], linear regression [6], [105], maximum entropy model [106], Latent Dirichlet Allocation (LDA) [107], Markov chain based models [108], principal component analysis (PCA) [109], probabilistic factor analysis [110], neural networks and fuzzy systems [111], [112] among the techniques used for model-based CF. In the following, we review a number of frequently used model-based methods. Although there is a rich literature on model-based CF, these algorithms have limited applicability in real scenarios.

1) SLOPE ONE

Lemire and Maclachlan proposed three slope one algorithms that pre-compute the average difference between the ratings of two items for users who have rated both items, in the form of $f(x) = x + b$ predictors, where b is a constant and variable x represents the ratings [91]. For each pair of items, slope one calculates how much an item is more preferred than the other one (*popularity differential*), which is then used to predict the user's rating of one of those items, given their rating of the other. Note that the distinction between slope one algorithms is due to the way that the relevant differences are selected for the prediction. For example, consider two users u and u', and two items i and j, as shown in Fig.2. The ratings of user u on items i and j are 1 and 1.5, respectively. User u' gave a rating of 2 to item i and did not rate item j. slope one tries to predict the rating that user u' would give to item j, using both ratings of user u and the rating of a common item between users u and u' (i.e., item i). Since u rated j 0.5 point $(1.5 - 1)$ higher than i, one can predict u' will give j a rating of 2.5 $(2 + 0.5)$. As described by this example, slope one utilizes the information of other users who have rated the same item, other items rated by the same user and the data points that fall neither in the user vector nor in the item vector (e.g., rating of user u to item i) when predicting an unknown rating.

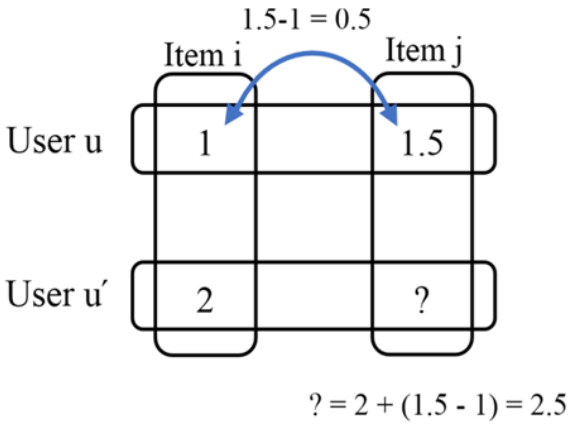

FIGURE 2. Basis of slope one algorithm; adopted from [91]. The purpose is to predict rating of item j for user u' based on the slope 1 algorithm. User u has provided ratings for items i and j. However, user u' has just provided a rating for item i. Therefore, the rating of item j for user u' can be predicted based on the difference of ratings provided for items i and j by user u.

Given two rating vectors v_i and w_i, $i = \{1, 2, \ldots, n\}$, slope one searches for the best predictor of the form $f(x) = x + b$ to predict w from v by minimizing $\sum_i (v_i + b - w_i)^2$. Therefore, b must be chosen to be the average difference between the two vectors. The average deviation of item i with respect to item j

is calculated as:

$$dev_{ji} = \sum_{u \in s_{ji}(\text{trainSet})} \frac{r_{uj} - r_{ui}}{card(s_{ji}(\text{trainSet}))} \tag{14}$$

where $u \in S_{ji}$ ($trainingSet$) is the set of users who have rated both items i and j in the training set, r_{ui} and r_{uj} are the ratings given by user u to items i and j, respectively, and $card(S)$ is the number of elements in set S. Note that the symmetric matrix defined by dev_{ji} can be computed once and updated quickly when new data is entered. Afterward, unseen ratings are predicted as follows:

$$\tilde{r}_{uj} = \frac{1}{card(R_j)} \sum_{i \in R_j} (dev_{ji} + r_{ui}) \tag{15}$$

where $R_j = \left\{ i \,|\, i \in S\,(u)\,, i \neq j, card(S_{ji}\,(trainSet)) > 0) \right\}$ is the set of all relevant items and $S(u)$ is the set of items rated by user u.

2) WEIGHTED SLOPE ONE

One may notice that in the computations of slope one, the number of observed ratings is not included. Considering the scenario shown in Fig. 3 where u has rated both j and i, we want to predict the utility of item q for user u. If 100 users had rated both q and j, and only 10 users had rated both q and i, the rating given by u to j is probably a better predictor for q (i.e., more reliable) than the rating to i.

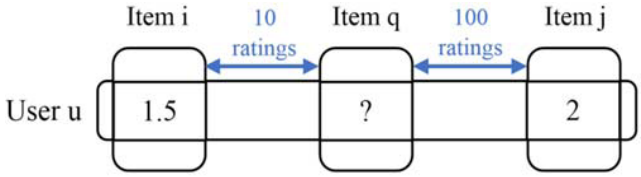

FIGURE 3. Basis of weighted slope 1 algorithm. User *u* has provided ratings for items *i* and *j*. The purpose is to predict ratings of item q for user *u*. The main idea of weighted slope one algorithm is to consider the number of users who have ratings for both of items *i* and *q* (i.e. 10 users) and both of items *j* and *q* (i.e. 100 users). Therefore, the rating given to item *j* by user *u* is probably better and more reliable predictor for item *q*.

The weighted ratings are predicted using Eq. (16) as follows [91]:

$$r^{w}_{uj} = \frac{\sum_{i \in s(\text{u}) - \{\text{j}\}} (dev_{ji} + r_{ui}) \times (\text{card}(s_{ji}(\text{trainSet})))}{\sum_{i \in s(\text{u}) - \{\text{j}\}} (\text{card}(s_{ji}(\text{trainSet})))} \tag{16}$$

3) MATRIX FACTORIZATION METHODS

Due to the limitations of memory-based CF algorithms regarding scalability and sparsity issues, new algorithms tried to address these issues using dimensionality reduction techniques [36], [75]. Matrix Factorization (MF) methods such as Singular Value Decomposition (SVD) are among these techniques [36], [92]. In their simplest form, these algorithms factorize the rating matrix into two low-rank matrices: users profile and items profile. High similarity between item and user profiles results in a recommendation. According to [113], these methods have several advantages such as better accuracy (in regard to kNN-based algorithms), good scalability, and relatively easy learning process. However, their main burden is the difficulty in learning the model. In the following, we give details on a number of methods based on matrix factorization.

a: REGULARIZED SINGULAR VALUE DECOMPOSITION

In recommendation algorithms based on SVD, the SVD technique is used to first detect the latent relationships between the users and the items and then to generate a low-dimensional representation of the original rating matrix space to calculate the neighborhood in the reduced space [75]. SVD factorizes the rating matrix into a product of two low-rank matrices. It produces a set of uncorrelated eigenvectors which represents the users and the items. For a rating matrix M with $m \times n$ dimensions and rank r, SVD is calculated as:

$$SVD(\text{M}) = U \times S \times V^T \tag{17}$$

where m and n are the total number of users and items respectively, and dimensions of U, S and V are $m \times m$, $m \times r$ and $r \times n$, respectively. U and V are two orthogonal matrices, S is a diagonal matrix which is called the "singular matrix" and have r nonzero diagonal elements. Note that the dimensions of matrices U and V are reduced to $m \times r$ and $r \times n$, respectively and the values on the diameter of S are sorted decreasingly. The first r columns of U and V represent the orthogonal eigenvectors associated with the r nonzero eigenvalues of MM^T and M^TM, respectively. U and V are called the *left* and *right* singular vectors, respectively [75], [114]. One can keep only k of r singular values (highest values) and discard lower entries. $(r - k)$ columns from U and $(r - k)$ rows from V^T are eliminated to produce U_k and V_k^T matrices. U_k and V_k are multiplied together using S_k to produce M_k. The reconstructed matrix M_k is the closest rank-k matrix to M, with respect to the Frobenius norm of matrix. Indeed,

$$M_k = U_k \times S_k \times V_k^T \tag{18}$$

Now, the rating prediction for user u and item i, $\tilde{r_{ui}}$, is calculated using dot product as:

$$\tilde{r}_{ui} = \bar{r}_u + U_k \sqrt{S_k^T(\text{u})} \times \sqrt{S_k} \times V_k^T(\text{i}) \tag{19}$$

SVD has several advantages and often results in predictions with good accuracy. SVD addresses the synonymy problem by helping users who have rated similar, but not exact items to be mapped into the space spanned by the same eigenvectors. Furthermore, the low-rank approximation of the original space is better than the original space itself due to eliminating small singular values which cause noise in the user-item relationship [115]–[117]. According to [75], [76], and [117], algorithms based on SVD can make the neighborhood formation process of CF systems very scalable, often resulting in better performance. The space storage of SVD takes $O\,(m + n)$, since it only stores two matrices

of size $m \times k$ and $k \times n$. Therefore, the space storage of SVD is very efficient compared to neighborhood-based CF algorithms that is $O(m^2)$. Despite its popularity and being implemented in several studies [118], [119], SVD is slow and requires dealing with the missing values. Furthermore, its optimization method is non-convex. A solution to deal with the missing values is to replace them with the users' average ratings or those of the items [75]. A typical model-based recommendation algorithm has two steps: one for model-building (offline) and one for execution (online). A major defect of SVD is its time-consuming offline decomposition step. For an $m \times n$ rating matrix, the required time of this step is the order of $O((m+n)^3)$ [115], [116], while the complexity of online phase in neighborhood-based CF is $O(m^2 n)$, much lower than SVD [75]. Incremental SVD was proposed to help improve the computation time [117], which first computes a model with an appropriate size, and then use a projection method to build incrementally upon that [117].

During the Netflix competition in December 2006, regularized SVD (RVD) was proposed by Brandyn Webb (also called FunkSVD) to improve SVD using a learning rate, regularization constants and a method for clipping predictions [120]. This algorithm minimizes the squared error between the actual ratings and predicted estimations for all available votes [121]. RSVD considers a regularization parameter, guarantees the convergence, and improves the generalizability of the model by optimizing the regularization parameter. For minimization process, RVD uses gradient descent that can achieve good accuracy by choosing appropriate parameters.

b: NON-NEGATIVE MATRIX FACTORIZATION

In the Non-negative MF (NMF) algorithm [122], there is a low-dimensional linear model with a non-negative constraint to indicate the rating matrix. This means that the profile of the users and items in a NMF, should have only positive values [36], [123]. This method uses multiplicative update rules for minimizing the least squares error between the actual ratings and the predicted ones.

c: PROBABILISTIC MATRIX FACTORIZATION

Probabilistic MF (PMF) utilizes a probabilistic linear model to represent the latent features of users and items. According to Mnih and Salakhutdinov [124], implementation of this algorithm on the large and sparse dataset of Netflix showed promising results. This model linearly scales with the number of ratings.

d: BAYESIAN PROBABILISTIC MATRIX FACTORIZATION

Usually, low-rank MF algorithms are fitted to the data using a Maximum-a-Posteriori (MAP) estimate of the model parameters, which in the case of inaccurate tuning of regularization parameters, will often result in over-fitting. Salakhutdinov and Mnih presented a Bayesian version of PMF with automatic controlling of all model parameters and hyper-parameters [125]. They used Markov Chain Monte Carlo (MCMC) method to train their model and applied it on the Netflix dataset. Their results indicated better accuracy than PMF method.

