Expert Systems With Applications 188 (2022) 116036

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Attention-based dynamic user modeling and Deep Collaborative filtering recommendation

Ruiqin Wang [a,c], Zongda Wu [b], Jungang Lou [a,c], Yunliang Jiang [a,c,*]

[a] School of Information Engineering, Huzhou University, Huzhou, China
[b] School of Mechanical and Electrical Engineering, Shaoxing University, Shaoxing, China
[c] Zhejiang Province Key Laboratory of Smart Management & Application of Modern Agricultural Resource, Huzhou, China

ARTICLE INFO

Keywords:
Short-term preferences
Long-term preferences
Dynamic preference modeling
Matching score prediction
Time-aware attention

ABSTRACT

Deep learning (DL) techniques have been widely used in recommender systems for user modeling and matching function learning based on historical interaction matrix. However, existing DL-based recommendation methods usually perform static user preference modeling by using historical interacted items of the user. In this article, we present a time-aware deep CF framework which contains two stages: dynamic user preference modeling based on attention mechanism and matching score prediction based on DL. In the first stage, short-term user preferences are modeled by the time-aware attention mechanism that fully considered the predicted item, the recent interacted items and their interaction time. The resulting short-term preferences are combined with long-term preferences for dynamic user preference modeling. In the second stage, high-order user-item feature interactions are learned by two types of DL models, Deep Matrix Factorization (DMF) and Multiple-Layer Perception (MLP), and the feature interaction vectors of the two models are fused in the last layer of the model to predict the matching score. Extensive experiments on five datasets indicate that our method is superior to the existing time-aware and DL-based recommendation methods in top-k recommendations significantly and consistently.

1. Introduction

With the development of social networks service platforms (e.g., Facebook, YouTube and Twitter) and e-commerce platforms (e.g., Amazon, Netflix and Taobao), the problem of information overload becomes more and more serious, and recommender systems have been recognized as an effective method to solve this problem. Collaborative filtering (CF) recommendation is one of the most popular recommendation methods. It selects similar neighbors for users or items by measuring the correlation between them based on the historical interaction information, and make predictions based on the preferences of the neighbors (Zheng, Li, & Liao, 2010; Liu, Chen, & Cai, 2012).

There are two types of CF technology: memory-based and model-based. The Matrix factorization (MF) algorithm is the most widely used model-based CF method. It maps users and items into a common low-dimensional latent space and measures the user-item similarity by dot product (Hu, Zhao, Wang, & Lai, 2017; Zhao, Huang, Wang, & Huang, 2018; Wang, Wang, Jia, & Yin, 2018; Wang, Cheng, Jiang, & Lou, 2019). Effective learning of latent factors plays an important role in MF methods. However, traditional MF methods only have limited learning ability because they can only capture linear relationships between the user and the item. Moreover, traditional MF methods are limited by static user modeling, which uses historical interactions of the user to build a user preference model without considering the interaction time of each item. Therefore, the resulted preference model can only represent long-term preferences but cannot reflect the change of user interests over time.

Recently, researchers have started to use DL techniques to enhance the nonlinear representation learning capabilities in CF recommendations. DL models with different working principles, such as Boltzmann machines, stacked auto-encoders and Multi-Layer Perception (MLP), are used to learn feature representations between users and items (Li, Kawale, & Fu, 2015; Xue, Dai, Zhang, & Huang, 2017; Deng, Huang, Wang, Lai, & Yu, 2019). In addition to learning better user and item representation, DL technology also has powerful abilities for matching functions learning (He, Liao, Zhang, & Nie, 2017; Beutel et al., 2018). In matching functions learning, the high-order feature interactions are often captured by the DNN model. Although DNN endows the model

* Corresponding author at: Zhejiang Province Key Laboratory of Smart Management & Application of Modern Agricultural Resource, Huzhou, China.
E-mail addresses: wrq@zjhu.edu.cn (R. Wang), zongda1983@163.com (Z. Wu), ljg@zjhu.edu.cn (J. Lou), jyl@zjhu.edu.cn (Y. Jiang).

https://doi.org/10.1016/j.eswa.2021.116036
Received 28 March 2021; Received in revised form 26 August 2021; Accepted 3 October 2021
Available online 7 October 2021

with great flexibility and high capacity, it has limited ability to catch low-order interactions, which is the basic function of MF methods. This is why many recommendation models strive to incorporate DNN with MF. In addition, the input quality of a DNN model has a great influence on the training results of the model. Using an unoptimized input, the training process of a DNN model can easily fall into the local optimum. Existing DNN-based recommendation systems either neglect this issue or use pre-training to obtain initial parameters. However, the pre-training process will introduce too much training time to the model. To deal with the above limitations, we propose a two-stage recommendation model. In the first stage, the dynamic user preferences are modeled by an attention network. In the second stage, high-order feature interactions are captured by two DL models for matching score prediction.

Recently, temporal recommendation methods are proposed to model the sequence behaviors of users in user preference modeling. Markov chain (Zhang, Ni, Li, & Yang, 2018), recurrent neural network (Li, Ren, Chen, Ren, Lian, & Ma, 2017) and attention mechanism (Liu, Zeng, Mokhosi, & Zhang, 2018) have been widely in session-based recommendations. However, a simplified assumption made by most of the existing temporal recommendation methods is to treat the interaction history as an ordered sequence, regardless of the timestamp of each interaction. Only a small number of approaches (Wang et al., 2020; Li et al., 2020; Wang, Jiang, & Lou, 2021; Wang, Jiang, & Lou, 2021) explicitly model the interaction timestamps to learn the impact of different time intervals on user modeling. However, the existing time interval-aware approaches are usually used in session-based recommendations, not in CF recommendation tasks, which is the research task of this article. To deal with the above limitations, we present a time-aware attention model that takes full consideration of the predicted item, the recent interacted items and their interaction time to model dynamic user preferences. Specifically, we use a two-layer DNN to determine the importance of each historical interacted item for short-term preference modeling. The resulting short-term preferences are then combined with the long-term preferences to obtain an accurate user preference model.

In general, the main contributions of our work are summarized as follows:

- We proposed a novel time-aware attention mechanism, which take full consideration of the predicted item, the recently interacted items and their interaction time to model the dynamic user preferences.
- We proposed a two-stage CF recommendation model. In the first stage, the attention network is used to model dynamic user preferences. In the second stage, two DL models are used to model the high-order user-item feature interactions to make predictions.
- We performed extensive experiments on five real-world datasets of different sizes. Experimental results indicate the superiority of our model over the existing time-aware and DL-based recommendation methods in top-k recommendations.

2. Related work

User preference modeling and matching function learning are two key parts of CF recommendation systems. In this section, we introduce the application of attention mechanism and DL technology in CF recommendations, as well as time-aware recommendation methods, which are closely related to our work.

