*J*ournal of ***AI*** *and* ***D****ata* ***M****ining*
Vol 7, No 3, 2019, 393-402
DOI: 10.22044/JADM.2019.7026.1830

Ensemble-based Top-*k* Recommender System Considering Incomplete Data

M. Moradi and J. Hamidzadeh*

Faculty of computer engineering and information technology, Sadjad University of Technology, Mashhad, Iran.

Received 29 April 2018; Revised 31 December 2018; Accepted 07 April 2019
**Corresponding author: J_Hamidzadeh@sadjad.ac.ir (J. Hamidzadeh).*

Abstract
Recommender systems have been widely used in e-commerce applications. They are a sub-class of information filtering system used to either predict whether a user will prefer an item (prediction problem) or identify a set of *k* items that will be of user-interest (Top-*k* recommendation problem). Demanding sufficient ratings to make robust predictions and suggesting qualified recommendations are two significant challenges in recommender systems. However, the latter is far from satisfactory because human decisions are affected by environmental conditions, and they might change over time. In this paper, we introduce an innovative method to impute ratings to missed components of the rating matrix. We also design an ensemble-based method to obtain Top-*k* recommendations. In order to evaluate the performance of the proposed method, several experiments have been conducted based on 10-fold cross-validation over real-world datasets. The experimental results show that the proposed method is superior to the state-of-the-art competing methods regarding the applied evaluation metrics.

Keywords: *Top-k Recommender Systems; Incomplete Data; Ensemble Learning.*

1. Introduction

The subject of recommender system has been known as an effective method to find information that is more preferable. The recommender systems are information filtering systems that are used to either predict whether a user will prefer an item (prediction problem) or identify a set of *k* items that will be of user-interest (Top-*k* recommendation problem) [1].

Three main categories for the recommender systems include collaborative filtering, content-based, and hybrid methods. Among them, collaborative filtering-based methods result in the most efficient recommendations. The technique of collaborative filtering can be divided into two categories: model-based and memory-based (neighborhood-based) [2]. Model-based methods use a collection of ratings to learn a model, which is then used to make rating predictions. Neighborhood-based collaborative filtering algorithms are based upon the fact that similar-minded users display similar rating pattern, and similar items receive similar ratings. Similar items mean the items that are similar to the items that the user has already rated or purchased. One of the major challenges of neighborhood-based systems is the sparsity of ratings, whereas the unrated items are very less compared to the items in the system, predicting that the accurate user's preference become very difficult. There are several reasons for why having unrated items. One reason is that a new item or user has recently registered to a recommender system. Thus there are no adequate user/item interactions in the system to determine the future of both item-to-item or personalized user-history-based recommendations. Another reason is that the registered user intentionally has not rated the items.Top-*k* recommender system works by estimating a user's rating for further items, based on interests, and recommends items that have high predicted ratings. In the recent years, some studies have focused on Top-*k* recommender systems [3-7]. Appropriate diversity of suggested items is an important issue that should be taken attention. In other words, if all items in the list are very similar, the possibility that the users are not satisfied with items is high, and this leads to

decreasing the quality of the user's experience with the recommender system. On the other hand, if the recommended items are chosen from different types, there will be a higher chance that the user might like at least one of them.

The aim of the proposed method is to suggest a list of *k* preference items to the users. Also as the real-world datasets are sparse, a new method is designed to fill the missing ratings and then make the Top-*k* recommendations. To create the Top-*k* list that considers diversity and satisfies the user's needs, an ensemble-based method is designed. Ensemble methods are commonly used in the field of data classification to improve the robustness of learning algorithms, and it is a common way in the recommender system field. The logic behind utilizing ensemble methods in this area is that a recommender system uses users' collaborations to improve the quality of the recommendations. In this case, the users that are considered as experts can be considered as the ones whose decisions affecting the system. Hence, the system is faced with multi-experts (users), and applying one classifier may not lead to accurate results. Methods such as [8-11] are examples of using ensemble learning methods in the recommender system. So far, the collaborative filtering problem has been viewed as a generalization of classification [2]. Hence, the majority of studies in recommender systems field use data mining techniques [12]. Examples of these techniques include clustering, regression, Artificial Neural Network (ANN) [13], Singular Value Decomposition (SVD), matrix completion technique [14, 15], decision tree [16], and association rule [17-22]. Some mathematical theories such as Dempster-Shafer (DS) theory [23, 24], probability theory, belief function theory, rough set theory and fuzzy set theory [25, 26] have been developed for representing data imperfections and uncertainty. The DS theory provides a flexible method for modeling information without requiring a probability to be assigned to each element in a set [27].

We experimentally evaluated the proposed method on seven real-world different datasets arising in various applications based on 10-fold cross-validation. The proposed method achieved better results in comparison with the state-of-the-art-methods.

The rest of this paper is organized as what follows. In Section 2, we review the related works. Details of the proposed method are described in Section 3. In Section 4 the experimental results are discussed using benchmark datasets. Conclusions and the future works are presented in Section 5.

2. Related works

The Top-*k* recommendation, firstly proposed in [28], has recently attracted increased research interest because it generates a ranked list of results that are related to the user satisfaction [29]. Kang, Peng [30] have presented a matrix-completion based method for the Top-*k* recommendation problem. Their method recovers the user-item matrix by solving a rank minimization problem. To better approximate the rank, a non- convex function is applied. Xia, Li [31] have proposed a robust to the noise recommender system utilizing matrix-completion, which can recover the user-location matrix considering structural noise and provide recommendations solely based on check-in records. SLIM, proposed by Ning and Karypis [32], employs a sparse linear model in which the recommendation score for a new item can be calculated as an aggregation of other items. A sparse aggregation coefficient matrix *W* is learned to make the aggregation very fast. *W* is learned by solving an l_1- norm and l_2-norm regularized optimization problem to make the aggregation very fast. Kabbur and Karypis [33] have proposed a Top-*k* recommendation method that models the user preference as a combination of having the global preference and interest-specific preference. The proposed method uses a nonlinear model for predicting the recommendation score, which is used to perform the Top-*k* recommendation task. The recommendation score is computed as a sum of the scores from the components representing global preference and interest-specific preference. Besides, various models use learning to rank [34] techniques to optimize binary relevance ranking metrics. For example, the methods proposed in [35-37] compute near-optimal ranked lists concerning the Area Under the Curve (AUC), Average Precision (AP) [38], and Reciprocal Rank metrics [39].