e: NON-LINEAR PROBABILISTIC MATRIX FACTORIZATION

Non-linear PMF (NLPMF) uses Gaussian process latent variable models for recommendation. The model is optimized using stochastic gradient descent method. Lawrence and Urtasun applied this model to EachMovie and Movielens datasets and achieved good results [126].

f: REGULARIZED SINGLE-ELEMENT-BASED NMF

Non-negative MF methods have been proposed primarily for computer vision applications, while CF problems differ from them due to their sparsity of the user-item ratings matrix. In [127], a regularized single-element-based NMF (RSNMF) model was proposed which is especially suitable for solving CF problems subject to the constraint of non-negativity.

g: ALTERNATING NON-NEGATIVE LATENT FACTOR

High computational and storage complexity are the major challenges in NMF-based CF recommender systems. To address this problem, an alternating direction method (ADM)-based non-negative latent factor (ANLF) model was proposed in [128] to apply the ADM-based optimization with regard to each single feature and obtain high convergence rate as well as low complexity.

h: INHERENTLY NON-NEGATIVE LATENT FACTOR

In [129], a novel inherently non-negative latent factor (INLF) was proposed to extract non-negative latent factors from high-dimensional and sparse matrices. To this end, a single-element-dependent mapping is used to bring the output factors and decision variables for making the parameter training unconstrained and compatible with general training schemes. Experiments are performed on six high dimensional and sparse matrices and the results showed that INLF model is able to acquire NLFs more efficiently than other methods.

i: EFFICIENT SECOND-ORDER LATENT FACTOR

In [130], an efficient second-order latent factor (ESLF) model was proposed based on second-order optimization to achieve higher accuracy. To this end, a Hessian-free optimization framework is applied to avoid direct usage of the Hessian matrix by computing its product with an arbitrary vector. Moreover, a model is considered to extract latent factors from the given incomplete matrices via a second-order optimization process. Experimental results on two datasets indicated that ESLF model can offer higher prediction accuracy with reasonable computational efficiency.

Table 1 represents the summary of CF recommender algorithms mentioned above. Note that these algorithms are only representative for recommendation algorithms and many extensions to these classical memory- and model-based have been proposed in the literature, some of which were also discussed in this manuscript.

TABLE 1. Summarization of recommender algorithms. The recommender algorithms are classified into three main groups including content-based, collaborative filtering, and hybrid methods.

Recommendation Approaches		Algorithms	Applied Techniques
Content-based RS	Heuristic-based CBF	- *K* Nearest Neighbor - Clustering	
	Model-based CBF	- Bayesian - Clustering - Neural networks	
Collaborative Filtering RS	Memory-based CF	- User-Based - Item-Based - RA - UOS - MLCF - ULPE - IGPE	- *K*-Nearest Neighbor - Graph Theory - Decision tree - Web mining - Support vector machine - Bayesian models - Clustering - Association mining -Principal Component Analysis (PCA) -Bayesian networks
	Model-based CF	- Slope 1 - Weighted Slope 1 - RSVD - NMF - PMF - BPMF - NPMF - RSNMF - ANLF - INLF - ESLF	- Clustering - Latent semantic analysis - Neural networks - Linear regression -Probabilistic models -Maximum entropy
Hybrid RS		- Feature combination - Recommendation results combination Other	- Bayesian - Clustering - Linear combination - Probabilistic models - Maximum entropy

In this manuscript, we do not consider distributed implementation of CF algorithms. However, as the size of the dataset increases, calculation of similarity scores might not be easy, and can even be impractical for very large-scale datasets. A possible approach to solve the issue is to use distributed processing approaches such as mMapReduced-based distributed framework on Hadoop, which have been shown to significantly improve the computational performance [131].

IV. EVALUATION METRICS

There are various metrics in the field of RSs for evaluating the performance of different algorithms [4], [132]. In this section we review the most common evaluation metrics to assess the performance of RSs algorithms. Evaluations can be offline or online [4], [63]. In an offline analysis, the dataset is collected and a proportion of ratings are hidden from the recommender algorithm as a test set. Then, the target algorithm uses the rest of data (i.e., the training set) to predict new ratings or rank the unseen items. Afterward, one metric or combination of evaluation metrics is used to measure the quality of recommendations. Offline analysis has the advantage of being quick, while its major drawback is that we cannot measure the true satisfaction of users regarding the recommended items. On the other hand, online evaluations are conducted in a live experiment, observing the users' behavior, recommending items and measuring their satisfaction using their feedback or tracking their acts such as click-through rate. However, conducting an online evaluation is expensive and often impossible. In this paper, we used offline analysis to evaluate recommendation algorithms on different existing datasets.

There are different perspectives on evaluation metrics; some are based on the recommendation list itself, such as accuracy, coverage, diversity and novelty, some are based on the system's or users' point of views independent of the recommender, namely confidence, robustness, adaptivity, scalability, trust, risk and privacy [133], [134]. It is logical that different metrics evaluate different aspects of algorithms [132], hence, the focus of this paper is to evaluate the algorithms discussed in section 3 using several evaluation metrics including accuracy metrics, rank-based metrics, diversity, novelty and coverage. Each algorithm first produces the predicted ratings, the results are then sorted, and for each user, the top-*N* items with the highest predicted ratings are recommended. The metrics evaluate different properties of these top-*N* items. Note that in this study, in order to have integrity among all datasets, only the ratings of users and the relationships due to these ratings are used, and other data such as timestamps of the ratings are not included.

Among the first studies of RS, Shardanand and Maes [71] used *reversal*(errors between the predicated and true ratings) as an evaluation metric. Konstan *et al.* [13] used *receiver operating characteristic curve* (ROC curve) for evaluating RS, which had been used for evaluation in the context of information filtering beforehand. Breese *et al.* [64] analyzed several CF algorithms and introduced some extensions such as *default vote* and evaluated them using *Mean Absolute Error* (MAE) and *half-life utility*. Several researchers have used accuracy metrics such as MAE and *Root Mean Squared Error* (RMSE) to evaluate RS, while some others used non-accuracy metrics. Mobasher *et al.* [135] used *coverage* as a measure of quality of recommenders that is defined as the proportion of items that recommender can suggest to users. McNee *et al.* [136] measured the degree to which recommendations where *surprising* or *non-obvious*. Others have measured the *explainability* of a recommender that is how well a recommender can explains its recommendations to users. A few studies have discussed that these metrics do not measure the users' satisfaction of the recommender, while others have argued that the users' satisfaction may not be the ultimate goal of using a recommender in some cases. In the following, we give a comprehensive overview of the metrics that can be used to evaluate the performance of CF methods where the only available information is the rating

histories. We categorized these metrics into three groups: *accuracy* metrics, *ranking-based* metrics, and *diversity, novelty and coverage*.

A. ACCURACY METRICS

Herlocker *et al.* [4] categorized accuracy metrics into three classes, namely, *predictive accuracy* metrics, *rank accuracy* metrics and *classification accuracy* metrics, which are reviewed in the following.

1) PREDICTIVE ACCURACY METRICS

In order to measure the accuracy of the predicted ratings generated by a recommender according to the real ratings (i.e., the rating in the test set) of users, predictive accuracy measures such as MAE or RMSE are frequently used in RS. These metrics are mostly useful where the predicted ratings are shown to the users of system [4]. Although these metrics are easily deployable due to their simplicity, the predictive accuracy metrics consider the ratings' space to be uniform, which is not the case in real systems [132]. Moreover, these metrics treat all ratings the same, regardless of their position in the recommendation list [61], i.e., a one-star error for an item on the top of the recommendation list penalizes the system the same as a one-star error for an item at the end of the list, which probably will not be recommended to the user ever. Thus, these metrics are not suitable in a system with the goal of *finding good items*, which means the users only care about errors of items in the top-N list [4]. As a solution, one can put more weight on the errors in the top-N items of recommendation list than the rest of the prediction list.

a: MEAN ABSOLUTE ERROR

MAE is used to measure the average absolute deviation of the predicted ratings from the real ratings of users, as:

$$MAE = \frac{1}{|\text{TestSet}|} \sum_{(u,i)\in \text{TestSet}} |\tilde{r}_{ui} - r_{ui}| \tag{20}$$

where $\tilde{r}_{ui}$ represents the predicated rating of the system for user u and item i. MAE has been used for evaluation of RSs in various studies [64], [71], [137], [138]. Often, the averaged MAE for all users is shown as a general performance of a RS. In order to compare recommenders using different rating scales, one can normalize the total MAE by dividing the mean MAE value over all users by the maximum rating ($Rating_{max}$) minus the minimum rating ($Rating_{min}$) in the system [15]:

$$NMAE = \frac{MAE}{Rating_{\max} - Rating_{\min}} \tag{21}$$

b: ROOT MEAN ABSOLUTE ERROR

RMSE is a variation of MAE which puts more emphasis on large errors by using power 2 on the deviation of predicted ratings from real ratings:

$$RMSE = \frac{1}{|\text{TestSet}|} \sum_{(u,i)\in \text{TestSet}} \sqrt{(\tilde{r}_{ui} - r_{ui})^2} \tag{22}$$

and the normalized RMSE as:

$$NRMSE = \frac{RMSE}{Rating_{\max} - Rating_{\min}} \tag{23}$$

c: ASYMMETRIC LOSS

In systems which recommending bad items as good ones is worse than recommending good items as bad ones, one can use asymmetric loss to evaluate the system as follows:

$$AL_u = \frac{1}{n} \times \sum_{i,j} loss(r_{ui}, \tilde{r}_{ui}) \tag{24}$$

If the recommended item is liked by the user, *loss* equals zero, however for an item is liked by the user but not recommended by the system, *loss* is defined as,

$$loss = r_{ui} - \tilde{r}_{ui} \tag{25}$$

and finally if a disliked item is recommended to the user, the loss is:

$$loss = (\tilde{r}_{ui} - r_{ui}) \times (1 + (Rating_{avg} - r_{ui} + 1) \times 0.5) \tag{26}$$

2) RANK ACCURACY METRICS

In order to measure the relationship between the order of the items in a recommended list with the order that each user has given to the same items, rank accuracy metrics are used. These metrics are useful when one recommends a ranked list of items to users [4].

a: PEARSON CORRELATION

Pearson Correlation measures the linear relationship between two list of predicted ratings and real ratings of a user. Hill et. al. used this metric to evaluate their recommender algorithm [139]. One should first calculate the PC for each user, and then get the average by dividing sum of all PC values by the number of users in the system, as:

$$PC_u = \frac{\sum_{i=1}^{N} (r_{ui} - \bar{r}_u)(\tilde{r}_{ui} - \bar{\tilde{r}}_u)}{\sqrt{\sum_{i=1}^{N} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i=1}^{N} (\tilde{r}_{ui} - \bar{\tilde{r}}_u)^2}} \tag{27}$$

b: SPEARMAN CORRELATION

Spearman's rank correlation coefficient or Spearman's rho, is a nonparametric metric of statistical dependence between predicted and real lists and is named after Charles Spearman [140]. The difference between Pearson and Spearman correlations is that in the latter, r_i and $\tilde{r}_i$ are the ranks of corresponding items in the real and predicted rating lists, respectively, rather than the value of ratings. Indeed, Spearman's rank correlation is Pearson correlation of the rank vectors, and is calculated as:

$$SC_u = \frac{\sum_{i=1}^{N} (r_i - \bar{r}_u)(\tilde{r}_i - \bar{\tilde{r}}_u)}{\sqrt{\sum_{i=1}^{N} (r_i - \bar{r}_u)^2} \sqrt{\sum_{i=1}^{N} (\tilde{r}_i - \bar{\tilde{r}}_u)^2}} \tag{28}$$

c: KENDALL TAU CORRELATION

Kendall rank correlation is used to measure the similarity of the orderings of two ranked lists which was proposed by Kendall [141]. It is defined as:

$$\tau = \frac{C - D}{C + D} \tag{29}$$

where C is the number of pairs of items for which the recommender predicted in the same order as real rating list of user (Concordant pairs) and D is the number of pairs of items for which the recommender predicted the wrong order (Discordant pairs). When there are no discordant pairs of items ($D = 0$), i.e., the ranking of items in the predicted and the real rating lists are exactly the same, the Kendall tau value will be 1, and for two completely dissimilar lists the value of this metric is -1 ($C = 0$). Considering a ranked list with N items, there are $N \times (N-1)/2$ ordered pairs of items, thus Eq. (29) is equal to:

$$\tau = \frac{C - D}{N(\mathrm{N} - 1)/2} \tag{30}$$

Sometimes a user gives several items the same ratings or a recommender predicts the same ratings for several items. In such cases, we use a variation of Kendall Tau correlation proposed by Herlocker *et al.* [4] that is defined as:

$$\tau \approx \frac{(C - D)}{\sqrt{(C + D + TR)(C + D + TP)}} \tag{31}$$

where TR is the number of item pairs with the same real ratings and TP is the number of items with the same predicted ratings.