2.1. Attention-based CF recommendations

Recently, more and more researchers began to use the attention mechanism to model user preferences and achieved state-of-the-art performance. ACF (Chen et al., 2017) is the first multimedia CF recommendation model based on the attention network, which seamlessly incorporates the attention mechanism into the classic CF models. It uses two attention models to capture the importance of different interacted items and different components of the item for user preference modeling. Attentional Factorization Machine (AFM) (Xiao, Ye, He, Zhang, Wu, & Chua, 2017) improves Factorization Machine (FM) by introducing different importance to different features of FM, and the importance is learned through an attention mechanism. NAIS (He et al., 2018) is an item-based CF in which the importance of different historical items in modeling user preferences is captured by the attention network. At the cost of only a few additional parameters, the attention network provides NAIS with more presentation capabilities than traditional item-based CF methods. Deep Match to Rank (DMR) (Lyu, Dong, Huo, & Ren, 2020) is a state-of-the-art attention-based CF method, which integrates matching task and ranking task into a unified model for Click Though-Rate (CTR) prediction. DMR includes two types of networks: user-to-item network and item-to-item network. In the user-to-item network, the relevance between the user and item is obtained by the inner product of their latent representations. In the item-to-item network, the similarities between the interacted items and the target item are captured by the attention network. Yan, Wang, Li, Zheng, and Han (2021) propose a self-attention metapath-based model on heterogeneous information networks to model the relations between users and items for the rating prediction task. DNESA (Zhang et al., 2021) integrates the attention mechanism with network embedding, concentrating on task-related parts and avoiding noisy parts of the network. The proposed attention mechanism can capture the evolving character of the network and learn the node embedding dynamically.

2.2. DL-based CF recommendations

The great success of DL in machine translation, speech recognition and image identification has triggered an extensive theoretical analysis of its representative power. Studies (Andoni et al., 2014; Veit et al., 2016) have shown that, with sufficient hidden layers and neural units, DNN can approximate any continuous function with arbitrary precision. In recent years, using DNN for recommendation becomes increasingly popular. DNN can not only learn non-trivial feature representations but also learn nonlinear activation functions. Existing DNN-based recommendation methods include the following two types of application: 1) use DNN to extract abstract feature representations of users and items from side information (i.e., acoustic features of music, textual descriptions of items, cross-domain user behaviors and item semantic information in the knowledge base); and 2) use DNN to learn the user-item feature interactions and the matching score(Wang, Jiang, & Lou, 2020).

In the task of representation learning, Den Oord et al. (2013) first proposed a deep content-based music recommendation model, which uses the deep Convolutional Neural Network (CNN) to learn audio signal representation for the recommendation. Wang and Wang (2014) propose an efficient hybrid method based on a deep belief network and a probabilistic graphical model, seamlessly unifying the feature learning and CF into an automated process. Collaborative Deep Learning (CDL) (Wang, Wang, & Yeung, 2015) is a hierarchical DL model that jointly performs deep representation learning based on content information and CF recommendation based on rating feedback matrix. Elkahky, Song, and He (2015) propose a cross-domain recommendation approach. It maps users and items to a common latent space and then performs joint representation learning from user features and item features in different domains based on a novel multi-view DL model. Collaborative knowledge base embedding (CKE) (Zhang, Yuan, Lian, Xie, & Ma, 2016) is an integrated representation learning model based on heterogeneous network embedding. It uses two kinds of DL models, stacked convolutional auto-encoders and stacked denoising auto-encoders, to jointly learn the semantic representations of items and the latent preferences of users for CF recommendation. DLCRS (Aljunid and Manjaiah, 2020) uses the linear interaction of user-item vectors as the input of the DL framework for the matching score prediction, which is completely different from the traditional MF and DL methods. The MF model uses the dot product of user-item vectors to model the matching

score. The DL model uses the connection of the user and item vectors as the input of the DL framework for matching score prediction.

In matching function learning, the Neural Collaborative Filtering (NCF) (He et al., 2017) uses the concatenation of the user embedding and the item embedding as the input of a DNN model to predict the user-item matching score. The NFM (He & Chua, 2017) adds a bi-interaction pooling layer to the classical factorization machine to obtain low-dimensional feature interaction vectors and has achieved great success. However, this method requires side information in feature representation learning, which is not easy to obtain in practical applications. NeuMF (He et al., 2017) replaces the dot product used in MF methods with MLP to learn the high-order matching functions. Although MLP has high capacity and nonlinear characteristics, it cannot model low-order feature interactions. Therefore, NeuMF unifies MF and MLP by concatenating their prediction vectors at the last layer to make linear and nonlinear matching function learning. Wang et al. (2018) propose a novel recommendation model, which combines the advantages of the embedding-based model and the tree-based model to make the recommendation results explainable. ConvNCF (Du et al., 2019) replaces the concatenation operation in NeuMF with an outer product operation, making the model more powerful in learning the pairwise correlation between users and items.

Recently, many hybrid recommendation models with both cross-learning and DL capabilities have emerged. Wide&Deep (Cheng et al., 2016) jointly trains the linear model based on cross features and the nonlinear model based on DNN. However, the success of Wide&Deep lies in the correct selection of cross features, which is an exponential issue, and there is no clear and efficient solution. Deep & Cross network (DCN) (Wang, Fu, Fu, & Wang, 2017) is a recommendation model based on web-based automatic feature representation learning. DCN can efficiently capture effective feature interactions and learn high-order nonlinear interactions without exhaustive searching and manual feature engineering, and the calculation cost is low. DNNRec (Kiran, Kumar, & Bhasker, 2019) is a novel DL-based hybrid recommendation method. It uses embedding, side information and deep network for latent factor learning and matching function learning. The proposed solution in DNNRec has become the benchmark of existing DL-based CF methods on recommendation accuracy and time complexity. Deep collaborative filtering (DeepCF) (Deng et al., 2019) combines the strength of DMF and MLP for matching score learning to avoid the flaws of the two methods and strengthen their merits. Bobadilla, Alonso, and Hernando (2020) utilize the potential of the reliability concept to enhance the performance of the rating prediction and the top-k recommendation by adding prediction error to the deep learning layer. Specifically, they use a three-stage deep learning architecture to extract the non-linear relations between predictions, reliabilities, and accurate recommendations.

2.3. Time-aware recommender systems

Traditional recommender systems usually model long-term user preferences. However, users' preferences change over time, and items that interacted in the past cannot represent the current preferences of users. The time factor has been considered in the time-aware CF recommender systems in the past (Ding & Li, 2005; Campos, Díez, & Cantador, 2014), and it has been found to perform better than the item-based CF method. The practice of using exponential decay on old ratings and enabling gradual forgetting to emphasize more recent ratings has been found to be useful in CF recommendations (Vinagre & Jorge, 2012). TimeSVD++ (Koren, 2010) is a well-known time-aware recommender model that introduces time-variant bias for each user and item at each time step to improve SVD++, which is the champion algorithm of the Netflex Prize.