3. Proposed method

In this section, we present the main contributions of the proposed method, as follow: (1) a smooth method to fill incomplete ratings; and (2) a novel method to suggest Top-*k* items. Now, we discuss the proposed method in detail.

3.1. Filling incomplete rating matrix

The following subsection mainly focuses on estimating the missing values of rating matrix considering uncertainty in the users' preferences. This step can be considered as a preprocessing phase. At the end of this stage, we get the complete rating matrix. Now, we will describe the pre-processing phase.

The purpose of this stage is to determine which class label should be placed instead of the missing value. Let a rating matrix $R = \{\boldsymbol{r}_1, \dots, \boldsymbol{r}_M\}$ where M is the number of recommender system users. Each element $\boldsymbol{r}_i$ $(i = 1 \dots M)$ represents a vector $\boldsymbol{r}_i = [r_{i1}, \dots, r_{iN}]^T$, where N is the number of items and r_{ij} denotes the rating that user U_i assigns to item I_j. The user rates an item via assigning the preference label which is selected from a finite rank-ordered set $\Theta_{\text{pref}} = \{\theta_1, \theta_2, \dots, \theta_L\}$. The set Θ_{pref} contains the class labels. The vector $\boldsymbol{r}_i$ can be considered with some missing dimensions (unrated items). It means that user U_i has not rated item I_j. To fill the missing values with class labels, Eq. (1) is driven considering the affecting factors such as: the number of specific class samples and the distance between incomplete vector (test sample) and samples of other classes. To reduce the effect of uncertain ratings, the coefficient $\tilde{\mu}$ is added.

$$g_{\theta_l}\left(r_{ij}\right) = \frac{\tilde{\mu}_l \sum_{j=1}^{n_{\theta_l}} \left\| \boldsymbol{r}_i - \boldsymbol{r}_j \right\|^2}{n_{\theta_l}} \tag{1}$$

where n_{θ_l} is the number of samples in class θ_l ($\theta_l \in \Theta_{pref}$), and $\tilde{\mu}_l$ is a parameter that indicates the normalized number of $\boldsymbol{r}_i$'s adjacent samples for a specific class in the predefined adjacency radius. Its calculation is shown as an algorithm in

Table 1. It should be noted that in all equations, the missing values are ignored, and the known values in the same dimension of each sample are used.

Table 1. Procedure for $\tilde{\mu}_l$ calculation.

1.	Set an adjacency radius per test sample to distinguish the adjacent samples. The adjacency radius is a coefficient of class $\theta_l$bandwidth.
2.	Count the number of samples belonging to class θ_l.
3.	Compute $\tilde{\mu}$: divide the obtained value from step 2 by the total number of samples placed in adjacency radius.

To separate the samples linearly, they are transformed to a high-dimensional space by mapping function φ. Using the kernel function presented in Eq. (2), the inner products between the images of samples can be substituted in feature space.

$$K(\boldsymbol{r}_i, \boldsymbol{r}_j) = <\varphi(\boldsymbol{r}_i), \varphi(\boldsymbol{r}_j)> = \varphi(\boldsymbol{r}_i)^T \varphi(\boldsymbol{r}_j) \tag{2}$$

By substitution of Eq. (2) in Eq. (1), Eq. (3) can be driven.

$$\begin{aligned} g_{\theta_l}\left(r_{ij}\right) &= \frac{\tilde{\mu}_l \sum_{j=1}^{n_{\theta_l}} \left\| \boldsymbol{r}_i - \boldsymbol{r}_j \right\|^2}{n_{\theta_l}} = \frac{\tilde{\mu}_l \sum_{j=1}^{n_{\theta_l}} \left[\left(\phi(\boldsymbol{r}_i) - \phi(\boldsymbol{r}_j)\right)^T \left(\phi(\boldsymbol{r}_i) - \phi(\boldsymbol{r}_j)\right) \right]}{n_{\theta_l}} \\ &= \frac{\tilde{\mu}_l \sum_{j=1}^{n_{\theta_l}} [\mathrm{K}(\boldsymbol{r}_i, \boldsymbol{r}_i) - 2\mathrm{K}(\boldsymbol{r}_i, \boldsymbol{r}_j) + \mathrm{K}(\boldsymbol{r}_j, \boldsymbol{r}_j)]}{n_{\theta_l}} \end{aligned} \tag{3}$$

Using the radial basis kernel as shown in Eq. (4), a discriminate function can be given as in Eq. (5).