The Kendall Tau does not consider the position of the correct ranked items, e.g., between the first and the second rating and between 50th and 51th ratings. One solution for this issue can be to add more weight to concordant pairs at the top of the list than those towards the end of the list [61].

d: NORMALIZED DISTANCE-BASED PERFORMANCE MEASURE

For comparing two weakly ordered ranked list, normalized distance-based performance measure (NDPM) was proposed by Yao [142] and is defined by:

$$NDPM = \frac{2D - CU}{2NP} \tag{32}$$

where D is the number of discordant pairs, CU is the number of pairs for which one system gives a tie and the other ranked list does not, and NP is the total number of pairs in the real ranked list minus tied ones. Indeed, the modified Kendall Tau correlation penalize the system even when there are tied pairs in the real ranked list; however, $NDPM$ only penalizes the recommender for tied pairs of predicted ranked list for which one item is strictly preferred in the real list.

3) CLASSIFICATION ACCURACY METRICS

In order to measure how many times the system can classify a relevant item as a good one or an irrelevant item as a bad one, classification accuracy metric are used. A relevant item is an item that the user has liked in the real ratings. For binary ratings, obviously, there are liked and disliked items and not the ratings. Usually, the disliked and non-rated items are grouped together, and hence, the liked items (sometimes showed as purchased items) are the relevant ones. However, for numerical rating scales, we define the relevant items as the items for which the user gave a higher rating than the average rating of all the items he/she voted for. For classification metrics, deviation from the real ratings is tolerated as long as the relevant items are recommended on the top-N list. Unlike the rank accuracy metrics, these metrics are most useful when evaluating recommenders with binary or unary ratings. In order to apply the classification accuracy metric for datasets with numerical rating scales, for each user we consider items rated more than his/her average rating as items which he/she has liked. Precision, recall and F1 score metrics belong to this group.

One problem with these metrics is that the items with no real rating cannot be considered as irrelevant, because the user might had not seen or consumed them at all [4]. In order to handle the sparsity of the input data, there are a number of approaches. One simple solution is to eliminate all unrated items and predict the top-N list only for items for which we have the users' rating. However, this method likely leads to biased recommendation, i.e., the items which the user has not yet consumed will never be measured and get recommended to the user. The second solution for treating sparse datasets is to consider slightly negative default ratings in recommending items that has not been rated [64]. However, the default rating may differ greatly from the true rating for unobserved item [4]. A third approach, which is employed in this manuscript, is to perform the predictions for all unobserved items, but evaluate in only for top-N items in the list.

a: PRECISION AND RECALL

Precision and recall are among the most frequently used metrics of information retrieval field introduced by Cleverdon and Kean [143] (1968). They have been among the first series of the metrics used to evaluate recommendation algorithms [54], [75], [76], [114]. These metrics use a confusion matrix that divides the items into 4 different groups (Table 2). In this matrix, relevant items which are recommended by the system, are placed in the true positive (TP) group, and those relevant items that the system failed to detect as relevant for the user go to false negative (FN) group. Irrelevant items which are incorrectly recommended by the system are placed in the false positive (FP) group, and finally, the irrelevant items that are

TABLE 2. Confusion matrix for calculating precision and recall metrics. TP represents true positive, TN represents true negative, FP means false positive and FN means false negatives.

	Recommended	Not recommended
Relevant	TP	FN
Irrelevant	FP	TN

correctly not recommended to the user are considered in the true negative (TN) group.

Precision is calculated as the ratio of the relevant items which are recommended to the number of all recommended items, as:

$$P = \frac{TP}{TP + FP} \tag{33}$$

And recall is calculated as the ratio of the relevant items which are recommended to the number of all relevant items, as:

$$R = \frac{TP}{TP + FN} \tag{34}$$

As mentioned before, precision and recall are not directly applicable to evaluate recommendation algorithms, and we need to know each item is relevant or not, which means every item must be rated by the user. Thus, we used $P@N$ and $R@N$ instead (N being the size of the recommendation list) defined by [76]:

$$P@N_u = \frac{TP}{N} \tag{35}$$

$$R@N_u = \frac{TP}{|\,\mathrm{Rel}_u\,|} \tag{36}$$

where Rel_u is the set of the items relevant to user u.

b: F1 SCORE

Since precision and recall are inversely correlated [143], it is needed to consider both of them when evaluating different algorithms. Moreover, $P@N_u$ and $R@N_u$ are dependent on the length of the recommendation list. Therefore, researchers have often used F1 score, as a combination of precision and recall, as defines by [75], [76].

$$F1_u = \frac{2(P@N_u) \times (R@N_u)}{(P@N_u) + (R@N_u)} \tag{37}$$

B. RANK-BASED METRICS

Instead of comparing the exact value of the predicted ratings with the real ones, rank-based metrics examine the order and position of the items displayed to the user in the recommendation list. *Half-Life Utility, Discounted Cumulative Gain, Rank-Biased Precision and Recovery Rate* are among this group of metrics.

1) HALF-LIFE UTILITY

Half-life utility metric evaluates the utility of a recommendation list based on a hypothesis that as the rank of the item in the recommendation list decreases, the probability of user's tendency to examine it reduces exponentially [64]. Since users usually tend to pay attention to items at the top of the recommended list, a "half-life" threshold is defined as the rank of the item on the list for which there is a 50-50 chance that user will examine it. Half-life utility is defined by:

$$H_u = \sum_{i=1}^{N} \frac{\max(r_{ui} - d, 0)}{2^{(i-1)/(h-1)}} \tag{38}$$

where h is the half-life threshold and d is the neutral vote (we set it as the user's average rating. For simulations, we set $h = 5$ as suggested by Breese *et al.* [64]. We calculated H_u for each user and obtained the average over all users to compute the overall H_{total} score; the greater the value of H_{total}, the better recommender acts according to this metric. This index is calculated as:

$$H_{total} = \frac{\sum_{u=1}^{m} H_u}{m} \tag{39}$$

2) NORMALIZED DISCOUNTED CUMULATIVE GAIN

Discounted Cumulative Gain (DCG) evaluates the usefulness of an item based on its rank in a recommended list. The more relevant items are with higher ranks, the more valuable the recommendation list is for the user and one becomes more satisfied with the system which saves his/her time. DCG is defined by [144]:

$$DCG\_b_u = \sum_{i=1}^{b} R_i + \sum_{i=b+1}^{N} \frac{R_i}{\log_b r_i} \tag{40}$$

DCG is calculated for each user regarding his/her real and predicted lists. r_i is the rank of the item in the recommendation list, e.g., for the first item at the top of the list we have $r_i = 1$. The lower is the rank of the item in the recommendation list (i.e., toward the end of the list), its share in the cumulated gain becomes less. The discounting function to reduce this share is log-harmonic. Different values of b, as the base of the logarithm, control the degree of reduction in items' shares in DCG; i.e., the greater the values of b, the slower the shares decrease. For the item at rank 1 and smaller than b, it is logical to use the first part of Eq. (4), rather than using the logarithmic discount function. According to [144], we set $b = 2$ for the experiments. R_i indicates the relevancy of item i (with rank r_i) in the recommendation list; as mentioned before we consider an item relevant to a user, if he/she has rated it more than the average rating of the total items he/she voted. If the item is relevant, we have $R_i = 1$, otherwise $R_i = 0$. For comparison purposes, we need to eliminate the effect of different sizes of the recommendation lists in the metric. The computed DCG value is normalized through dividing by the maximum possible gain (i.e., the perfect ranking according to user's preference), which is the exact order of items given by the user in real rating list:

$$NDCG_u = \frac{DCG_u}{DCG_{\max}} \tag{41}$$

We calculated $NDCG_u$ for each user and get the average over all users, to compute the overall $NDCG_{total}$ score for the dataset, as:

$$NDCG_{total} = \frac{\sum_{u=1}^{m} NDCG_u}{m} \tag{42}$$

This score will be between 0 and 1. The greater is the value of $NDCG_{total}$, the better the recommender works for

relevant items regarding their ranks in the recommendation list.

3) RANK-BIASED PRECISION

Similar to DCG, the Rank-biased Precision (RBP) metric attempts to evaluate the recommendation list through giving more shares to highly ranked relevant items for each user. The difference here is the discounting function which is a geometric sequence instead of the logarithmic one in DCG [145]. The underlying assumption here is that users often examine the first item of the recommended list, and then with probability p, they may check the next item, i.e., there is $(1-p)$ chance that they will not choose the next item. For example, considering $p = 0.8$, the user will check the first item, then examines the second item with probability of 0.8, the third item with probability of 0.8^2, and finally the i^{th} item with probability of 0.8^{i-1}. RBP for each user's recommended list is defined by:

$$RBP_u = (1-p)\sum_{i=1}^{N}(R_i \times p^{i-1}) \tag{43}$$

where r_i and R_i are defined the same as for DCG. Since $\sum_{i=1}^{\infty} p^{r_i-1} = 1/(1-p)$, the value of RBP is between 0 and 1; the greater this value, the better the system performs according to RBP. The overall RBP averaged for all users is calculated as:

$$RBP_{total} = \frac{\sum_{u=1}^{m} RBP_u}{m} \tag{44}$$

The performance of the recommender depends on the choice of p such that for small values of p, the user only examines the top-ranked items, while for large values of p, it may also examines the items in lower ranks.