In most cases, time factor is considered in Session-Based Recommendations (SBR), where Markov chains, recurrent neural network and attention mechanism have been used to model the sequential behaviors of users in the current session. HSMM (Zhang et al., 2018) uses a hidden semi-Markov model to track the changes in user interests. It models the heterogeneity of user interests and focuses by capturing the duration of the user stays in different latent interest states. GRU4Rec (Hidasi & Karatzoglou, 2017) uses the RNN model to SBR tasks, where practical aspects of the task are considered, and the loss function is modified to make the model more powerful. NARM (Li et al., 2017) uses the RNN and the attention network to model users' general interests and main purpose, respectively. STAMP (Liu et al., 2018) uses the attention network to extract the general interests of the user. It uses the most recent visit behavior to model the current interests of the user. ATRank (Zhou et al., 2018) introduces the multi-head self-attention mechanism in Transformer for rating predictions, which can speed up the training speed and improve the prediction accuracy. MARank (Yu, Zhang, Liang, & Zhang, 2019) is a multi-level attentive ranking model, which can unify individual- and union-level item interactions into preference modeling. Individual-level interaction highlights the transition order between item, and union-level interaction represents the relation between a single item and a group of items. SR-GNN (Wu, Tang, Zhu, Wang, & Tan, 2019) builds a session graph based on the sequential information in the session and then models the item and session representation through the information propagation relationship between graph nodes. CoSAN (Luo, Zhao, Liu, Zhuang, & Sheng, 2020) adopts a collaborative self-attention mechanism for SBR tasks. It extracts the main purpose of the user in the current session by considering the related items in neighborhood sessions. Gan and Cui (2021) propose a sequence-based DL model for movie recommendation. It explores user movie interest space to reflect user similarities and greatly improves dynamic recommendation performance.

However, a simplified assumption made by most existing temporal recommendation approaches is to treat the interaction history as an ordered sequence, regardless of the time interval between each interaction; that is, they consider the relative time order instead of the specific interaction times of each interacted item. To overcome the above limitations, Song, Qin, Jiang, and Liao (2018) propose a temporal recommender model. It uses joint past-present matrix decomposition and CF with Markov property to learn the temporal trend of user interests in fine granularity. (Wang et al., 2019) propose a time-aware attention model, which uses absolute temporal signals to represent the periodic behaviors of users and uses relative time signals to represent the temporal relation between items. CalendarGNN (Wang et al., 2020) uses multiple attention mechanisms to model complex interactions with various periodicity patterns (hourly, weekly, and weekday) in user behavior. TiSASRec (Li et al., 2020) explicitly models the interaction timestamps to learn the impact of different time intervals on the next-item prediction. Chorus (Wang et al., 2020) considers item relations (relational items in history sequences) and temporal dynamics (elapsed time) to model the embedding of the target item in different sequence contexts. TLSAN (Zhang et al., 2021) is a long- and short-term preference modeling method based on a time-aware attention mechanism. It learns users' temporal taste by time position embedding and captures long- and short-term user preferences by feature-wise attention layers for next-item recommendations.

Compared with the existing DL-based and time-aware recommendation methods, our method is novel in the following aspects:

- Existing DL-based recommendation methods usually use the embedding of ID or implicit feedback information as the input of a DL model to make user preference modeling and matching function learning. Our method uses a two-stage recommendation strategy; that is, capture dynamic user preferences via an attention network in the first stage and making predictions based on the generated user preferences via DL frameworks in the second stage.
- Existing time-aware attention mechanisms are usually used in session-based recommendation tasks, and the method proposed in this article introduces a novel time-aware attention mechanism to

the CF context to improve the top-k recommendation performance of the traditional CF recommendation tasks.

- Different from the existing recommendation methods based on static user preference modeling, the proposed time-ware attention mechanism fully considers the predicted item, the recently interacted items and their interaction time for user preference modeling. It can capture the dynamic changes of users' interest at different times and in different contexts.
- In this article, the time embedding is learned through model training, which is completely different from existing methods. Existing methods usually use position embedding or time kernel function to model the temporal behaviors of the user. Position embedding ignores the timestamp of each interaction, and it is difficult to design a reasonable time kernel function.

3. Time-aware Attention-based Deep Collaborative filtering

We will first introduce the problem statement of our method, namely Time-aware Attention-based Deep Collaborative Filtering (TADCF). Then, the general framework of TADCF and its working principles will be presented. Next, the component modules of the framework and the hyperparameters learning method will be described in detail. Finally, we will discuss the relationship between our method and the state-of-the-art DL-based recommendation methods. The time complexity analysis and training algorithm of our method will also be included in this section. Table 1 lists commonly used symbols in this article.

3.1. Problem statement

Given the user-item interaction matrix R based on the implicit feedback information and interaction time matrix T of each interaction, the task of the TADCF is to predict the missing interactions in R through time-aware attention mechanisms and different types of DL models.

The main problem of the recommendation method based on implicit feedback is that there is no negative feedback. Unobserved interactions do not mean that users don't like the item. Because the number of items in the system is large, the user may have not seen the item at all. However, we assume that the user prefers an observed item to an unobserved one. Therefore, in this article, the model is trained by randomly sampling negative samples from unobserved items.

3.2. General framework

Figure 1 gives the general framework of the TADCF model. We can see from the figure that the proposed model is a two-stage recommendation model with implicit interaction matrix R and interaction time matrix T as inputs. The rows and columns of R are used to encode long-term preferences of users and embedding of items, respectively. The predicted item, the recently interacted items and their interaction time are used to capture the short-term user preferences through an attention network. Dynamic user preferences are modeled by combining the short- and long-term user preferences. Then, high-order feature interactions and the matching score of user-item pair are modeled by two different types of DL models, DMF and DNN. In the DMF part, two separate DNN models are used to learn the latent representations of users and items. The linear user-item feature interactions are then modeled by the inner product of the user and item latent representations. In the DNN part, the connection of dynamic user preference vector and item embedding vector is used as the input of the DNN model to learn nonlinear user-item feature interactions. Finally, the interaction vectors of the two models are fused in the last layer of the model to predict the user-item matching score, and all model hyperparameters are learned in a unified way.

Table 1
Summary of symbols used in this article.

Symbol	Description
$R \in \mathscr{R}^{m \times n}$	Interaction matrix
$T \in \mathscr{R}^{m \times n}$	Interaction time matrix
$e^v \in \mathscr{R}^{n \times d}$	Item embedding vector
$e^t \in \mathscr{R}^{t \times d}$	Time embedding vector
$P \in \mathscr{R}^{m \times d}$	Dynamic user preferences
$P^l \in \mathscr{R}^{m \times d}$	Long-term user preferences
$P^s \in \mathscr{R}^{m \times d}$	Short-term user preferences
$U \in \mathscr{R}^{m \times d}$	User latent factor
$V \in \mathscr{R}^{n \times d}$	Item latent factor
$\alpha \in \mathscr{R}^{m \times k}$	Weight of historical interacted items
β	fusion coefficient of short-term preferences
m	User number
n	Item number
k	The size of recently interacted items
d	The size of latent factors
t	The size of time interval

3.3. Embedding

For simplicity, we use the rows and the columns of the interaction matrix, namely the historical interaction vectors, as user preference vector and item feature vector, which are multi-hot binary vectors. As can be seen from the subsequent description, the user preference vector here actually represents the long-term preferences. Generally, many users and items and relatively small interactions often result in very sparse and high-dimensional representation vectors. To reduce the dimensionality, a linear embedding is used in the embedding layer to transform the high-dimensional one-hot vectors into low-dimensional real-value vectors as follows:

$$e_j^v = W_v v_j \tag{1}$$

where $v_j$is the implicit feedback vector of item j, corresponding to the jth column of R; W_v is the item encoding matrix.