$$K(\boldsymbol{r}_i, \boldsymbol{r}_j) = \mathrm{e}^{-\frac{\left\| \boldsymbol{r}_i - \boldsymbol{r}_j \right\|^2}{2\gamma^2}} \tag{4}$$

$$g_{\theta_l}\left(r_{ij}\right) = \frac{2\sum_{j=1}^{n_{\theta_l}} \tilde{\mu}_l \left(1 - e^{\frac{-\left\| \boldsymbol{r}_i - \boldsymbol{r}_j \right\|^2}{2\gamma_{\theta_l}^2}} \right)}{n_{\theta_l}}, \; l = 1, \dots, L \tag{5}$$

where γ_{θ_l} is the bandwidth of each class. In the proposed method, two assumptions are considered: (1) each rate of rating matrix is considered as a sole-based classifier and (2) a rating matrix has a Gaussian (normal) distribution in which the Gaussian centers are the samples in a rating matrix. An important note is that these two assumptions are implemented for the first time in the present work. To calculate the bandwidth of each sample, a Parzen window [40] with Gaussian kernel is placed at each sample. Parzen window is a well-known non-parametric way to estimate the probability density function of a random variable. Given n_{tot} samples $r_j,\ j = 1, 2, \dots, n_{tot}$ in the L-dimensional space, where $L = \left|\Theta_{pref}\right|$, which follows an unknown distribution. Their pdf can be estimated using the expansion

$$p(r_i) \approx \frac{1}{n_{tot}} \sum_{j=1}^{n_{tot}} \frac{1}{(2\pi)^{L/2} h^L} \exp\left(-\frac{(r_i - r_j)^T (r_i - r_j)}{2h^2} \right), \tag{6}$$

for sufficiently large n_{tot} and sufficiently small values of h. Note that, h is a user-defined parameter. Using the above equation, the bandwidth of each sample is set as γ and is calculated by the following formula.

$$\gamma_i = \beta \frac{\int_{x_i - \varepsilon}^{x_i + \varepsilon} p(r_i)\, dx}{\underset{i}{Max} \int_{x_i - \varepsilon}^{x_i + \varepsilon} p(r_i)\, dx}, \tag{7}$$

where β is an adaptable tuning parameter that should be defined by the user. Trivially, Eq. (7) is between zero and one. The lower γ shows that the number of samples near the specified center is low, or that sample is near boundary of the class(es). On the other side, the greater γ indicates that the center is surrounded by other centers or it is near the similar class(es). Briefly, in the sparse region, and

also near boundary of the class(es) γ is low, and in dense regions, or the neighborhood of the similar classes γ is great.

To obtain the bandwidth of each class, γ_{θ_l}, the average bandwidth of any sample belonging to class θ_l should be calculated by

$$\gamma_{\theta_l} = \frac{\sum_{i=1}^{n_{\theta_l}} \gamma_i}{n_{\theta_l}} \tag{8}$$

To impute the most appropriate values in r_i, $g_{\theta_l}(r)$ is computed for all classes θ_l ($\theta_l \epsilon \Theta_{\text{pref}}$). The final rating is computed by

$$r_{ij} = \arg\min_{\theta_l} g_{\theta_l}\left(r_{ij}\right) \tag{9}$$

According to the above equation, the missing rating will be assigned to the class that has the minimum value of $g_{\theta_l}(r)$. If the computed results for $g_{\theta_l}(r)$ are equal to each other, one class is chosen randomly. As mentioned earlier, a rating $\theta_l, \theta_l \epsilon \Theta_{\text{pref}}$ in recommender systems has been considered as a class label. At the end of this stage, the complete vectors are obtained.

3.2. Predicting Top-*k* items

At this stage, a new method is proposed to predict the most preference *k* items for the users. The Top-*k* recommendation problem is treated as a ranking one. As mentioned earlier, the collaborative filtering problem can be viewed as a generalization of classification, where the prediction problem is the classification of a specific item from a huge amount of items *I* based on a user *U*. A classification result can be drawn by *P* (the output of the classifier) using any standard classifier. In the proposed method, the Softmax classifier is chosen and presented in Eq. (10).

$$\begin{aligned} &P(I = \Theta_{\text{pref}} \mid U) = \frac{e^{I_{\Theta_{\text{pref}_u}}} \times \text{p}(\Theta_{\text{pref}})}{\sum_{j \in \Theta_{\text{pref}}} e^{I_{j_u}} \times \text{p}(\Theta_{\text{pref}})}; \\ &u \in U = \{u_1, ..., u_{\text{M}}\}, I = \{I_1, ..., I_N\}, \\ &\Theta_{\text{pref}} = \{\theta_1, \ldots, \theta_L\} \end{aligned} \tag{10}$$

where $u \in \mathbb{R}^N$ represents the user and $I \in \mathbb{R}^N$ represents the specific item. Note that, the set *I* contains the items already items by the other users. To predict the top *k* items ($k \leq N$) and also to identify the user's preference for a specific item, the Dempster-Shafer combination [41] is used. For each item, DP is considered as an $M \times L$ matrix, where *M* is the number of users (classifiers), and *L* is the number of classes ($L = |\Theta_{pref}|$, $\Theta_{\text{pref}} = \{\theta_1, \theta_2, \ldots, \theta_L\}$). The Softmax classifier outputs that have been computed from Eq. (10) are elements of DP. The decision profile is constructed as follows:

$$DP(r) = \begin{bmatrix} p_{1,1}(r) \ldots p_{1,L}(r) \\ \ldots \\ p_{M,1}(r) \ldots p_{M,L}(r) \end{bmatrix}. \tag{11}$$

About the generated DP matrices, the *L* decision template (DT) can be constructed as follows:

$$DT_i = \frac{1}{card\{\theta_l\}} \sum_{r \in \theta_l} DP(r), \text{for } i = 1, 2, \ldots, L \tag{12}$$

To calculate the proximity Φ between DT_j^i (the ith row of the decision template for class j) and DP for every class $j = 1, \ldots, \theta_l$, and for every classifier $i = 1, \ldots, M$, Eq. (13) is presented.

$$\Phi_{j,i}(r) = \frac{(1 + \left\| DT_j^{\ i} - DP_i(r) \right\|^2)^{-1}}{\sum_{k=1}^{L} (1 + \left\| DT_k^{\ i} - DP_i(r) \right\|^2)^{-1}} \tag{13}$$