4) RECOVERY RATE

Recovery Rate evaluates the performance based on the correct ranking of the items. The value of Recovery metric for user u is defined as:

$$\text{Recovery}_u = \frac{1}{|N_R|} \times \sum_{i \in N_R} \frac{r_i}{C_i} \tag{45}$$

where N_R is the set of relevant items in the real ratings list of user u, r_i indicates the rank of item i in the recommendation list, and C_i is the number of candidate items to recommend to user u. The average Recovery rate for the users is calculated as:

$$\text{Recovery}_{total} = \frac{\sum_{u=1}^{m} \text{Recovery}_u}{m} \tag{46}$$

C. DIVERSITY, NOVELTY AND COVERAGE

So far, we reviewed the accuracy and ranking of recommendations generated by recommenders. However, we need to go further and evaluate them based on other criteria; measures to answer questions such as: what percentage of items in the recommendation lists is new to user; how much they surprise the user; how many of them are popular items; do the algorithm always recommend the same group of items to all users; how much similar are the users' recommendation lists; and alike. Recently, there has been a shift in the community to design recommenders that not only provide accurate recommendations, but also brings the most satisfaction to the users [146]–[148]. Indeed, in many real applications, the users would like to be recommended diverse and novel items. In this section, we review a number of metrics developed to measure how much a recommender is novel, diverse or covers the items available in the system.

1) DIVERSITY

Diversity is a metric with two concepts: intra-diversity and inter-diversity. Intra-diversity is to measure how different are the items recommended to a user, i.e., the diversity of each recommendation list. To this end, we need to measure the similarities between item pairs in the recommendation lists. This similarity can be obtained from the content information of items or using similarity measures such as cosine or Pearson similarity for item rating vectors [149]. The Intra-diversity of N items of recommendation list for user u can be calculated as:

$$IntraDiversity_u = \frac{1}{N(\mathrm{N}-1)} \times \sum_{i \neq j} s(I_i, I_j) \tag{47}$$

where $s(I_i, I_j)$ is the similarity between items i and j. Note that the lower is the value of *IntraDiversity*, the more diverse items the system recommends to the user. In order to compute the overall $IntraDiversity_{total}(N)$ for all users of a data set, one can calculate $IntraDiversity_u(N)$ for each user and then get the average of them, as:

$$\text{IntraDiversity}_{total} = \frac{\sum_{u=1}^{m} IntraDiversity_u}{m} \tag{48}$$

Inter-diversity indicates the extent of difference between the recommendation lists of all users [150]. We can measure this notion using Hamming distance between the recommendation lists of users' u and u' as:

$$H_{uu'} = 1 - (\frac{Q_{uu'}}{N}) \tag{49}$$

where $Q_{uu'}$ is the similarity between the two users' recommendation lists; this similarity is defined as the number of common items in both lists. It is worth mentioning that the value of this metric is highly dependent on the number of users and the items. The average metric can be calculated as:

$$H_u = \frac{1}{m(m-1)} \sum_{u=1}^{m} \sum_{u \neq u'} H_{uu'} \tag{50}$$

2) SELF-INFORMATION BASED NOVELTY

Although we assume that the users would like the items similar to what they have liked in the past, it is also a logical assumption that most users would also want to get recommendations for items they have not yet seen, even the items they won't expect. If the novel recommendations get users' attention and they like them, the recommender is successful from this point of view. A number of metrics has been proposed for evaluating RSs novelty [63] and several researchers have used novelty to evaluate their recommenders [6], [123], [151], [152]. In order to define a proper measure for the novelty, let us first define a metric to measure popularity of the recommendations. For more popular items, it is more probable that a user have seen or consumed them, and thus, they are less novel for him/her. One can simply measure the popularity of an item by its degree, resulting in the average popularity score as:

$$Popularity = \frac{1}{m \times \mathrm{N}} \sum_{u=1}^{m} \sum_{i=1}^{N} d_i \tag{51}$$

where d_i is the degree of item i. According to a variation of novelty which uses the notion of popularity, namely Self-Information Based Novelty (SIBN), also known as surprisal or unexpectedness, popular items are less novel. SIBN for item i, $SIBN_i$, is defined as [123],

$$SIBN_i = \log_2(\frac{m}{d_i}) \tag{52}$$

The average SIBN can be simply obtained by making the average of the above relation over all users. For a user u, we can calculate SIBN of all top-N items recommended to him/her. We can also use an altered version of this metric, effective novelty, which only considers the novelty of items relevant to each user from the top-N recommendation list [151], [152]. If we are evaluating a recommender based on the novelty of its recommended items, it is logical to consider only the novelty of those items which actually the user would like and consume. Effective SIBN is defined as:

$$ESIBN = \frac{1}{m} \sum_{i=1}^{N} R_i \times SIBN_i \tag{53}$$

3) COVERAGE

A good recommender not only recommends likeable items to users, but also covers a wide variety of items which the system has to offer. In its simplest form, coverage indicates that as the percentage of the items a recommender can suggest grow higher, the system is performing better. Note that lower values of coverage implies less diversity [61], [137].

a: CATALOG COVERAGE

Catalog coverage refers to the percentage of distinct items in the top-N recommendation lists of users:

$$C = \frac{I_r}{n} \tag{54}$$

where n is the total number of items in the system and I_r indicates the total number of distinct items in users' top-N lists. An improvement to this definition is to consider only the items which are relevant to a user.

b: ENTROPY COVERAGE

In addition to the number of items a recommender covers, the frequency of offering different items may vary; some items are frequently recommended to various users (probably popular ones), and some appears less in the recommendation lists. To consider how many times an item is recommended to the users, one can use a modified version of the coverage presented above. Entropy Coverage, *EC,* is defined as:

$$EC = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{55}$$

where p_i is the percentage of the recommendation lists that contains item i.

4) UNIFIED EVALUATION METRIC

In [147], a novel method is proposed to combine different evaluation metrics as a unified metric for evaluating the performance of recommendation systems. One can choose a number of metrics and properly combine them to have a unified evaluation metric. Here, we choose diversity, novelty, coverage, precision, and RBP, as calculated using Eqs. (50), (52), (54), (33), and (44), respectively. Then, the unified metric is calculated based on the selected metrics using the following equation:

$$UM\left(\overline{H}, \overline{N}, \overline{C}, \overline{P}, \overline{RBP}\right) = \frac{5}{\frac{1}{\overline{H}} + \frac{1}{\overline{N}} + \frac{1}{\overline{C}} + \frac{1}{\overline{P}} + \frac{1}{\overline{RBP}}} \tag{56}$$

where, $\overline{H}$, $\overline{N}$, $\overline{C}$, $\overline{P}$, and $\overline{RBP}$ are the normalized values for the diversity, novelty, coverage, precision, and RBP metrics, respectively. [147] has proposed an approach to compute the normalized values. Table 3 summarizes the evaluation metrics discussed above.

V. RESULTS AND DISCUSSION

In order to compare the performance of RSs in terms of different evaluation metrics, we used a machine with Intel(R) Corei7-3632 CPU @ 2.1GHz and 6GB of main memory, and JAVA language on a Unix-based operating system (Ubuntu 12.04). All the parameters affecting the computation time (e.g. neighborhood size, learning rate, and number of iterations) were set the same for all algorithms. We used 10-fold cross-validation strategy to compare the results of the recommendation methods. In other words, each dataset is divided into ten folds and in each run nine folds are used as the training set and the remaining fold is used as the test set. The experiments were performed in 10 independent runs, where each time one of the folds was randomly selected and used as the test set. Finally, the average results over these 10 runs are reported. In the training phase of each recommendation method, first a model is constructed and similarity values

TABLE 3. Summarization of evaluation metrics. The evaluation metrics are classified into four groups including accuracy-based metrics, rank-based metrics, diversity-, novelty-, and coverage-based metrics, and unified metric.

		Symbol	Metric	The lower (or higher), the better
1	Accuracy metrics	NMAE	Normalized Mean Absolute Error	Lower
2		NRMSE	Normalized Root Mean Square Error	Lower
3		AL	Asymmetric Loss	Lower
4		PC	Pearson Correlation	Higher
5		T	Kendall's Tau Correlation	Higher
6		SC	Spearman Correlation	Higher
7		NDPM	Normalized Distance-based Performance Measure	Lower
8		P	Precision	Higher
9		R	Recall	Higher
10		F1	F1 Score	Higher
11	Rank-based metrics	HLU	Half-Life Utility	Higher
12		NDCG	Normalized Discounted Cumulative Gain	Higher
13		RBP	Rank-Biased Precision	Higher
14		RR	Recovery Rate	Lower
15	Diversity, Novelty and coverage	ID	Intra-list Diversity	Higher
16		H	Hamming Distance (Diversity)	Higher
17		Pop	Popularity	Lower
18		SIBN	Self-Information Based Novelty	Higher
19		ESIBN	Effective Self-Information Based Novelty	Higher
20		C	Catalog Coverage	Higher
21		EC	Entropy Coverage	Higher
22	Unified Metric	UM	Unified Metric	Higher

are computed. Then, the model is used in the test phase to predict ratings for the unseen items and also a list of recommendations is generated to the users. The execution time of a recommendation method refers to both training time (i.e. offline execution time) and test time (i.e. online execution time).

A. DATASET

We used five different datasets including Movielens,[3] Epinions,[4] LastFM,[5] BookCrossing[6] and Jester[7] to perform the experiments. The Movielens dataset is one of the most frequently used datasets within the community of RSs. In Movielens, items are movies and the ratings are in the scale of 1 to 5 stars. This dataset is gathered over various periods of time, depending on the size of the set, several versions of this dataset are available. We used Movielens_10M dataset in this study, in which users with less than 20 ratings have already been removed. It contains 10 million ratings applied to 10,000 movies by 72,000 users. The Epinions dataset includes the opinions of users about various types of commercial items as numerical ratings in the range of 1 to 5. This dataset consists of 49,290 users who rated at least once on 139,738 items. The LastFM is a music recommendation dataset which contains a list of top most listened songs for each user. There are 2100 users and 18,745 songs in this dataset. It should be noted that there is no ratings in a specific range in LastFM dataset. Instead, this dataset includes the number of times those songs were played by the users. Therefore, we normalized this information into a range between 0 and 1 for our experiments. The BookCrossing is a book recommendation dataset which contains 1.1 million ratings of 270,000 books by 90,000 users. The ratings are on a scale from 1 to 10, and implicit ratings are also included. The Jester dataset contains the ratings about 100 jokes which are provided by 24,938 users. In this dataset, the users have rated between 15 and 35 jokes, and these ratings are real numbers from -10 to 10.

[3]https://grouplens.org/datasets/movielens/
[4]http://www.trustlet.org/epinions.html
[5]https://grouplens.org/datasets/hetrec-2011/
[6]http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[7]http://eigentaste.berkeley.edu/dataset/

B. PERFORMANCE EVALUATION

We implemented the memory-based and model-based CF algorithms introduced in the previous sections, and the results are summarized in Tables 4-8 for Movielens, Epinions, LastFM, BookCrossing and Jester datasets, respectively. In the tables, the top-performer algorithm is shown in boldface, while the one with the second best performance is highlighted in underlined boldface. The results reveal that the top-performing algorithms are spread across different evaluation metrics. Furthermore, they exhibit rather different profile for different datasets. For example, while UOS is among the best-performers in Movielens and BookCrossing datasets based on the precision and F1 measures, it has poor performance in other datasets. User-based CF and MLCF show weak performance in Jester dataset, being the top-performer or the second top-performer in terms of none of the evaluation metrics. However, they are among the medium-level performers in Movielens dataset. BPMF, PMF, UOS and IGPE are the top-performers in Movielens dataset. Slope 1 is among the top-performers in Epinions and Jester datasets. PMF is among those with top performance in Epinions and

LastFM datasets. Moreover, RSVD has high performance on Epinions dataset in terms of several evaluation measures. One can also make pairwise comparisons between the algorithms. For example, item-based CF often has better performance than user-based CF. However, it has expensive computations to test the algorithm, which is not a burden in real scenarios in which only the training phase of the algorithm matters.