Using the same embedding method, we can obtain the embedding vector of the period of the historical interacted items as follows:

$$e_j^t = W_t ts_j \tag{2}$$

where W_t is the time encoding matrix; ts_j is the time interval of item j, which is defined as follows:

$$ts_j = t_i - t_j \tag{3}$$

where t_j is the interaction timestamp of historical interacted item j and t_i is the interaction timestamp of the predicted item i. Both t_i and t_j come from the interaction time matrix T.

3.4. Dynamic user preference modeling

Accurate user preference modeling is the premise and foundation for a high-quality recommendation. A widely used method in recommendation systems is to predict the user's preference for the current item given the user's interaction history. Traditional user modeling methods usually take the entire interaction history of a user as his/her preferences profile, which completely ignores the interest drift of the user. Generally, users' interests and preferences change dynamically with time and context. Suppose a person likes comedy when he is young, but as he grows up, he begins to like drama. As another example, when a person is viewing computer hardware on an e-commerce website, we should not use his historical preferences on other domains such as music or sporting as a reference for predicting his interests in the current product. To overcome the above limitations, in this section, we will present a novel attention mechanism, which can capture the dynamic user preferences in different time and different contexts.

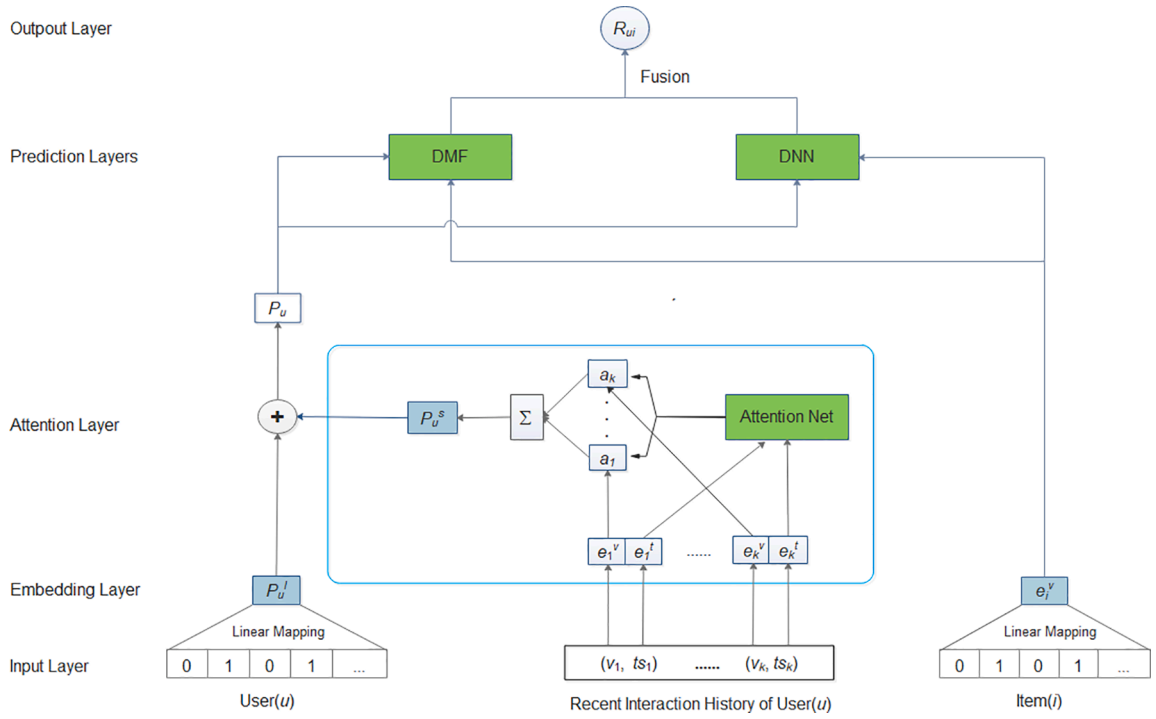

Fig. 1. Framework of the TADCF.

Unlike existing item-level attention methods that select representative items from the entire user interaction history, here only the recently interacted items are used to model the short-term user preference. Because the recently interacted items are more representative than the historical interacted items a long time ago, and most importantly, it can reduce the computation complexity of the user preference modeling process. In this article, we use the recent k historical interacted items, and the value of k is set by experiments. If the number of historical interactions of some users is less than k, we fill it with a globally meaningless constant to make the historical interaction vectors have the same length.

First, the three embedding vectors corresponding to the predicted item, the recent interacted item and its interaction time are connected and used as the input of a DNN model to learn the importance of each historical interacted item for short-term preference modeling. Here, a two-layer neural network model is used as the attention network as follows:

$$a(j) = W_2{}^T \varnothing\left(W_{11}e_i^v + W_{12}e_j^v + W_{13}e_j^t + b_1\right) + b_2 \tag{4}$$

where $a(j)$ is the resulting importance weight of the historical interacted item j; W_* and b_k are weight matrices and biases of the attention network, respectively; $\varnothing(x) = \max(0, x)$ is the ReLU activation function.

Then, the final attention weight of each recent interacted item is obtained by the Softmax normalization as follows:

$$a(j) = \frac{\exp(a(j))}{\sum_{l \in R_k(u)} \exp(a(l))} \tag{5}$$

where $R_k(u)$ is the recent k interacted items of user u.

Next, the weighted sum of embedding vectors of recently interacted items is used as the short-term user preferences, as shown below:

$$P_u^s = \sum_{j=1}^{k} a(j) \bullet \mathrm{e}_j^v \tag{6}$$

Finally, the dynamic user preference vector is modeled as the combination of the short- and long-term preferences as follows:

$$P_u = \beta \bullet P_u^s + (1-\beta) \bullet \mathrm{P}_u^l \tag{7}$$

where $\beta \in [0, 1]$ is the fusion coefficient. For simplicity, the long-term user preference is modeled as an arithmetic mean of the user's historical interacted items, as shown below:

$$P_u^l = \frac{1}{|R(u)|} \sum_{j \in R(u)} e_j^v \tag{8}$$

where $R(u)$ is the items that user u has interacted with in the past, and $|R(u)|$ is the number of items in $R(u)$.

3.5. DL-based feature interaction learning

Deep learning (DL) technology is very suitable for complex matching function learning because it can approximate any continuous function. The DNN is the most widely used DL method due to its simple principle and high capabilities. In this section, we adopt the DNN as the feature interaction learning and matching score prediction model. Specially, we propose a two-way DL framework based on two types of DL model, DMF and DNN. In the DMP part, two separate DNNs are used to extract the user and item latent representations, respectively. Then, the linear user-item feature interactions are obtained by the inner product of their latent representations. In the DNN part, the combination of the dynamic user preference vector and the item embedding vector is used as the input of the DNN to learn the non-linear feature interactions between them. The two interaction vectors are fused in the output layer of the TADCF for matching score prediction.