Using Eq. (13), belief degrees for every class $j = 1, \ldots, \theta_l$, and for every classifier $i = 1, \ldots, M$ are calculated.

$$b_j(DP_i(r)) = \frac{\Phi_{j,i}(r) \prod_{k \neq j} (1 - \Phi_{k,i}(r))}{1 - \Phi_{j,i}(r)[1 - \prod_{k \neq j} (1 - \Phi_{k,i}(r))]} \tag{14}$$

Membership degree is determined by Eq. (15). Its strategy uses the product of the belief degree. Final DS label with membership degrees is defined as follows:

$$\mu_j(r) = \prod_{i=1}^{M} b_j\left(DP_i(r)\right), \text{for } j = 1, \ldots, L \tag{15}$$

The class that has the maximum value of the Dempster-Shafer membership degree is the rating of the item.

$$\theta_I = \arg\max_r \mu_j(r) \tag{16}$$

The list of items is then sorted by the items that have the greatest rating and returned to the user.

Briefly, the proposed method has tended to handle incomplete data in collaborative filtering systems. Also it introduces an ensemble-based method that uses a combination of interests of users to suggest diverse Top-*k* items that satisfy the user's needs.

4. Experimental results

In this section, the datasets used are first introduced concisely. Next, the experimental setup and evaluation metrics are discussed. Then the training and recommendation phase designs are evaluated. Finally, the performance of the method is tested using evaluation metrics. The experiments in this

section are conducted to evaluate the performance of the proposed method against other relevant methods [3, 30, 31, 42].

4.1. Dataset

To analyze the effectiveness of the proposed method, several experiments were conducted on Movielens 100K, MovieLens 1M, MovieLens 10M, MovieLens 20M, Ciao, Book-Crossing, and Jester. Table 2 contains the number of users, items, ratings, rating scale, and addresses of the mentioned datasets. Rating scale contains finite rank-order θ_{pref}, e.g. Movielens ratings take from $\theta_{pref} = \{1, 2, 3, 4, 5\}$ where one is the lowest and five is the highest ratings.

Table 2. Dataset characteristics.

Dataset name	# of users	# of items	#of ratings	Rating scale (θ_{pref})	Address
MovieLens 100K	943	1,682	100,000	1 to 5	http://grouplens.org
MovieLens 1M	6040	3,883	1,000,209	1 to 5	
MovieLens 10M	69,878	10,681	10,000,054	0.5 to 5	
MovieLens 20M	138,493	27,278	20,000,263	0.5 to 5	
Ciao	7,375	99,746	278,483	1 to 5	http://dvd.ciao.co.uk
Book-Crossing	278,858	271,379	1,149,780	1 to 10	http://www2.informatik.uni-freiburg.de/~cziegler/BX/
Jester	59,132	140	1,761,439	-10 to 10	http://www.ieor.berkeley.edu

4.2. Experimental setup

In this sub-section, we describe the experimental tools and the parameters. Simulations were run in the MATLAB R2013 on an Intel processor at 2.30 GHz with 4 GB of RAM.

In the Parzen window approach, to estimate densities, we fixed the size and the shape of region R. To find the best value of length side, we examined for $h = \{2^{-2}, 2^{-1}, 2^{0}, 2^{+1}, 2^{+2}\}$. The best h is a value where the lowest classification error occurs. In the experimental results, we infer that h depends on the dataset characterization.

β is another adjustable tuning parameter that should be defined by the user. Its value was set to 0.5.

To generate the size-*k* ranked list, *k* was equal to 10.

4.3. Evaluation of metrics

In the Top-*k* recommender systems, precision and recall were used for the accuracy and performance evaluations [43]. The computation of recall and precision proceeds as follows:

$$\text{recall} = \frac{\#hits}{|T|} \tag{17}$$

$$\text{precision}(k) = \frac{\#\ hits}{k.|T|} \tag{18}$$

where #*hits* is the number of items in the test set *T* that is also present in the Top-*k* recommended items returned for each user. To measure the deviation of recommendation from the user's specific value, MAE metric was calculated by Eq. (19).

$$MAE = \frac{1}{n}\sum_{u,i}|p_{u,i} - r_{u,i}| \tag{19}$$

where n denotes the cardinality of rating matrix of test data, $p_{u,i}$ is the predicted rating, and $r_{u,i}$ is the real rating of user u for item i.

Other measures that are used include Hit Rate (HR) and Average Reciprocal Hit Rank (ARHR) which are formulated in Eqs. (20) and (21).

$$HR = \frac{\#hits}{\#users} \tag{20}$$

$$ARHR = \frac{1}{\#users}\sum_{i=1}^{\#hits}\frac{1}{pos_i} \tag{21}$$

where $\#users$ is the total number of test users, $\#hits$ is the number of users whose item in the test set is present in the size-*k* recommendation list, and pos_i is the position of the test item in the ranked recommendation list for the i^{th} hit. Besides, the computational time was applied as another evaluation metric. Computational time was considered as the length of time required to perform a computational process.

4.4. Result of interpretation

To validate the proposed method, experiments were conducted over the real-world datasets taken from the addresses shown in Table 2. A 10-fold cross-validation was used to demonstrate the performance of the methods. For each run, each dataset is split into the training and test sets by randomly selecting one of the non-zero entries for each user to be part of the test set. Considering the training set, a ranked list of *k* items is generated for each user. The method is then evaluated by comparing the ranked list of recommended items with the item in the test set. Table 3 shows the comparisons of accuracy, precision, recall, and MAE for the proposed method and the competing recommender systems on Movielens 100K, MovieLens 1M, MovieLens 10M, MovieLens

20M, Ciao, Book-Crossing, and Jester, respectively. The average rank provides a reasonable comparison of the methods; the performance of each method is ranked. The ranks are reported in enclosed parenthesis. **Bold** values highlight the best results of the experiments, and the rank of each method is enclosed in the parenthesis.