The performed experiments show that NMAE and NRMSE metrics might not have significant correlation. NMAE is used to measure the closeness of predicted ratings to the ground-true rating scores. To calculate the absolute error value, the predicted ratings are compared with the real ones. This procedure is repeated for all of the taken-out ratings, and then an average of all the values is considered as the final NMAE measure reported in the tables. On the other hand, NRMSE shows contributions of the absolute errors between the predicted and actual ratings. It is clear that lower NMAE and NRMSE correspond to higher prediction accuracy. NMAE considers every error of equal value, while NRMSE squares the error before summing it and tends to penalize large errors more heavily. Therefore, these two metrics may have no correlations theoretically. Based on the experimental results, NMAE and NRMSE metrics are completely correlated for the LastFM and BookCrossing datasets as shown in Tables 6 and 7. Moreover, the results show that NMAE and NRMSE metrics are correlated just for one of the best methods based on the Epinions (Table 5) and Jester (Table 8); NMAE and NRMSE metrics are correlated for RSVD method and are not correlated for Item-based and NMF methods based on the Epinions dataset. In addition, NMAE and NRMSE metrics are correlated for Item-based method and not for UOS and BPMF methods based on the Jester dataset. Finally, the results of the experiments based on the Movielens dataset show that NMAE and NRMSE metrics are correlated for the best methods. BPMF and NMF methods are respectively the best and second best predictors based on both NMAE and NRMSE metrics.

Our experiments show that there is no golden recommendation algorithm showing the best performance in all of the evaluation metrics. Furthermore, there is rather large variability in the position of the algorithms in terms of different evaluation metrics across different datasets. Therefore, there would not be a *best choice* (i.e., golden algorithm), and one should choose the most useful approach and corresponding metrics according to the recommender's goals and application, while considering the limitation it would face regarding the data and time available. Often, each research work considers some of these evaluation metrics and assesses the performance of their proposed algorithm on them. However, to have a fair conclusion on the performance, one should consider various evaluation metrics and apply the algorithms on various datasets.

The unified metric as expressed by equation (56) makes it possible to better compare the algorithms and find the best performing one. Based on this metric, PMF is the best algorithm in all datasets except for LastFM dataset for which

TABLE 4. Performance of CF algorithms on different evaluation metrics - Movielens dataset. The best result for each metric is shown in boldface, while the second best result is shown in underlined boldface.

Category		Accuracy-based					Diversity and novelty			Coverage		Rank-based			Unified	Time (sec)	
Metrics/algorithms		NMAE	NRMSE	τ	P@10	R@10	F1	H	Pop	N	EN	C	EC	HLU	NDCG	RBP	RR	UM	training	testing
memory-based approach	User-Based	0.256	0.391	0.254	0.002	0.029	0.004	0.381	0.472	4.976	0.112	0.183	0.204	0.072	0.362	0.085	0.219	0.010	**22.172**	513.35
	Item-Based	0.262	0.385	0.247	0.091	0.105	0.098	0.403	0.351	4.384	**0.354**	0.273	0.196	0.193	0.395	0.121	**0.186**	**0.184**	39.145	9813.2
	RA	0.268	0.416	0.271	0.034	0.043	0.038	**0.634**	0.511	4.539	0.152	0.356	0.175	0.098	0.369	0.093	0.214	0.108	28.211	689.91
	UOS	0.281	0.428	0.114	**0.113**	0.094	**0.103**	0.452	0.412	4.517	0.143	**0.412**	0.113	0.174	0.315	0.058	0.359	0.154	25.361	389.24
	MLCF	0.283	0.432	0.151	0.043	0.101	0.060	0.501	0.418	4.697	0.125	0.312	0.102	0.229	0.283	0.118	0.354	0.129	**23.215**	493.21
	ULPE	0.297	0.451	0.315	0.085	**0.141**	**0.106**	0.362	0.515	5.315	0.226	0.241	0.094	0.216	0.262	0.115	0.251	0.173	43.127	525.31
	IGPE	0.295	0.447	**0.557**	0.052	**0.126**	0.074	0.253	0.456	6.128	0.175	0.274	0.076	**0.247**	0.219	**0.163**	0.228	0.146	365.24	583.32
model-based approach	Slope 1	0.265	0.413	0.268	0.003	0.044	0.006	0.242	0.558	5.812	0.093	0.125	0.152	0.101	0.385	0.061	0.312	0.014	57.148	436.56
	W-Slope 1	0.292	0.445	0.274	0.005	0.049	0.009	0.281	0.573	5.763	0.081	0.226	0.225	0.142	0.388	0.066	0.197	0.022	56.327	413.62
	RSVD	0.251	0.377	0.259	0.071	0.061	0.066	0.294	**0.291**	4.856	0.292	0.256	0.209	0.151	**0.524**	0.102	**0.192**	0.152	215.34	339.68
	NMF	**0.248**	**0.361**	0.286	0.099	0.062	0.076	0.234	0.382	6.215	0.084	0.118	0.213	0.123	0.436	0.076	0.326	0.134	49.046	**324.52**
	PMF	0.272	0.424	0.252	**0.125**	0.076	0.095	**0.612**	0.356	**6.983**	**0.368**	0.277	**0.228**	0.231	0.497	**0.146**	0.225	**0.235**	1359.1	358.42
	BPMF	**0.235**	**0.352**	0.281	0.107	0.034	0.052	0.276	**0.259**	**7.217**	0.193	0.253	**0.231**	**0.254**	**0.546**	0.089	0.273	0.171	47.123	**289.34**
	NLPMF	0.286	0.439	**0.334**	0.021	0.052	0.029	0.546	0.298	5.211	0.249	**0.536**	0.216	0.202	0.403	0.142	0.351	0.083	532.61	986.14

TABLE 5. Performance of CF algorithms on different evaluation metrics – EPINIONS dataset. The best result for each metric is shown in boldface, while the second best result is shown in underlined boldface.

Category		Accuracy-based						Diversity and novelty			Coverage		Rank-based			Unified	Time (sec)	
Metrics/algorithms		NMAE	NRMSE	τ	P@10	R@10	F1	H	Pop	N	EN	C	EC	HLU	NDCG	RBP	RR	UM	training	testing
memory-based approach	User-Based	0.216	0.291	0.341	0.030	0.069	0.042	0.111	0.004	2.470	0.527	0.112	0.002	0.266	0.204	0.060	0.241	0.071	**5.417**	153.72
	Item-Based	**0.206**	0.284	0.016	0.013	0.002	0.004	0.047	**0.001**	1.883	0.031	0.093	0.002	0.003	0.003	0.001	0.356	0.004	6.208	3762.5
	RA	0.238	0.333	0.189	0.032	0.071	0.044	0.113	0.004	2.500	0.534	0.130	0.002	0.289	0.207	0.065	0.251	0.077	5.792	246.27
	UOS	0.256	0.330	0.041	0.004	0.008	0.005	0.009	**0.001**	0.076	0.005	0.007	0.0001	0.010	0.007	0.002	**0.003**	0.005	6.714	106.78
	MLCF	0.272	0.358	**0.346**	0.028	0.061	0.038	0.082	0.003	2.111	0.279	0.104	0.001	0.210	0.171	0.049	0.204	0.062	**5.631**	134.88
	ULPE	0.316	0.404	0.305	0.019	0.013	0.015	0.016	0.012	0.894	0.029	0.054	0.0001	0.146	0.083	0.023	**0.025**	0.027	8.347	169.54
	IGPE	0.287	0.377	**0.490**	0.020	0.029	0.024	0.019	0.003	0.875	0.021	0.058	0.002	0.110	0.089	0.023	0.044	0.030	102.38	195.33
model-based approach	Slope 1	0.353	0.465	0.072	0.071	0.095	0.082	0.289	0.004	**4.292**	**0.629**	0.191	0.004	**0.406**	0.283	0.206	0.352	0.174	15.305	118.19
	W-Slope 1	0.324	0.451	0.086	0.083	0.103	0.092	0.352	0.008	3.014	0.587	0.253	0.007	0.384	0.297	0.203	0.327	0.197	15.411	117.22
	RSVD	**0.209**	**0.268**	0.106	**0.135**	**0.139**	**0.137**	0.383	**0.002**	3.465	**0.682**	**0.311**	0.104	0.297	0.304	**0.238**	0.301	**0.265**	53.248	97.315
	NMF	0.221	**0.283**	0.173	0.096	0.120	0.107	0.259	0.006	3.957	0.416	0.117	**0.118**	0.263	0.328	0.186	0.319	0.170	12.036	**95.647**
	PMF	0.227	0.299	0.124	**0.134**	**0.143**	**0.138**	**0.498**	0.004	4.156	0.495	0.284	**0.159**	0.347	**0.375**	**0.215**	0.294	**0.265**	817.92	99.305
	BPMF	0.215	0.318	0.235	0.112	0.086	0.097	0.436	**0.002**	4.048	0.314	0.206	0.114	**0.412**	0.341	0.179	0.267	0.218	10.127	**91.114**
	NLPMF	0.241	0.347	0.259	0.098	0.094	0.096	**0.487**	0.005	**4.453**	0.367	**0.309**	0.109	0.369	**0.402**	0.193	0.251	**0.229**	158.35	346.22

TABLE 6. Performance of CF algorithms on different evaluation metrics - LastFM dataset. The best result for each metric is shown in boldface, while the second best result is shown in underlined boldface.