Given the dynamic user preference P_u and item embedding e_i^v, two separate DNNs are used to learn the latent presentation vectors of users and items as follows:

$$U_u = M_{l_1}(\varnothing(M_1 P_u + c_1)\cdots) + c_{l_1} \tag{9}$$

$$V_i = N_{l_2}\left(\varnothing\left(N_1 e_i^v + d_1\right) + \cdots\right) + d_{l_2} \tag{10}$$

where M_* and c_* are the weight matrices and biases of the user model, respectively; N_* and d_* are the weight matrices and biases of the item

model, respectively; l_1 and l_2 are the layer numbers of the user model and the item model, respectively.

After obtaining the latent representation vectors of the user and the item, the linear feature interactions between them can be obtained by the inner product as follows:

$$y_{ui}^{DMF} = (H_u U_u) \otimes (H_v V_i) \tag{11}$$

where $\otimes$ is the inner product between two vectors; H_u and H_v are mapping matrices, which are used to obtain user and item vectors with the same dimensions.

In addition to the above linear feature interaction modeling, another DNN model is also used to learn the non-linear high-order feature interactions as follows:

$$y_{ui}^{MLP} = L_l{}^T \sigma\left(\sigma\left(L_1\left[P_u, e_i^v\right] + g_1\right) + \cdots.\right) + g_l \tag{12}$$

where L_* and g_* are the weight matrices and biases of the model, $[P_u, e_i^v]$ is the vector concatenation of the dynamic user preference vector P_u and the item embedding vector e_i^v, l is the layer number of the model.

3.6. Fusion and learning

The previous section presented two types of DL models, DMF and MLP, used to learn linear and nonlinear feature interactions between the user and the item. Both models have their advantages and can learn the matching function from different perspectives. To maintain their respective advantages, we integrate the two models into a unified framework based on a seamless fusion strategy. The specific approach is as follows: concatenating the prediction vectors of the two DL models and feeding it into a full-connected layer to learn the user-item matching score. The prediction vector of the DMF model is the element-wise product of the user-item latent representations, and the prediction vector of the MLP model is the output vector at its last hidden layer.

Specifically, given the prediction vectors of the two kinds of matching function learning models, the final matching score of the integrated model is as follows:

$$\widehat{r}_{ui} = \sigma\left(W_{out}[y_{ui}^{DMF}, y_{ui}^{MLP}\right] \tag{13}$$

where W_{out} is the output weight matrix of the model, $\sigma(\bullet)$ is the Sigmoid activation function, and $[,]$ is the concatenation of two vectors. Combining of the two vectors can result in a better representation because both methods have their advantages and learn from different perspectives. In addition, the subsequent fully-connected layer gives the integrated model the ability to learn different weight for different latent factors.

There are two kinds of objective functions commonly used in recommendation systems: point- and pair-wise loss. In this work, we use only the point-wise loss function and leave the other in future work. Squared loss is the most commonly used method in point-wise loss, but it can only be used in explicit feedback scenario. Our model is a CF model based on implicit feedback information, so the binary cross-entropy loss function is used as the objective function as follows:

$$\mathscr{L}(\Theta) = \sum_{(u,i)\in R^+ \cup R^-} r_{ui} log \widehat{r}_{ui} + (1 - r_{ui})\left(1 - \log \widehat{r}_{ui}\right) \tag{14}$$

where R^+ and R^- denote positive and negative sample sets, respectively; r_{ui} is the real feedback $\widehat{r}_{ui}$ is the predicted matching score, Θ is the hyperparameter set of the model. The goal of the above objective function is to minimize the binary cross-entropy between the real feedback and the predicted matching score, which is equivalent to maximum the likelihood estimation.

In order to clearly describe the implementation process of the proposed model TADCF, Algorithm 1 shows the detailed training process of the model.

3.7. Relation to existing DL-based recommendation methods

The TADCF is an integrated recommendation model with dynamic user presentation modeling and linear and nonlinear feature interactions learning abilities. The state-of-the-art CF recommendation methods based on the DL model, Wide&Deep and DeepCF, have some similarities with the TADCF model.

The DNN component of the TADCF is similar to that of the Wide&Deep, and the main difference between them lies in the input vector. The input of the DNN model of the Wide&Deep is the original feature vectors, and the input of the DNN model of the TADCF is the dynamic user preference vector and the item embedding vector. The disadvantage of using the original feature vectors as the input is that the high dimensionality of the original features will increase the burden of hidden layers and affect the training speed of the model. Another disadvantage is that the original features may contain meaningless features or even noise, which will affect the prediction accuracy of the model.

The TADCF and the DeepCF use a DMF and an MLP as their components, and the TADCF is an improved version of the DeepCF. The main difference between the TADCF and the DeepCF lies in the user modeling strategy. In the DeepCF, the user preferences are represented as his/her entire past interaction history. Therefore, it is a static user modeling process. In the TADCF, we consider the predicted item, the recently interacted items and their interaction time to capture the user preferences. Therefore, the resulted user preferences are dynamic and can evolve automatically in different prediction time and in different context.

3.8. Time complexity analysis

Training time is an important factor need to be considered in recommendation systems to make the cost-effectiveness trade-off (Cunha, Mangaravite, Gomes, Canuto, & Gonalves, 2021). Our model TADCF is a two-stage CF recommendation model; that is, the stage of user preference modeling based on attention mechanism and the stage of feature interaction learning bases on DL. Both stages have DNN models, and the main cost comes from the hidden layers of the DNNs. The time complexity of the matrix–vector multiplication in the hidden layer is $O(d_{l-1}d_l)$, where d_l is the number of neurons in the lth hidden layer. In addition, in the embedding layer, the matrix–vector multiplication is also used to obtain the user and item embedding vectors, and its time complexity is $O(kd)$, where d is the dimension of the embedding vector, $d \ll k$ and $k \in \max(m, n)$. In the output layer, there is a full-connected layer to make final matching score prediction, and its time complexity is $O(d)$. Therefore, the overall training time of the TADCF model is $O\left(kd + \sum_{l=1}^{L} d_{l-1}d_l\right)$, which is similar to the existing DL-based CF recommendation models, NeuMF and DeepCF. Therefore, the high performance of our model is not obtained by introducing additional calculation complexity.

4. Experiments

To show the superiority of our method compared with the state-of-the-art DL-based and time-aware recommendation methods and the effectiveness of the key techniques proposed in this article, we will conduct extensive experiments to answer the following research questions:

- **RQ1.** Does our model TADCF achieve better performance than the state-of-the-art DL-based and time-aware recommendation methods?
- **RQ2.** Is the time-aware attention mechanism proposed in this article effective for dynamic user preference modeling?
- **RQ3.** How the hyperparameters of the TADCF model affect the top-k recommendation performance, and to what extent?

Algorithm 1: Training algorithm of TADCF.

Input: *loop*: # of training iterations,

ne: # of negative sampling ratio,

k: number of recent interactions

l_1, l_2, l: number of layers of different DNN models

β: coefficient of long-term preferences,

$R \in \mathscr{R}^{m \times n}$: interaction matrix,

$T \in \mathscr{R}^{m \times n}$: interaction time matrix.