As shown in table 3, the proposed method obtains the best results for Book-crossing dataset in all evaluation metrics. Expect for the MovieLens 1M and Jester datasets, the MAE metric of the proposed method is better than the others. Comparisons regarding the recall metric show that the proposed method has better results in all datasets, expect for MovieLens20M; in this dataset, it obtains the recall rank 2. Totally, the proposed method has the best average rank in all metrics. This achievement is due to the ensemble-based approach that has been designed.

To independently compare the performance of the competing methods, we use the Receiver Operating Characteristic (ROC) curve. To plot the ROC curve, the true positive rate against the false positive rate should be computed. **Error! Reference source not found.**Figure 1 shows the ROC curve graphically. These graphical representations help us understand the efficiencies of the methods. Recall that the ROC curve demonstrates the trade-off between true positive rate (TPR) and false positive rate (FPR) (see Table 4).

Any increase in TPR will be accompanied by a decrease in FPR. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate will be the test. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate is the test. The area under the ROC curve measures the accuracy. An area of 1 represents a perfect test; an area of 0.5 represents a worthless test [44].

Table 5 demonstrates the calculated AUC for each method. As shown, the proposed method performs better than all the other methods across all the datasets.

Comparison of the HR and ARHR metrics is presented in table 6. As one can see in this table, the ensemble-based proposed method leads to significant improvements in either HR or ARHR. Also another ensemble-based method, [42], achieves rank 2 in most cases.

As the previous tables show, the proposed method is successful in presenting the Top-*k* recommendation list, although this success is costly and time-consuming. The presented computational time of methods is calculated as the sum of training and testing time. In table 7, the average running times (across training sets) are reported. We can see that the proposed method is consistently slower than the other methods. We attribute it to ensemble-based approach, which makes it time-consuming.

5. Conclusions and future works

The recommender systems are important research fields in e-commerce. In this paper, we introduced an efficient method to identify a list of user's preference items (Top-*k* recommendation problem). In the predicting process of collaborative filtering, to two points should be paid attention. First, the need for sufficient ratings to make robust predictions and secondly, recommendation quality. However, the latter is far from satisfactory because human decisions are affected by the environmental conditions and they might change over the time.

In this paper, we aimed to overcome the data imperfection limitations. To further enhance the accuracy of Top-*k* recommender systems, we first filled the missing ratings and then made the Top-*k* recommendation list. We imputed proper ratings to the missing ones in an innovative way. Then, to suggest Top-*k* recommendation items, an ensemble classifier was designed, which operated with an acceptable error rate. Regarding the experimental results, the proposed method gained a better performance compared with the state-of-the-art method.

The experimental results on the real-world datasets demonstrate the capability of the method in producing reliable results.

In our future work, we intend to consider the behavioral patterns of the users in recommender systems and try to find online method enabling us to make flexible and reliable predictions that can address the cold start problem. Another work that we are interested in is to consider the concept drift in recommender systems.

Table 3. The evaluation metrics comparison.

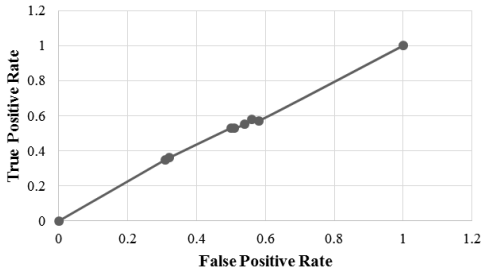
Dataset	Evaluation metrics	[3]	[42]	[30]	[31]	The proposed method
MovieLens 100K	Precision	0.59(2)	0.50(3)	**0.6(1)**	0.48(4)	0.50(3)
	Recall	0.53(4)	0.87(2)	0.46(5)	0.57(3)	**0.89(1)**
	MAE	0.48(3)	0.48(3)	0.59(4)	0.46(2)	**0.29(1)**
MovieLens 1M	Precision	0.51(4)	**0.7(1)**	0.51(4)	0.53(3)	0.55(2)
	Recall	0.36(5)	0.83(2)	0.44(4)	0.52(3)	**0.88(1)**
	MAE	0.55(4)	**0.20(1)**	0.58(5)	0.45(3)	0.28(2)
MovieLens 10M	Precision	0.52(3)	**0.54(1)**	0.51(4)	**0.54(1)**	0.53(2)
	Recall	0.35(5)	0.67(4)	0.42(3)	0.67(2)	**0.87(1)**
	MAE	0.57(4)	0.35(2)	0.57(4)	0.47(3)	**0.25(1)**
MovieLens 20M	Precision	0.57(2)	0.53(3)	0.53(3)	0.51(4)	**0.62(1)**
	Recall	0.53(4)	0.63(3)	0.40(5)	**0.76(1)**	0.73(2)
	MAE	0.53(5)	0.36(2)	0.52(4)	0.44(3)	**0.28(1)**
Ciao	Precision	0.59(1)	0.52(3)	0.51(4)	0.52(3)	0.55(2)
	Recall	0.57(3)	0.60(2)	0.40(5)	0.50(4)	**0.71(1)**
	MAE	0.47(3)	0.37(2)	0.53(4)	0.59(5)	**0.29(1)**
Book-Crossing	Precision	0.51(3)	0.55(2)	0.51(3)	0.56(2)	**0.71(1)**
	Recall	0.55(3)	0.59(2)	0.21(4)	0.55(3)	**0.62(1)**
	MAE	0.54(3)	0.25(2)	0.58(5)	0.55(4)	**0.18(1)**
Jester	Precision	0.47(5)	**0.67(1)**	0.53(2)	0.50(4)	0.52(3)
	Recall	0.58(3)	0.57(3)	0.41(5)	0.63(2)	**0.90(1)**
	MAE	0.53(5)	**0.24(1)**	0.52(4)	0.43(3)	0.30(2)
Average rank	Precision	2.86(2)	**2.00(1)**	3.00(3)	3.00(1)	**2.00(1)**
	Recall	3.86(3)	2.57(2)	4.43(4)	2.57(2)	**1.14(1)**
	MAE	3.86(4)	1.86(2)	4.29(5)	3.29(3)	**1.29(1)**

Table 4. Comparison of true positive and false positive rates.