Category		Accuracy-based						Diversity and novelty			Coverage		Rank-based			Unified	Time (sec)	
Metrics/algorithms		NMAE	NRMSE	τ	P@10	R@10	F1	H	Pop	N	EN	C	EC	HLU	NDCG	RBP	RR	UM	training	testing
memory-based approach	User-Based	**0.002**	0.008	0.242	0.008	0.013	0.010	0.027	**0.004**	1.025	0.101	0.016	0.000	0.000	0.018	0.005	**0.181**	0.012	**3.647**	156.24
	Item-Based	**0.001**	**0.003**	0.026	0.025	0.001	0.002	0.353	**0.003**	5.244	0.001	0.087	0.001	0.000	0.000	0.000	0.935	N/A	32.141	4325.7
	RA	**0.002**	**0.004**	0.016	0.055	0.067	0.060	**0.371**	0.005	5.049	**0.729**	0.100	0.002	0.001	0.091	0.038	0.967	0.086	4.018	284.59
	UOS	**0.002**	0.005	0.028	0.017	0.030	0.022	0.076	**0.004**	1.928	0.239	0.029	0.001	0.000	0.001	0.016	0.458	0.029	6.347	128.34
	MLCF	0.003	0.005	0.023	0.017	0.030	0.021	0.066	**0.004**	1.784	0.220	0.031	0.001	0.000	0.003	0.014	0.439	0.028	**3.914**	149.38
	ULPE	0.011	0.046	0.132	**0.171**	**0.176**	**0.173**	0.259	0.411	2.301	**0.725**	0.005	0.001	0.003	0.005	0.059	0.318	0.022	26.347	186.35
	IGPE	0.023	0.081	**0.279**	0.123	**0.143**	0.132	0.221	0.386	2.025	0.465	0.018	0.005	0.002	0.003	**0.108**	0.294	0.062	110.36	287.36
model-based approach	Slope 1	0.017	0.056	**0.246**	0.006	0.008	0.007	0.268	0.435	4.289	0.117	0.012	0.124	0.001	0.107	0.004	0.362	0.010	46.068	124.12
	W-Slope 1	0.021	0.069	0.221	0.007	0.012	0.009	0.312	0.412	4.873	0.106	0.146	0.169	0.001	0.112	0.006	0.335	0.016	44.251	124.26
	RSVD	0.005	0.009	0.189	0.083	0.107	0.093	0.356	0.116	3.654	0.564	0.187	0.146	0.004	0.236	0.047	**0.276**	0.116	63.218	115.39
	NMF	0.012	0.034	0.213	0.056	0.024	0.034	0.224	0.392	**5.648**	0.149	0.214	**0.203**	0.002	0.259	0.101	0.432	0.132	14.283	**98.321**
	PMF	0.016	0.041	0.156	**0.179**	**0.143**	**0.159**	**0.398**	0.185	5.268	0.632	**0.295**	**0.249**	**0.102**	0.298	**0.149**	0.384	**0.259**	889.65	131.28
	BPMF	0.008	0.024	0.182	0.108	0.115	0.111	0.341	0.210	4.783	0.327	0.265	0.196	**0.007**	**0.365**	0.089	0.446	0.176	9.116	**106.38**
	NLPMF	0.020	0.053	0.238	0.112	0.125	0.118	0.365	0.256	**5.487**	0.279	**0.324**	0.104	0.003	**0.489**	0.103	0.351	**0.267**	219.57	416.58

TABLE 7. Performance of CF algorithms on different evaluation metrics - BookCrossing dataset. The best result for each metric is shown in boldface, while the second best result is shown in underlined boldface.

Category		Accuracy-based					Diversity and novelty			Coverage		Rank-based			Unified	Time (sec)	
Metrics/algorithms		NMAE	NRMSE	τ	P@10	R@10	F1	H	Pop	N	EN	C	EC	HLU	NDCG	RBP	RR	UM	training	testing
memory-based approach	User-Based	0.183	0.246	0.413	0.013	0.009	0.010	0.074	**0.001**	2.320	0.401	0.004	0.000	0.278	0.165	0.045	0.090	0.014	**8.056**	169.37
	Item-Based	0.196	0.285	0.392	0.087	**0.107**	0.096	0.087	0.005	1.826	0.483	0.054	0.003	0.241	0.124	0.157	0.053	0.098	18.503	4012.3
	RA	0.187	0.248	0.125	0.026	0.015	0.019	0.225	**0.002**	4.497	**0.854**	0.011	0.001	**0.631**	0.283	0.102	0.181	0.034	9.245	274.76
	UOS	0.238	0.324	0.254	**0.149**	**0.172**	**0.160**	**0.446**	0.004	3.014	0.362	0.215	0.002	**0.315**	0.108	0.037	0.246	0.117	9.923	129.43
	MLCF	0.160	0.212	0.457	0.011	0.007	0.008	0.053	**0.001**	1.959	0.179	0.004	0.000	0.227	0.139	0.037	0.074	0.013	**8.752**	157.04
	ULPE	**0.154**	**0.202**	**0.732**	0.005	0.002	0.003	0.001	0.003	0.258	0.016	0.000	0.000	0.028	0.020	0.005	**0.003**	N/A	13.071	184.11
model-based approach	IGPE	0.169	0.246	**0.851**	0.012	0.081	0.021	0.213	0.007	1.008	0.094	0.027	0.012	0.104	0.149	0.094	0.028	0.035	139.21	217.05
	Slope 1	0.225	0.318	0.312	0.047	0.054	0.050	0.164	0.052	**5.234**	0.108	0.054	0.268	0.183	0.235	0.008	0.102	0.029	24.201	138.22
	W-Slope 1	0.213	0.297	0.306	0.073	0.071	0.072	0.208	0.073	4.829	0.116	0.176	**0.312**	0.197	0.221	0.016	0.096	0.057	24.705	136.57
	RSVD	**0.143**	**0.195**	0.285	0.105	0.101	0.103	0.256	**0.002**	4.136	0.387	**0.259**	0.162	0.253	0.298	0.135	0.116	0.191	71.103	**109.32**
	NMF	0.201	0.252	0.349	**0.142**	0.087	**0.108**	0.315	0.047	5.031	0.207	0.193	0.281	0.102	0.342	0.109	0.103	0.194	19.140	**106.21**
	PMF	0.213	0.274	0.316	0.113	0.093	0.102	0.378	0.025	3.625	**0.583**	0.217	**0.296**	0.286	0.461	**0.194**	0.083	**0.218**	893.07	123.14
	BPMF	0.172	0.246	0.274	0.097	0.009	0.016	0.341	0.004	3.947	0.342	0.109	0.251	0.248	**0.496**	**0.163**	0.067	0.166	21.514	111.52
	NLPMF	0.235	0.310	0.339	0.108	0.103	0.105	**0.492**	0.067	**5.892**	0.438	**0.327**	0.203	0.180	**0.534**	0.118	**0.021**	**0.210**	181.12	397.61

TABLE 8. Performance of CF algorithms on different evaluation metrics - Jester dataset. The best result for each metric is shown in boldface, while the second best result is shown in underlined boldface.

Category		Accuracy-based					Diversity and novelty			Coverage		Rank-based			Unified	Time (sec)	
Metrics/algorithms		NMAE	NRMSE	τ	P@10	R@10	F1	H	Pop	N	EN	C	EC	HLU	NDCG	RBP	RR	UM	training	testing
memory-based approach	User-Based	0.223	0.287	0.115	0.104	0.088	0.095	0.497	0.142	2.530	0.716	0.530	0.089	0.834	0.031	0.200	0.497	0.235	**1.016**	136.12
	Item-Based	**0.176**	**0.228**	0.098	**0.134**	0.058	0.081	**0.606**	0.083	5.202	0.782	0.490	0.179	0.921	0.086	0.271	0.614	**0.309**	**0.963**	3719.4
	RA	0.219	0.279	0.159	0.091	0.081	0.086	0.549	0.169	2.247	0.518	0.500	0.077	0.954	0.106	0.179	0.469	0.213	1.857	241.17
	UOS	**0.200**	0.256	0.142	0.087	0.073	0.080	0.472	0.138	2.323	0.633	0.450	0.078	0.759	0.084	0.214	0.498	0.213	2.196	115.21
	MLCF	0.201	0.263	0.105	0.088	0.073	0.080	0.362	0.131	2.051	0.408	0.540	0.076	0.763	0.184	0.179	0.441	0.200	1.524	127.64
	ULPE	0.281	0.374	**0.532**	0.092	0.074	0.082	0.223	0.014	5.276	0.457	0.079	0.058	0.345	0.261	0.053	**0.255**	0.104	12.843	152.91
model-based approach	IGPE	0.260	0.328	**0.907**	0.050	0.051	0.050	0.156	**0.003**	4.642	0.381	0.060	0.017	0.106	0.127	0.073	**0.102**	0.086	91.238	186.19
	Slope 1	0.267	0.341	0.005	0.079	**0.125**	0.097	0.584	0.054	**6.327**	**0.846**	0.570	0.227	**0.973**	0.254	0.379	0.738	0.252	24.176	112.53
	W-Slope 1	0.282	0.354	0.006	0.061	0.105	0.077	0.508	0.053	5.830	**0.821**	0.570	0.222	**0.961**	0.235	0.379	0.738	0.209	23.725	113.70
	RSVD	0.215	0.264	0.146	0.097	**0.117**	0.106	0.527	0.041	5.126	0.536	**0.635**	0.182	0.587	0.278	**0.431**	0.615	0.286	51.382	102.41
	NMF	0.219	0.273	0.161	0.041	0.049	0.045	0.391	0.048	**6.243**	0.347	0.416	**0.263**	0.412	0.314	0.254	0.506	0.145	5.720	**85.108**
	PMF	0.228	0.285	0.120	**0.142**	0.108	**0.123**	0.564	0.051	6.016	0.793	0.482	**0.245**	0.216	**0.372**	**0.468**	0.482	**0.352**	689.38	119.26
	BPMF	0.209	**0.242**	0.115	0.082	0.096	0.088	0.432	**0.008**	5.978	0.468	0.289	0.209	0.649	**0.397**	0.224	0.539	0.212	3.421	**96.534**
	NLPMF	0.253	0.297	0.246	0.110	0.115	**0.112**	**0.671**	0.017	5.134	0.394	**0.746**	0.174	0.771	0.365	0.193	0.470	0.271	167.15	376.24

it is the second best algorithm. Note that the definition of the unified metric is based on combining five different evaluation metrics, and one can consider other metrics as well in defining the unified metric.

In the experiments, scalability metric is evaluated based on execution time of the recommendation methods. The execution time refers to two different modes including training time (i.e. offline execution time) and test time (i.e. online execution time). In the training phase, a model is constructed (for model-based CF) or similarity values are calculated (for memory-based CF). The constructed model and calculated similarity values are used in the test phase to predict ratings for the unseen items. It should be noted that in some recommendation methods the stopping criteria is needed to be initialized before performing the main procedure of the methods. These criteria are set based on the optimal conditions that are suggested by their corresponding original papers. A recommendation method with the lowest training/test time can be more scalable than others for the offline/online mode. It can be concluded from Tables 4-7 that user-based CF and MLCF are best and second best, respectively, in terms of the training time for Movielens, Epinions, LastFM and BookCrossing datasets, while item-based and user-based CF methods are the top-performers for Jester dataset (Table 8). Moreover, the experimental results show that BPMF and NMF methods achieve the best test times in most cases.

VI. CONCLUSION

In this paper, we compared a number of well-known memory- and model-based Collaborative Filtering (CF) algorithms in terms of various evaluation metrics. Recommendation algorithms can have different objectives and addressing the recommendation task from different aspects. We considered a number of well-known and state-of-the-art CF algorithms and briefly discussed their advantages and drawbacks. The memory-based CF algorithms are usually faster at query time, while model-based CF algorithms are more scalable and have better performance for sparse data. However, the latter have expensive model building, which increases the time of training step. Therefore, there will be a trade-off between performance and scalability to address. We considered various evaluation metrics ranging from accuracy-based metrics, to ranking-based metrics, execution time, coverage, novelty and diversity metrics. We compared the RSs algorithms in terms of these evaluation metrics on Movielens, Epinions, LastFM, BookCrossing and Jester datasets which have been frequently used as the benchmarks in RSs. Our results showed that there is no a golden algorithm that outperforms others in terms of all (or even most) evaluation metrics. The top-performing algorithms are almost spread for different evaluation metrics, which indicates that the assessment criterion specific to the particular application should be seriously taken into account in choosing the appropriate recommendation algorithm. We also assessed the algorithms based on a unified evaluation metric, for which the probabilistic matrix factorization recommendation algorithm showed the best performance among others.

REFERENCES

[1] F. Zhang, T. Gong, V. E. Lee, G. Zhao, C. Rong, and G. Qu, "Fast algorithms to evaluate collaborative filtering recommender systems," *Knowl.-Based Syst.*, vol. 96, pp. 96–103, Mar. 2015.