Output: W_v: embedding matrix for item,

W_t: embedding matrix for interaction time,

$W_{k,}$: weight matrices of layer k of attention network,

b_k: bias of layer k of attention network,

M_k: weight matrices of user representation learning model,

c_k: biases of user representation learning model,

N_k: weight matrix of item representation learning model,

d_k: biases of item representation learning model,

H_u: mapping matrix of user latent vector,

H_v: mapping matrix of item latent vector,

L_k: weight matrix of matching score learning model,

g_k: biases of matching score learning model,

W_{out} : output weight matrix of the integrated model.

1: Initialization:

2: randomly initialize $\Theta, \Theta = \{W_u, W_t, W_k, M_k, N_k, H_u, H_v, L_k, W_{out}, b_k\ c_k, d_k$ and $g_k\}$

3: set $R^+ \leftarrow$ none zero elements in R;

4: set $R^{-\text{all}} \leftarrow$ zero elements in R;

5: set $R^- \leftarrow$ sampling $ne \times \|R^+\|$ elements from $R^{-\text{all}}$.

6: **for** l in (1…*loop*) **do**

7: **for** each element (u, i) of user u and item i in $R^+ \cup R^-$ **do**

8: set e_i^v based on Eq.1 with input of v_i;

9: **for** j from 1 to h **do**

10: set e_j^v based on Eq.1 with input of v_j;

11. set e_j^t by Eq.2 and 3 with input of t_i and t_j;

12: set a_j by Eq.4 and 5 with inputs of e_i^v, e_j^v and e_j^t;

13: **end for**

14: set P_u^s by Eq.6 with inputs of a_k and e_j^v;

15: set P_u^l by Eq.8 with input of $R(u)$;

16: set P_u by Eq.7 with inputs of P_u^s,P_u^l and β;

17: set U_u by Eq.8 with inputs of P_u;

18: set V_i by Eq.10 with inputs of e_i^v;

19: set y_{ui}^{DMF} by Eq.11 with input of U_u and V_i;

20: set y_{ui}^{MLP} by Eq.12 with input of U_u and V_i;

21: set $\widehat{r}_{ui}$ by Eq.13 with inputs ofy_{ui}^{DMF} and y_{ui}^{MLP};

22: set $\mathscr{L}(\Theta)$ by Eq.14 with inputs of r_{ui} and $\widehat{r}_{ui}$;

20: use BP algorithm to optimize the parameters in Θ.

19: **end for**

20: **end for**

The proposed model is implemented on Keras[1] and Tensorflow[2], both of which are publicly accessible DL framework on the Web.

4.1. Datasets

In this paper, five popular real-world datasets with different domains and different size are used in our experiments, including two movie datasets comes from GroupLens Research Group[3], MovieLens-100 k (M100k) and MovieLens-1 M (M1m)[4], a music dataset comes from Last.fm, LastFM (Lastfm)[5], and two datasets coms from Amazon, Amazon music (Amusic) and Amazon movies and TV (Amovie). It should be pointed that there is no interaction time information in Amusic and Lastfm, so we can only evaluate the effects of other techniques besides time embedding and time-aware attention mechanisms. Amovie is used to show the performance of our model in the larger dataset. Table 2 shows the summarization of the statistics of the datasets used in our experiment.

4.2. Evaluation method and metrics

We randomly select 20% of the instances in each dataset to be used as the testing set, and the remaining 80% of the instances are used as the training set. In order to assess the variability of the results and the generalization of the conclusions, we use a 5-fold cross-validation strategy to include repetitions. When modeling the dynamic user preferences, we only use the recent 20 interacted items of the user to establish his short-term preferences. In each testing, 100 unobserved interactions are randomly selected as the negative samples and mixed with the test item for matching score prediction, ranking them based on the prediction.

In this article, two popular ranking metrics, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG), are used for top-k recommendation performance evaluation. HR is a commonly used indicator to measure the recall rate of recommendation algorithms. The calculation formula of HR is defined as follows:

$$\mathrm{HR@}k = \frac{1}{n}\sum_{i=1}^{n} I(x) \tag{15}$$

where n is the number of test instances; $I(x) \in \{0,1\}$ is an indicator function that judges whether the test item $\times$ appears in the top k positions of the ranking list, and the value of k is fixed at 10 in our experiment.

NDCG measures the ranking quality recommendation results, in which the test item that appears in the higher position of the recommendation list has a higher score, as shown below:

$$\mathrm{NDCG@}k = \frac{DCG_k}{IDCG_k} \tag{16}$$

Table 2
Statistics of the datasets used in the experiments.

	M100k	M1m	Amusic	Lastfm	Amovie
Users	943	6040	1776	1741	35,896
Items	1682	3706	12,929	2665	28,586
Ratings	100,000	1,000,209	46,087	69,149	752,676
Sparsity	0.9369	0.9553	0.9980	0.9851	0.9993

[1] http://github.com/keras-term/keras
[2] http://github.com/tensorflow/tensorflow
[3] http://www.grouplens.org
[4] http://www.movieLens.umn.edu
[5] http://jmcauley.ucsd.edu/data/amazon/

$$\mathrm{DCG}_k = \sum_{i=1}^{k} \frac{2^{el_i} - 1}{\log_2(i+1)} \tag{17}$$

where DCG_k is the position-aware cumulative gain of items in the recommendation list, rel_i is the relevance of the position i in the recommendation list, k is the length of the recommendation list; IDCG_k is the maximum DCG value under ideal conditions, and it is used for normalization.

In the following experiments, we also use the rating prediction accuracy metrics, MAE and RMSE, on M100k and M1m datasets to make performance comparisons between our method and the baseline methods. Here, we use the interaction matrix based on explicit feedback (rating) information.

MAE (Mean Absolute Error) is defined as the average value of the distance between the predicted value $\hat{r}_{ui}$ and the true value r_{ui}. The formula is as follows:

$$\mathrm{MAE} = \frac{1}{N}\sum_{(u,i)\in\Omega} \left|\hat{r}_{ui} - r_{ui}\right| \tag{18}$$

where Ω denotes the testing set, and N is the size (number of samples) in the testing set.

RMSE (Root Mean Square Error) is the square root of the ratio of the square of the deviation between the predicted value and the true value to the number of observations. It measures the deviation between the predicted value and the true value and is more sensitive to outliers in the data. The formula is as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{(u,i)\in\Omega} \left(\hat{r}_{ui} - r_{ui}\right)^2} \tag{19}$$

4.3. Comparison baselines

In order to show the superiority of our method TADCF, we compare it with the state-of-the-art DL-based and time-aware recommendation methods, including:

- **DMF** (Xue et al., 2017). An MF method with DL-based representation learning, which uses the DL model to model the latent representation of users and items in a common latent space. It then predicts the user-item matching score by the dot product in the latent space. A novel loss function based on binary cross-entropy is designed to optimize on both implicit and explicit feedback. In this section, we only consider implicit feedback for fairness.
- **NCF** (He et al., 2017). A general neural CF model with DL-based matching function learning. For fair comparison, we use the implicit feedback information as the feature vectors of users and items, and the concatenation of the user and item vectors is used as the input of the MLP to predict the user-item matching score.
- **NeuMF** (He et al., 2017). A recently proposed CF model with DL-based matching function learning abilities, in which two types of feature interaction learning models, the general MF and the MLP, are integrated in a unified framework to model linear and nonlinear user-item interactions, respectively.
- **DeepCF** (Deng et al., 2019). The latest CF model with DL-based representation learning and feature interaction learning. It uses the implicit feedback information is used as the input of two different types of DL models, the DMF and the MLP, to model high-order user-item feature interactions. The outputs of the two models are fused for matching score prediction.
- **MARank** (Yu et al., 2019). A recently proposed time-aware recommendation model. It considers individual- and union level interactions of the most recent items and applies the multi-order attention mechanism to model user preferences.