Dataset	Rates	[3]	[42]	[30]	[31]	The proposed method
MovieLens 100K	TPR	0.53	0.87	0.46	0.57	0.89
	FPR	0.51	0.47	0.64	0.42	0.36
MovieLens 1M	TPR	0.36	0.83	0.44	0.52	0.88
	FPR	0.32	0.42	0.51	0.41	0.35
MovieLens 10M	TPR	0.35	0.67	0.42	0.67	0.87
	FPR	0.31	0.36	0.49	0.45	0.31
MovieLens 20M	TPR	0.53	0.63	0.40	0.76	0.73
	FPR	0.50	0.36	0.43	0.49	0.29
Ciao	TPR	0.57	0.60	0.40	0.50	0.71
	FPR	0.58	0.34	0.42	0.39	0.28
Book-Crossing	TPR	0.55	0.59	0.21	0.55	0.62
	FPR	0.54	0.21	0.35	0.40	0.10
Jester	TPR	0.58	0.57	0.41	0.63	0.90
	FPR	0.56	0.14	0.44	0.44	0.41

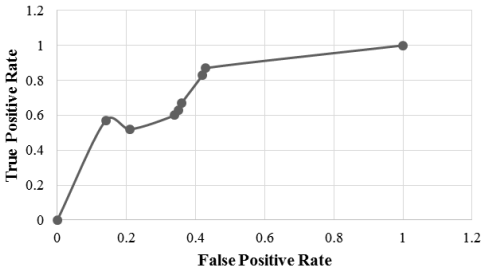
(a)

[3]



(b)

[42]



(c)

(d)

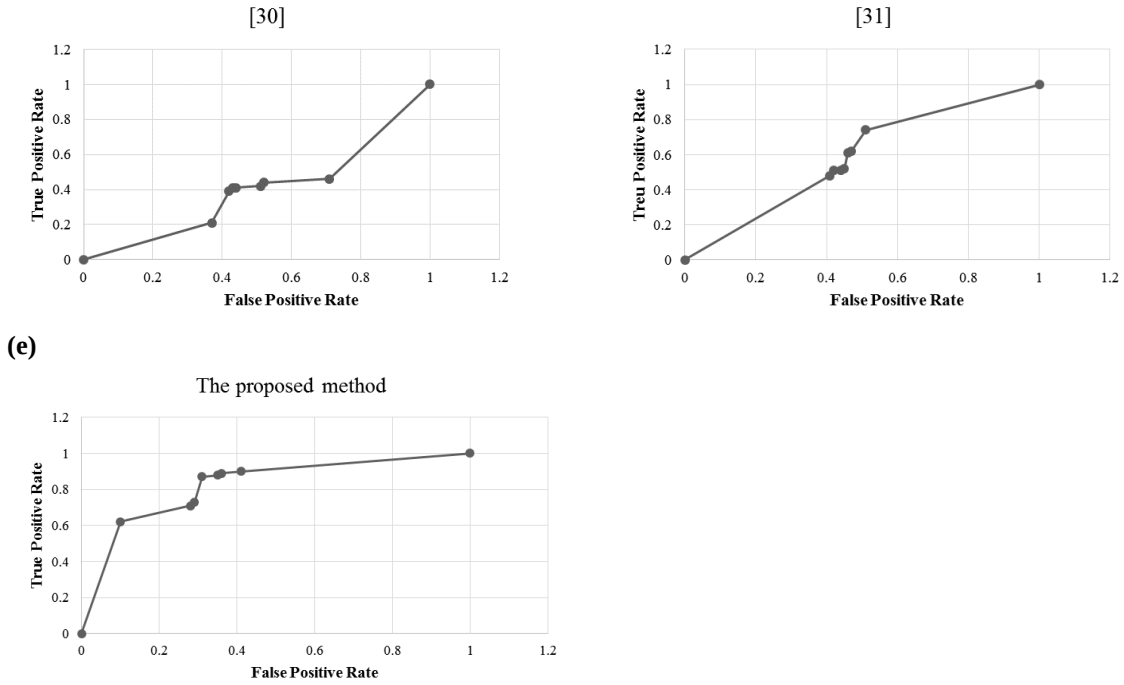


Figure 1. Comparison of ROC curve. (a) ROC curve of [3], (b) ROC curve of [42], (c) ROC curve of [30], (d) ROC curve of [31], (e) ROC curve of the proposed method.

Table 5. The AUC value.

	[3]	[42]	[30]	[31]	The proposed method
Area Under Curve (AUC)	0.6723	0.8357	0.5592	0.7711	**0.8936**

Table 6. Performance comparison of methods.