[2] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Trans. Web*, vol. 5, no. 1, 2011, Art. no. 2.

[3] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *Proc. Mach. Learn. Data Mining Pattern Recognit.* Berlin, Germany: Springer, 2007, pp. 548–562.

[4] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[5] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 1–35.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. ACM 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[7] P. Winoto and T. Y. Tang, "The role of user mood in movie recommendations," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 6086–6092, 2010.

[8] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. Application to the movies domain," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10990–11000, 2012.

[9] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos, "MusicBox: Personalized music recommendation based on cubic analysis of social tags," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 2, pp. 407–412, Feb. 2010.

[10] S. K. Lee, Y. H. Cho, and S. H. Kim, "Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations," *Inf. Sci.*, vol. 180, no. 11, pp. 2142–2155, 2010.

[11] C. Porcel and E. Herrera-Viedma, "Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries," *Knowl.-Based Syst.*, vol. 23, no. 1, pp. 32–39, 2010.

[12] 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, 2012.

[13] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying collaborative filtering to Usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[14] H. J. Lee and S. J. Park, "MONERS: A news recommender for the mobile Web," *Expert Syst. Appl.*, vol. 32, no. 1, pp. 143–150, 2007.

[15] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, 2001.

[16] T. Tang and G. McCalla, "Smart recommendation for an evolving e-learning system: Architecture and experiment," *Int. J. e-Learn.*, vol. 4, no. 1, pp. 105–129, 2005.

[17] 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, 2009.

[18] Y. Wen and Y. Shui-Sheng, "A survey of collaborative filtering algorithm applied in E-commerce recommender system," *Comput. Technol. Develop.*, vol. 16, no. 9, pp. 70–72, 2006.

[19] 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, 2011.

[20] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[21] S. Ahmadian, M. Meghdadi, and M. Afsharchi, "A social recommendation method based on an adaptive neighbor selection mechanism," *Inf. Process. Manage.*, vol. 54, no. 4, pp. 707–725, 2018.

[22] S. Ahmadian, M. Meghdadi, and M. Afsharchi, "Incorporating reliable virtual ratings into social recommendation systems," *Appl. Intell.*, vol. 48, no. 11, pp. 4448–4469, 2018.

[23] S. Ahmadian, M. Afsharchi, and M. Meghdadi, "An effective social recommendation method based on user reputation model and rating profile enhancement," *J. Inf. Sci.*, p. 0165551518808191, Oct. 2018.
[24] L. Luo, H. Xie, Y. Rao, and F. Lee Wang, "Personalized recommendation by matrix co-factorization with tags and time information," *Expert Syst. Appl.*, vol. 119, pp. 311–321, Apr. 2018.
[25] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.
[26] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Berlin, Germany: Springer, 2007, pp. 291–324.
[27] R. Van Meteren and M. Van Someren, "Using content-based filtering for recommendation," in *Proc. Mach. Learn. New Inf. Age, MLnet/ECML Workshop*, 2000, pp. 47–56.
[28] K. Lang, "Newsweeder: Learning to filter netnews," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 331–339.
[29] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 73–105.
[30] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowl.-Based Syst.*, vol. 157, pp. 1–9, Oct. 2018.
[31] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting Web sites," *Mach. Learn.*, vol. 27, no. 3, pp. 313–331, 1997.
[32] B. Krulwich, "Lifestyle finder: Intelligent user profiling using large-scale demographic data," *AI Mag.*, vol. 18, no. 2, p. 37, 1997.
[33] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artif. Intell. Rev.*, vol. 13, nos. 5–6, pp. 393–408, 1999.
[34] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.
[35] E. Bojnordi and P. Moradi, "A novel collaborative filtering model based on combination of correlation method with matrix completion technique," in *Proc. 16th CSI Int. Symp. Artif. Intell. Signal Process. (AISP)*, 2012, pp. 191–194.
[36] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Eng. Appl. Artif. Intell.*, vol. 46, pp. 58–66, Nov. 2015.
[37] B. Marlin, "Collaborative filtering: A machine learning perspective," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2004.
[38] I. M. Soboroff and C. K. Nicholas, "Related, but not relevant: Content-based collaborative filtering in TREC-8," *Inf. Retr.*, vol. 5, nos. 2–3, pp. 189–208, 2002.
[39] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Model. User-Adapted Interact.*, vol. 12, no. 4, pp. 331–370, 2002.
[40] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences," in *Proc. ISMIR*, vol. 6, 2006, p. 7.
[41] R. Xiong, J. Wang, N. Zhang, and Y. Ma, "Deep hybrid collaborative filtering for Web service recommendation," *Expert Syst. Appl.*, vol. 110, pp. 191–205, Nov. 2018.
[42] T. Miranda *et al.*, "Combining content-based and collaborative filters in an online newspaper," in *Proc. ACM SIGIR Workshop Recommender Syst.*, 1999.
[43] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 40, no. 3, pp. 66–72, 1997.
[44] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User Model. User-Adapted Interact.*, vol. 10, nos. 2–3, pp. 147–180, 2000.
[45] I. Soboroff and C. Nicholas, "Combining content and collaboration in text filtering," in *Proc. IJCAI*, vol. 99, 1999, pp. 86–91.
[46] E. Aslanian, M. Radmanesh, and M. Jalili, "Hybrid recommender systems based on content feature relationship," *IEEE Trans. Ind. Informat.*, to be published.
[47] M. M. Azadjalal, P. Moradi, A. Abdollahpouri, and M. Jalili, "A trust-aware recommendation method based on Pareto dominance and confidence concepts," *Knowl.-Based Syst.*, vol. 115, pp. 130–143, Jan. 2017.
[48] W. S. Lee, "Collaborative learning for recommender systems," in *Proc. ICML*, 2001, pp. 314–321.
[49] P. Moradi, S. Ahmadian, and F. Akhlaghian, "An effective trust-based recommendation method using a novel graph clustering algorithm," *Phys. A, Statist. Mech. Appl.*, vol. 436, pp. 462–481, Oct. 2015.
[50] C.-J. Zhang and A. Zeng, "Behavior patterns of online users and the effect on information filtering," *Phys. A, Statist. Mech. Appl.*, vol. 391, no. 4, pp. 1822–1830, 2012.
[51] L. Chen and P. Pu, "Critiquing-based recommenders: Survey and emerging trends," *User Model. User-Adapted Interact.*, vol. 22, nos. 1–2, pp. 125–150, 2012.
[52] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proc. 5th ACM Conf. Digit. Libraries*, 2000, pp. 195–204.
[53] J. A. Konstan, J. Riedl, A. Borchers, and J. L. Herlocker, "Recommender systems: A GroupLens perspective," in *Proc. Recommender Syst., Workshop (AAAI)*, 1998, pp. 60–64.
[54] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," in *Proc. AAAI/IAAI*, 1998, pp. 714–720.
[55] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.
[56] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 217–253.
[57] X. Ren, M. Song, E. Haihong, and J. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing*, vol. 241, pp. 38–55, Jun. 2017.
[58] A. Abbasi, A. Javari, M. Jalili, and H. R. Rabiee, "Enhancing precision of Markov-based recommenders using location information," presented at the Adv. Comput., Commun. Inform., 2014.
[59] D. W. Oard and J. Kim, "Implicit feedback for recommender systems," in *Proc. AAAI Workshop Recommender Syst.*, 1998, pp. 81–83.
[60] M. Jalili and M. Perc, "Information cascades in complex networks," *J. Complex Netw.*, vol. 5, no. 5, pp. 665–693, 2017.
[61] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Phys. Rep.*, vol. 519, no. 1, pp. 1–49, 2012.
[62] F. H. del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach," *Expert Syst. Appl.*, vol. 35, no. 3, pp. 790–804, 2008.
[63] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 257–297.
[64] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 1998, pp. 43–52.
[65] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1986.
[66] W. Zeng, M.-S. Shang, Q.-M. Zhang, L. LLü, and T. Zhou, "Can dissimilar users contribute to accuracy and diversity of personalized recommendation?" *Int. J. Mod. Phys. C*, vol. 21, no. 10, pp. 1217–1227, 2010.
[67] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unified relevance models for rating prediction in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, 2008, Art. no. 16.
[68] M.-S. Shang, L. Lü, W. Zeng, Y.-C. Zhang, and T. Zhou, "Relevance is more significant than correlation: Information filtering on sparse data," *Europhys. Lett.*, vol. 88, no. 6, p. 68008, 2009.
[69] J. Bobadilla, A. Hernando, F. Ortega, and A. Gutiérrez, "Collaborative filtering based on significances," *Inf. Sci.*, vol. 185, no. 1, pp. 1–17, 2012.
[70] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 1994, pp. 175–186.
[71] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating 'word of mouth,'" in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.* Reading, MA, USA: Addison-Wesley, 1995, pp. 210–217.
[72] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.
[73] N. Good *et al.*, "Combining collaborative filtering with personal agents for better recommendations," in *Proc. AAAI/IAAI*, 1999, pp. 439–446.
[74] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 1998, pp. 345–354.
[75] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system—A case study," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. TR-00-043, 2000.