- **TiSASRec** (Li et al., 2020). A state-of-the-art time interval-aware recommendation model. It explicitly models the timestamps of interactions to capture the importance of the absolute positions of items and the time intervals between them on the next item prediction.

There are signs that using DL methods is not as beneficial as one might expect (Dacrema, Boglio, Cremonesi, & Jannach, 2019; Ludewig, Mauro, Latifi, & Jannach, 2019). In order to demonstrate the superiority of the DL-based CF methods over traditional CF methods in top-n recommendations, we include the following baseline methods in our experiment:

- **Pop**. A statistical recommendation method that makes a recommendation based on the popularity of items (the number of historical interactions of items).
- **Item-CF** (Deshpande and Karypis, 2004). An item-based KNN recommendation algorithm. It uses the similarity between items to select the top-n neighbors of the target item and then make predictions based on the ratings of the neighbor items.
- **eALS** (He et al., 2017). An MF-based CF recommendation based on the implicit feedback information. It uses missing data as negative instances to train the model. Different from the classic MF method, it non-uniformly weights the instances according to their popularity.
- **PMF** (Badrul, George, Joseph, & John, 2000). A traditional CF method based on Probabilistic MF. It maps users and items into a unified latent space and then models user-item feature interactions by the inner product in the latent space.

4.4. Implementation details

To fairly compare the above recommendation methods, all of them were tested in the same environment (Intel i7-6800 K CPU @ 3.40GHZ 3.40GHZ, 32.0 GB RAM, NVIDIA GeForce GTX 1080 GPU) and use the same data source, namely the implicit feedback matrix and the interaction time matrix.

All hyperparameters of our method are set via cross-validation on the training set by experiments. Specially, we use the following parameter configurations: (1) in the embedding layer, we use a fully connected layer to obtain the item embedding of 64 dimensions. (2) in the attention layer, the short-term preferences are modeled by using the user's recent 20 interacted items, and a two-layer DNN model with dimensions of 80 and 40 is used to learn the contribution of each recently interacted item on user preference modeling; (3) in the representation learning phrase, two three-layer neural networks with 128, 64 and 32 neurons in each hidden layer are adopted to capture the latent vectors of users and items, respectively. Therefore, the number of latent factors is 64; (4) in the prediction layer, a four-layer MLP model with dimensions of 128, 64, 32 and 16 is adopted to the high-order user-item feature interactions; (5) in the objective optimization phrase, Adam optimization (Kingma & Ba, 2014) with mini-batch is used to train the model, and the learning rate and the batch size are 0.001 and 256, respectively.

In the comparison methods, for the DL-based methods (DMF, NCF, NeuMF and DeepCF) and the time-aware methods (ATRank and TiSASRec) that use the same datasets as our method, the hyperparameters in the original literature are used. For traditional non-neural methods (Pop, eALS, PMF and Item-CF), their hyperparameters are set via cross-validation on the training set by experiments. For the time-aware methods (ATRank and TiSASRec), we use the most recent interacted item of each user for testing and the remaining items for training. For the fairness of comparison, we did not filter out cold-start users and items as in the original literature. Before making any claims that our method is superior to existing methods, we also analyze the results using statistical significance tests.

4.5. Experimental results

4.5.1. Recommendation performance comparison (RQ1)

The recommendation performance of different comparison methods on datasets M100k, M1m and Amovie is shown in Table 3, and the best-performance is marked in bold. From the table, we can see the following observations:

- The performance of all methods on the ml-1 m dataset is worse than that on the ml-100 k dataset, which indicates that data sparsity has a great impact on the recommendation performance.
- MF-based models (eALS and PMF) outperform the statistical-based model (ItemPOP), indicating that linear interaction modeling bases on latent factor models plays a certain role in matching function learning.
- DL-based models (DMF and NCF) are superior to MF-based models (eALS and PMF), indicating that the DL technology has high capabilities in high-order representation modeling and nonlinear feature interaction modeling.
- The DMF performs worst in the DL-based models, and NCF performs better than the DFM, which indicates that the nonlinear learning ability of the MLP has great potential in user-item matching function learning.
- The performance of integrated models (NeuMF and DeepCF) is much better than the single model (DMF and NCF). Because the component models predict matching scores from different angles, fusing them makes the model more powerful and robust.
- The time-aware recommendation method (ATRank and TiSASRec) performs slightly better than the state-of-the-art DL-based recommendation methods DeepCF by considering the interaction timestamp information in user preference modeling. In comparison, the recommendation method considering timestamp (TiSASRec) achieves better performance than the recommendation method considering time sequence (ATRank). It shows that fine-grained time embedding can better capture users' dynamic preferences.
- On both datasets, the TADCF performs best, and the improvement is significant compared to the baseline methods. On the M100k dataset, the HR and NDCG of ATDCF are 1.87% and 4.43% higher than the best-performance baseline method, TiSASRec, respectively. On the M1m dataset, the HR and NDCG are 2.36% and 1.84% higher, respectively.
- Compared with DeepCF, the performance improvement of our model on the MAE and RMSE metrics is much smaller than that on the HR and NDCG metrics, which indicates that our model is more suitable for top-k recommendations than specific rating predictions.

In order to demonstrate that our method has a statistical advantage over the state-of-the-art time-aware and DL-based methods, we conducted a paired *t*-test to compare the average results of the TADCF with the DeepCF and TiSASRec. In each experiment, we randomly select a test item for each user as the sample set, so the size of the sample set is equal to the number of users. The last row in Table 3 shows the degree of improvement of the TADCF relative to the DeepCF, where the mark "*" indicates that the degree of improvement is statistically significant at $p < 0.05$; that is, the TADCF has significant statistical advantages over the DeepCF and TiSASRec with a confidence of 95%. We attribute the good performance of our model to the fact that attention-based dynamic user preference modeling and the seamless fusion of the two types of DL-based feature interaction learning. It is the novelty of our method and the main difference between our method and the state-of-the-art DL-based and time-aware recommendation methods.

The existing time-aware recommendation methods, including ATRank and TiSASRec, are usually used for session-based recommendations. In this article, we mainly study the CF recommendation task. Therefore, we will not make comparisons with the time-aware recommendation methods in the following experiments.

Table 3
Recommendation performance of different comparison methods.