Dataset	Rates	[3]	[42]	[30]	[31]	The proposed method
MovieLens 100K	HR	0.128(5)	0.130(3)	0.129(4)	0.214(2)	**0.303(1)**
	ARHR	0.072(4)	0.089(2)	0.073(3)	0.035(5)	**0.183(1)**
MovieLens 1M	HR	0.382(3)	0.289(5)	0.381(4)	0.392(2)	**0.597(1)**
	ARHR	0.168(3)	0.132(4)	0.128(5)	0.175(2)	**0.212(1)**
MovieLens 10M	HR	0.247(4)	0.254(3)	0.221(5)	0.342(2)	**0.343(1)**
	ARHR	0.204(3)	0.154(5)	0.197(4)	0.215(2)	**0.304(1)**
MovieLens 20M	HR	0.138(5)	0.214(3)	0.158(4)	0.231(2)	**0.253(1)**
	ARHR	0.096(4)	0.128(2)	0.096(4)	0.122(3)	**0.134(1)**
Ciao	HR	0.165(5)	0.204(2)	0.187(4)	0.194(3)	**0.213(1)**
	ARHR	0.094(5)	0.115(3)	0.096(4)	0.119(2)	**0.121(1)**
Book-Crossing	HR	0.065(3)	0.064(4)	0.049(5)	0.075(2)	**0.089(1)**
	ARHR	0.024(4)	0.024(4)	0.032(2)	0.028(3)	**0.063(1)**
Jester	HR	0.365(4)	0.386(2)	0.328(5)	0.381(3)	**0.393(1)**
	ARHR	0.198(4)	0.232(2)	0.185(5)	0.229(3)	**0.247(1)**

Table 7. Comparison of computational time. The results are given in the format *hh:mm:ss*.

Dataset	[3]	[42]	[30]	[31]	The proposed method
MovieLens 100K	00:01:16	00:00:47	00:02:41	00:01:48	00:02:07
MovieLens 1M	00:01:36	00:01:17	00:02:17	00:02:32	00:02:28
MovieLens 10M	00:05:10	00:02:56	00:03:48	00:04:08	00:05:35
MovieLens 20M	00:14:16	00:10:50	00:12:53	00:13:41	00:35:21
Ciao	00:06:23	00:03:23	00:06:14	00:06:38	00:07:19
Book-Crossing	00:47:19	00:22:43	00:52:23	00:58:37	01:09:26
Jester	00:11:03	00:15:06	00:46:45	00:48:11	00:55:01

References

[1] Linden, G, Smith, B & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, vol. 7, no. 1, pp. 76-80.

[2] Aggarwal, C. C. (2016). Recommender systems. Cham: Springer International Publishing.

[3] Qian Y., et al. (2019). EARS: Emotion-aware recommender system based on hybrid information fusion. Information Fusion, vol. 46, pp. 141-146.

[4] Liu, P., Zhang, L. & Gulla, J. A. (2019). Real-time social recommendation based on graph embedding and temporal context. International Journal of Human-Computer Studies, vol. 121, pp. 58-72.

[5] Yuan, X., et al. (2019). Singular value decomposition based recommendation using imputed data. Knowledge-Based Systems, vol. 163, pp. 485-494.

[6] Bharti, R. & Gupta, D. (2019). Recommending Top N Movies Using Content-Based Filtering and Collaborative Filtering with Hadoop and Hive Framework, in Recent Developments in Machine Learning and Data Analytics, Springer. pp. 109-118.

[7] Yin H., et al. (2018). Mobi-SAGE-RS: A sparse additive generative model-based mobile application recommender system. Knowledge-Based Systems, vol. 157, pp. 68-80.

[8] da Costa, A.F., Manzato, M.G., & Campello, R.J. (2019). Boosting collaborative filtering with an ensemble of co-trained recommenders. Expert Systems with Applications, vol. 115, pp. 427-441.

[9] Logesh R., et al. (2019). Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. Neural Computing and Applications, pp. 1-24.

[10] Zhang Y., et al. (2019). A cost-sensitive three-way combination technique for ensemble learning in sentiment classification. International Journal of Approximate Reasoning, vol. 105, pp. 85-97.

[11] Fadaei Noghani, F. & Moattar, M. (2017). Ensemble Classification and Extended Feature Selection for Credit Card Fraud Detection. Journal of AI and Data Mining, vol. 5, no. 2, pp. 235-243.

[12] Portugal, I., Alencar, P. & Cowan, D. (2017) The use of machine learning algorithms in recommender systems: a systematic review. Expert Systems with Applications, vol. 97, pp. 205-227.

[13] Paradarami, T. K., Bastian, N. D., & Wightman J.L. (2017). A hybrid recommender system using artificial neural networks. Expert Systems with Applications, vol. 83, pp. 300-313.

[14] Liu, C. L. & Wu, X. W. (2016). Large-scale recommender system with compact latent factor model. Expert Systems with Applications, vol. 64, pp. 467-475.

[15] Rafailidis D. (2018). A Multi-latent Transition Model for Evolving Preferences in Recommender Systems. Expert Systems with Applications,

[16] Hamidzadeh J. (2015). IRDDS: Instance reduction based on Distance-based decision surface. Journal of AI and Data Mining, vol. 3, no. 2, pp. 121-130.

[17] Sadeghi, R. & Hamidzadeh, J. (2016). Automatic support vector data description. Soft Computing, vol. 22, no. 1, pp. 147-158.

[18] Hamidzadeh J. & Moradi, M. (2018). Improved one-class classification using filled function. Applied Intelligence, pp.1-17.

[19] Yuan, Y., Luo X, & Shang, M.S. (2018). Effects of preprocessing and training biases in latent factor models for recommender systems. Neurocomputing, vol. 275, pp. 2019-2030.

[20] Viktoratos, I., Tsadiras, A. & Bassiliades, N. (2018). Combining community-based knowledge with association rule mining to alleviate the cold start problem in context-aware recommender systems. Expert Systems with Applications, vol. 101, pp. 78-90.