[76] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proc. 2nd ACM Conf. Electron. Commerce*, 2000, pp. 158–167.
[77] D. Liben-Nowelly and J. Kleinberg, "The link prediction problem for social networks," in *Proc. 12th Annu. ACM Int. Conf. Inf. Knowl. Manage.*, 2003, pp. 556–559.
[78] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Phys. A, Statist. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.
[79] A. Javari, J. Gharibshah, and M. Jalili, "Recommender systems based on collaborative filtering and resource allocation," *Social Netw. Anal. Mining*, vol. 4, no. 1, p. 234, Dec. 2014.
[80] L. Lü, T. Zhou, and Y.-C. Zhang, "Predicting missing links via local information," *Eur. Phys.*, vol. 71, no. 4, pp. 623–630, 2009.
[81] X.-S. He, M.-Y. Zhou, Z. Zhuo, Z.-Q. Fu, and J.-G. Liu, "Predicting online ratings based on the opinion spreading process," *Phys. A, Statist. Mech. Appl.*, vol. 436, pp. 658–664, Oct. 2015.
[82] N. Polatidis and C. K. Georgiadis, "A multi-level collaborative filtering method that improves recommendations," *Expert Syst. Appl.*, vol. 48, pp. 100–110, Apr. 2015.
[83] V. Formoso, D. Fernández, F. Cacheda, and V. Carneiro, "Using profile expansion techniques to alleviate the new user problem," *Inf. Process. Manage.*, vol. 49, no. 3, pp. 659–672, 2013.
[84] J. Ben Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Mining Knowl. Discovery*, vol. 5, pp. 115–153, Jan. 2001.
[85] G. Karypis, "Evaluation of item-based top-*N* recommendation algorithms," in *Proc. ACM 10th Int. Conf. Inf. Knowl. Manage.*, 2001, pp. 247–254.
[86] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Discovery and evaluation of aggregate usage profiles for Web personalization," *Data Mining Knowl. Discovery*, vol. 6, no. 1, pp. 61–82, 2002.
[87] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *Proc. AAAI Workshop Recommendation Syst.*, 1998, pp. 1–16.
[88] M. Ramezani, P. Moradi, and F. Akhlaghian, "A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains," *Phys. A, Statist. Mech. Appl.*, vol. 408, pp. 72–84, Aug. 2014.
[89] T. Hofmann, "Collaborative filtering via Gaussian probabilistic latent semantic analysis," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Informaion Retr.*, 2003, pp. 259–266.
[90] K. Yu, X. Xu, J. Tao, M. Ester, and H.-P. Kriegel, "Instance selection techniques for memory-based collaborative filtering," in *Proc. SDM*, vol. 2, 2002, p. 16.
[91] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proc. SIAM Int. Conf. Data Mining* , vol. 5, 2005, pp. 471–480.
[92] D. Z. Navgaran, P. Moradi, and F. Akhlaghian, "Evolutionary based matrix factorization method for collaborative filtering systems," in *Proc. 21st Iranian Conf. Electr. Eng. (ICEE)*, 2013, pp. 1–5.
[93] X. Yuan, L. Han, S. Qian, G. Xu, and H. Yan, "Singular value decomposition based recommendation using imputed data," *Knowl.-Based Syst.*, vol. 163, pp. 485–494, Jan. 2018.
[94] T. V. R. Himabindu, V. Padmanabhan, and A. K. Pujari, "Conformal matrix factorization based recommender system," *Inf. Sci.*, vol. 467, pp. 685–707, Apr. 2018.
[95] K. Miyahara and M. J. Pazzani, "Improvement of collaborative filtering with the simple Bayesian classifier," *Inf. Process. Soc. Jpn.*, vol. 43, no. 11, pp. 3429–3437, 2002.
[96] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-based recommendation system using Bayesian user's preference model in mobile devices," in *Ubiquitous Intelligence and Computing*. Berlin, Germany: Springer, 2007, pp. 1130–1139.
[97] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering," in *Proc. ACM SIGIR Workshop Recommender Syst.*, vol. 128. Berkeley, CA, USA: Univ. California, Berkeley, 1999, pp. 1–4.
[98] S. H. S. Chee, J. Han, and K. Wang, "RecTree: An efficient collaborative filtering method," in *Data Warehousing and Knowledge Discovery*. Berlin, Germany: Springer, 2001, pp. 141–151.
[99] 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)*, Barcelona, Spain, Aug. 2018, pp. 1139–1144.
[100] G. Noh, H. Oh, and J. Lee, "Power users are not always powerful: The effect of social trust clusters in recommender systems," *Inf. Sci.*, vol. 462, pp. 1–15, Sep. 2018.
[101] L. Getoor and M. Sahami, "Using probabilistic relational models for collaborative filtering," in *Proc. Workshop Web Usage Anal. User Profiling (WEBKDD)*, 1999.
[102] H. Hong, B. J. Kim, M. Y. Choi, and H. Park, "Factors that predict better synchronizability on complex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, p. 067105, Jun. 2004.
[103] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, Jan. 2004.
[104] L. Huang, W. Tan, and Y. Sun, "Collaborative recommendation algorithm based on probabilistic matrix factorization in probabilistic latent semantic analysis," in *Multimedia Tools and Applications*. New York, NY, USA: Springer, 2018.
[105] S. Vucetic and Z. Obradovic, "Collaborative filtering using a regression-based approach," *Knowl. Inf. Syst.*, vol. 7, no. 1, pp. 1–22, 2005.
[106] D. Y. Pavlov and D. M. Pennock, "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1441–1448.
[107] B. M. Marlin, "Modeling user rating profiles for collaborative filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 627–634.
[108] G. Shani, R. I. Brafman, and D. Heckerman, "An MDP-based recommender system," in *Proc. 18th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 2002, pp. 453–460.
[109] K. Honda, N. Sugiura, H. Ichihashi, and S. Araki, "Collaborative filtering using principal component analysis and fuzzy clustering," in *Web Intelligence: Research and Development*. Berlin, Germany: Springer, 2001, pp. 394–402.
[110] J. F. Canny, "Collaborative filtering with privacy," in *Proc. IEEE Symp. Secur. Privacy*, Feb. 2002, pp. 45–57.
[111] L. Terán and A. Meier, "A fuzzy recommender system for eElections," in *Electronic Government and the Information Systems Perspective*. Berlin, Germany: Springer, 2010, pp. 62–76.
[112] C. Porcel, A. G. López-Herrera, and E. Herrera-Viedma, "A recommender system for research resources based on fuzzy linguistic modeling," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5173–5183, 2009.
[113] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[114] D. Billsus and M. J. Pazzani, "Learning collaborative information filters," in *Proc. ICML*, vol. 98, 1998, pp. 46–54.
[115] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Rev.*, vol. 37, no. 4, pp. 573–595, 1995.
[116] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
[117] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Proc. 5th Int. Conf. Comput. Inf. Sci.*, 2002, pp. 27–28.
[118] D. DeCoste, "Collaborative prediction using ensembles of maximum margin matrix factorizations," in *Proc. ACM 23rd Int. Conf. Mach. Learn.*, 2006, pp. 249–256.
[119] M. Kurucz, A. A. Benczúr, and K. Csalogány, "Methods for large scale SVD with missing values," in *Proc. KDD Cup Workshop*, vol. 12, 2007, pp. 31–38.
[120] B. Webb. (2006). *Netflix Update: Try This at Home*. [Online]. Available: http://sifter.org/?simon/journal/20061211.html
[121] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "On the gravity recommendation system," in *Proc. KDD Cup Workshop*, 2007, pp. 1–6.
[122] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. 13th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2000, pp. 535–541.
[123] S. Vargas and P. Castells, "Rank and relevance in novelty and diversity metrics for recommender systems," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 109–116.
[124] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.
[125] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. ACM 25th Int. Conf. Mach. Learn.*, 2008, pp. 880–887.
[126] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with Gaussian processes," in *Proc. ACM 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 601–608.
[127] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[128] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q.-S. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, Mar. 2016.
[129] X. Luo, M. Zhou, S. Li, and M. Shang, "An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2011–2022, May 2018.
[130] X. Luo, M. C. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, "An efficient second-order approach to factorize sparse matrices in recommender systems," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 946–956, Aug. 2015.
[131] R. Xu, S. Wang, X. Zheng, and Y. Chen, "Distributed collaborative filtering with singular ratings for large scale recommendation," *J. Syst. Softw.*, vol. 95, pp. 231–241, Sep. 2014.
[132] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *J. Mach. Learn. Res.*, vol. 10, 2009, pp. 2935–2962.
[133] W. Wu, L. He, and J. Yang, "Evaluating recommender systems," in *Proc. IEEE ICDIM*, Aug. 2012, pp. 56–61.
[134] 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.
[135] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," in *Proc. ACM 3rd Int. Workshop Web Inf. Data Manage.*, 2001, pp. 9–15.
[136] S. M. McNee *et al.*, "On the recommending of citations for research papers," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 2002, pp. 116–125.
[137] 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.*, 1999, pp. 230–237.
[138] B. N. Miller, J. Riedl, and J. Konstan, "Experiences with GroupLens: Making usenet useful again," in *Proc. Usenix Winter Tech. Conf.*, 1997, pp. 219–231.
[139] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.* Reading, MA, USA: Addison-Wesley, 1995, pp. 194–201.
[140] C. Spearman, "The proof and measurement of association between two things," *Amer. J. Psychol.*, vol. 15, no. 1, pp. 72–101, 1904.
[141] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, pp. 81–93, Jun. 1938.
[142] Y. Y. Yao, "Measuring retrieval effectiveness based on user preference of documents," *J. Amer. Soc. Inf. Sci.*, vol. 46, no. 2, pp. 133–145, Mar. 1995.
[143] C. Cleverdon and M. Kean. (1966). *Factors Determining the Performance of Indexing Systems*. [Online]. Available: http://dspace.lib.cranfield.ac.uk/handle/1826/863
[144] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.
[145] A. Moffat and J. Zobel, "Rank-biased precision for measurement of retrieval effectiveness," *ACM Trans. Inf. Syst.*, vol. 27, no. 1, 2008, Art. no. 2.
[146] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 10, pp. 4511–4515, 2010.
[147] M. Izadi, A. Javari, and M. Jalili, "Unifying inconsistent evaluation metrics in recommender systems," presented at the ACM Int. Workshop Recommender Syst. Eval., Dimensions Design, 2014.
[148] A. Javari, M. Izadi, and M. Jalili, "Recommender systems for social networks analysis and mining: Precision versus diversity," in *Complex Systems and Networks: Dynamics, Controls and Applications*, J. Lü, X. Yu, G. Chen, and W. Yu, Eds. Berlin, Germany: Springer, 2016, pp. 423–438.
[149] T. Zhou, R.-Q. Su, R.-R. Liu, L.-L. Jiang, B.-H. Wang, and Y.-C. Zhang, "Accurate and diverse recommendations via eliminating redundant correlations," *New J. Phys.*, vol. 11, no. 12, p. 123008, 2009.
[150] T. Zhou, L.-L. Jiang, R.-Q. Su, and Y.-C. Zhang, "Effect of initial configuration on network-based recommendation," *Europhys. Lett.*, vol. 81, no. 5, p. 58004, 2008.
[151] A. Javari and M. Jalili, "Accurate and novel recommendations: An algorithm based on popularity forecasting," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 4, 2014, Art. no. 56.
[152] A. Javari and M. Jalili, "A probabilistic model to resolve diversity—Accuracy challenge of recommendation systems," *Knowl. Inf. Syst.*, vol. 44, no. 3, pp. 609–627, 2015.

MAHDI JALILI (M'09–SM'16) received the Ph.D. degree in computer and communications sciences from the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, in 2008. He was an Assistant Professor with the Sharif University of Technology, Tehran, Iran. He is currently a Senior Lecturer with the School of Engineering, RMIT University, Melbourne, VIC, Australia, and holds the Australian Research Council DECRA Fellowship and the RMIT Vice-Chancellor Research Fellowship. He is also an Associate Editor of complex adaptive systems modeling, complexity and mathematical problems in engineering. His current research interests include network science, dynamical systems, social networks analysis and mining, and human-brain functional connectivity analysis.

SAJAD AHMADIAN received the B.S. degree in computer engineering from Razi University, Kermanshah, Iran, in 2011, and the M.S. degree in computer engineering from the University of Kurdistan, Sanandaj, Iran, in 2014. He is currently pursuing the Ph.D. degree with the University of Zanjan, Zanjan, Iran. His research interests include recommender systems, social networks analysis, and machine learning.

MALIHEH IZADI received the B.Sc. degree from the University of Isfahan, Isafahan, Iran, in 2012, and the M.Sc. degree in information technology engineering from the Sharif University of Technology, Tehran, Iran, in 2014, where she is currently pursuing the Ph.D. degree in computer engineering. Her research interests include applied machine learning, natural language processing, and recommender systems.

PARHAM MORADI received the M.Sc. and Ph.D. degrees in computer science from the Amirkabir University of Technology in 2006 and 2011, respectively. He conducted a part of his Ph.D. research work at the Laboratory of Nonlinear Systems, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, form 2009 to 2010. He is currently an Associate Professor with the Department of Computer Engineering, University of Kurdistan, Sanandaj, Iran. His current research areas include machine learning, dimensionality reduction, social network analysis, and recommender systems.

MOSTAFA SALEHI received the Ph.D. degree in computer engineering from the Sharif University of Technology, Iran, in 2012. In 2013, he joined the University of Tehran as an Assistant Professor. His research interests include network science and multimedia networks.