[21] Shaikhina T., et al. (2017). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. Biomedical Signal Processing and Control,

[22] Padil, K. H., Bakhary, N., & Hao, H. (2017). The use of a non-probabilistic artificial neural network to consider uncertainties in vibration-based-damage detection. Mechanical Systems and Signal Processing, vol. 83, pp. 194-209.

[23] Shafer G. (1976). A mathematical theory of evidence. Princeton university press Princeton. vol. 1.

[24] Hamidzadeh, J. & Namaei, N. (2018). Belief-based chaotic algorithm for support vector data description. Soft Computing, pp. 1-26.

[25] Hamidzadeh, J., Javadzadeh, R. & Najafzadeh, A. (2015). Fuzzy Rule Based Diagnostic System For Detecting The Lung Cancer Disease. Journal of Renewable Natural Resources Bhutan, vol. 3.1, pp. 147-157.

[26] Castro, J., Year, R., & Martínez, L. (2018). A fuzzy approach for natural noise management in group recommender systems. Expert Systems with Applications, vol. 94, pp. 237-249.

[27] Maseleno, A. & Hasan, M. M. (2013). Move prediction in start kicking of sepak takraw game using dempster-shafer theory. Proceedings - 2012 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012, pp. 376-381.

[28] Wu C.W., et al. (2012). Mining top-k high utility itemsets. In Third IEEE international conference on data mining, pp. 19-26. IEEE, 2003.

[29] Cremonesi, P., Koren, Y. & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. fourth ACM conference on Recommender systems. ACM.

[30] Kang, Z., Peng, C. & Cheng, Q. (2016). Top-N Recommender System via Matrix Completion. Thirtieth AAAI Conference on Artificial Intelligence. 2016.

[31] Xia B., et al. (2018). Noise-tolerance matrix completion for location recommendation. Data Mining and Knowledge Discovery, vol. 32, no. 1, pp. 1-24.

[32] Ning, X. & Karypis, G. (2011). Slim: Sparse linear methods for top-n recommender systems. in Data Mining (ICDM), 2011 IEEE 11th International Conference on IEEE.

[33] Kabbur, S. & Karypis, G. (2014). Nlmf: Nonlinear matrix factorization methods for top-n recommender systems. IEEE International Conference on Data Mining Workshop. IEEE, 2014.

[34] Liu T.Y. (2009). Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval, vol. 3, no, 3, pp. 225-331.

[35] Shi Y., et al. (2012). TFMAP: optimizing MAP for top-n context-aware recommendation. 35th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2012.

[36] Rendle S., et al. (2009). BPR: Bayesian personalized ranking from implicit feedback. twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press.

[37] Shi Y., et al. (2012). CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. sixth ACM conference on Recommender systems. ACM.

[38] Manning, C.D., Raghavan, P. & Schütze, H. (2008). Introduction to information retrieval. vol. 1, Cambridge university press Cambridge.

[39] Voorhees, E. M. (1999). The TREC-8 Question Answering Track Report. in Trec.

[40] Parzen, E. (1962). On estimation of a probability density function and mode. The annals of mathematical statistics, vol. 33, no. 3, pp. 1065-1076.

[41] Kuncheva, L., Bezdek, J.C. & Duin, R.P.W. (2001). Decision template for multiple classifer fusion: An experimental comparison. Pattern Recogn Lett, vol. 34: pp. 228-37.

[42] Ben-Shimon, D., Rokach, L. & Shapira, B. (2016). An ensemble method for top-N recommendations from the SVD. Expert Systems with Applications, vol. 64, pp. 84-92.

[43] Ricci, F., Rokach, L. & Shapira, B. (2011). Introduction to recommender systems handbook, in Recommender systems handbook., Springer. pp. 1-35.

[44] Magee, P. & Tooley, M. (2011). The Physics, clinical measurement and equipment of anaesthetic practice for the FRCA. Oxford University Press.

نشریه هوش مصنوعی و داده کاوی

سیستم پیشنهاد دهنده ترکیبی Top-k با درنظر گرفتن داده ناکامل

منا مرادی و جواد حمیدزاده*

دانشکده مهندسی و فن آوری اطلاعات، دانشگاه صنعتی سجاد، مشهد، خراسان رضوی، ایران.

ارسال ۲۰۱۸/۰۴/۲۹؛ بازنگری ۲۰۱۸/۱۲/۳۱؛ پذیرش ۲۰۱۹/۰۴/۰۷

چکیده:

سیستم‌های پیشنهاددهنده به‌طور گسترده در تجارت الکترونیک مورداستفاده قرارگرفته‌اند. این سیستم‌ها زیرسیستمی از سیستم فیلترینگ اطلاعات هستند که برای پیش‌بینی اینکه کاربر یک کالا یا مجموعه‌ای از k-کالا را ترجیح می‌دهد مورداستفاده قرار می‌گیرند. در این سیستم‌ها، نیاز به امتیازات کافی برای پیش‌بینی‌های دقیق و ارائه پیشنهادهای مناسب، دو چالش پیش رو هستند زیرا تصمیم‌های انسانی تحت شرایط محیطی و در طول زمان تغییر می‌کند. در این مقاله، روشی نوآورانه برای پیش‌بینی مقادیر نامعلوم جدول امتیازات و همچنین روشی ترکیبی برای به دست آوردن لیست Top-k پیشنهادها ارائه شده است. به‌منظور ارزیابی روش پیشنهادی، آزمایش‌ها متعددی بر روی مجموعه داده‌های دنیای واقعی و به روش 10-fold cross-validation انجام شده است. نتایج نشان‌دهنده برتری روش پیشنهادی نسبت به روش‌های مقایسه شده است.

کلمات کلیدی: سیستم‌های پیشنهاددهنده Top-k، داده ناکامل، یادگیری ترکیبی.