	M100k				M1m				Amovie	
	HR	NDCG	MAE	RMSE	HR	NDCG	MAE	RMSE	HR	NDCG
POP	0.4141	0.2342	–	–	0.4415	0.2472	–	–	0.4575	0.2521
Item-CF	0.6023	0.3504	0.8947	1.1141	0.6524	0.3852	0.8935	1.1078	0.6676	0.3934
eALS	0.6144	0.3692	0.8844	0.9941	0.6748	0.4251	0.8754	1.0721	0.6887	0.4354
PMF	0.5875	0.3550	0.8914	0.9302	0.6578	0.4012	0.8969	1.0915	0.6733	0.4187
DMF	0.6111	0.3635	0.8721	0.9185	0.6781	0.4069	0.8551	0.9912	0.6845	0.4123
NCF	0.6522	0.4020	0.8219	0.8710	0.7092	0.4388	0.8256	0.9498	0.7184	0.4498
NeuMF	0.6807	0.4201	0.7036	0.7598	0.7207	0.4378	0.8066	0.9301	0.7301	0.4513
DeepCF	0.6838	0.4237	0.6996	0.7435	0.7251	0.4416	0.7885	0.9202	0.7312	0.4589
ATRank	0.7278	0.4565	0.6878	0.7311	0.7659	0.5191	0.7576	0.8667	0.7783	0.5263
TiSASRec	0.7589	0.4967	0.6701	0.7287	0.8003	0.5584	0.7332	0.8412	0.8122	0.5676
ADCF	**0.7731***	**0.5187***	**0.6624***	**0.7111***	**0.8192***	**0.5687***	**0.7254***	**0.8325***	**0.8234**	**0.5776**
Improvement	1.87%	4.43%	1.15%	2.42%	2.36%	1.84%	1.06%	1.03%	1.38%	1.76%

To further illustrate the performance of the TADCF in other domains, more experiments were carried out based on datasets Amusic and Lastfm. However, both datasets lack interaction time information. We remove the time embedding component from the TADCF and evaluate the effects of other techniques used in TADCF on top-k recommendations. We call the model without time factor TADCF-t. The experimental results of TADCF-t and baseline methods are shown in Table 4. We can see that the recommendation performance of the TADCF-t model is less than the original TADCF model, which can be seen from the NDCG of the DeepCF and the TADCF-t. It indicates that the interaction time information is important in accurate user preference learning. We can also see that even without the time factor, the TADCF-t still superior to most baseline methods and achieves performance comparable to the DeepCF. It shows the superiority of other key techniques we proposed in this article besides time embedding and time-aware attention mechanism, including short- and long-term user preference modeling, two-stage CF recommendation and matching function learning based on multiple DL models.

4.5.2. Impacts of the Time-aware attention mechanism (RQ2)

Since time-aware attention-based user modeling is the key technique proposed in our method, we conduct experiments to demonstrate its effectiveness in this section. The ranking quality of different DL-based recommendation methods with and without attention models are shown in Table 5 and Table 6. It can be seen from Table 5 that no matter which model, using an attention network can greatly improve the ranking performance. It should be pointed that we adopt a three-element attention mechanism, in which the predicted item, the recently interacted items and their interaction time are considered to model short-term user preferences. It is the difference between our method and existing attention-based methods such as ACF and NAIS, which often ignore the specific interaction time of the historical interacted items. It can be seen from Table 6 that if the interaction time of the recently interacted items is not considered in the process of attention weight learning, the ranking performance will decrease. This phenomenon indicates that the interaction time information plays an important role in user preference modeling and accurate top-n recommendations.

Table 4
Performance comparison of different methods on other datasets.

Method	Lastfm		Amusic	
	HR	NDCG	HR	NDCG
POP	0.5928	0.3562	0.4231	0.2422
Item-CF	0.8055	0.5156	0.4923	0.3454
eALS	0.8265	0.5362	0.5244	0.3742
PMF	0.8067	0.5176	0.5175	0.3650
DMF	0.8390	0.5404	0.5711	0.4035
NCF	0.8725	0.5805	0.5822	0.4120
NeuMF	0.8907	0.6101	0.7207	0.4401
DeepCF	0.8998	**0.6287**	0.7338	0.4437
ADCF-t	**0.9028**	0.6224	**0.7358**	**0.4457**

Table 5
Comparison of the effects of the attention mechanism.

Method	Attention-based?	M100k		M1m	
		HR	NDCG	HR	NDCG
DMF	Yes	0.6611	0.4135	0.7481	0.4769
	No	0.6111	0.3635	0.6781	0.4069
NCF	Yes	0.7059	0.4602	0.7544	0.4862
	No	0.6522	0.4020	0.7092	0.4388
DeepCF	Yes	0.7218	0.4769	0.7731	0.5225
	No	0.6838	0.4237	0.7251	0.4416
ADCF	Yes	0.7731	0.5187	0.7892	0.5687
	No	0.7581	0.4695	0.7255	0.5015

Table 6
Comparison of the effect of the time factor in the attention network.

Method	Time-based Attention?	M100k		M1m	
		HR	NDCG	HR	NDCG
DMF	Yes	0.6342	0.3702	0.7040	0.4132
	No	0.6111	0.3635	0.6781	0.4069
NCF	Yes	0.6759	0.4102	0.7244	0.4462
	No	0.6522	0.4020	0.7092	0.4388
DeepCF	Yes	0.7018	0.4369	0.7831	0.4625
	No	0.6838	0.4237	0.7251	0.4416
ADCF	Yes	0.8731	0.5887	0.7892	0.5687
	No	0.8551	0.5622	0.7682	0.5494

4.5.3. Sensitive analysis of hyperparameters (RQ3)

In this section, the impact of different hyperparameters in our model on recommendation performance will be analyzed. The hyperparameters to be analyzed include fusion coefficient in dynamic user modeling, latent factor size in the embedding layer and the number of hidden layers in the representation learning model and the matching score prediction model. In this section, we only select M1m as the test dataset because the test results on the M100k dataset are similar.

Figure 2 shows the HR and NDCG with different fusion coefficients when other parameters are unchanged. It can be seen from the figure that the fusion coefficient in the dynamic user modeling has a great influence on recommendation performance because accurate user modeling is the key to a high-quality recommendation. When the fusion coefficient is set to 0.5, the model gets the best performance, which shows that although short-term preferences are necessary to model user preferences, the role of long-term preferences is equally important.

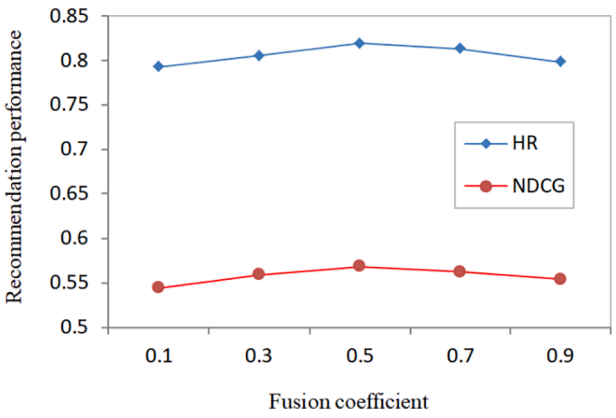

Fig. 2. Impact of fusion coefficient in dynamic user modeling.

Therefore, we cannot ignore long-term preferences in user modeling, so we include two types of preferences in the model.

Figure 3 shows the HR and NDCG on different latent factors of users and items in the embedding layer. As can be seen from the figure, in the beginning, the ranking performance gets better and better with the increase of the number of latent factors. Considering more influencing factors can achieve better recommendation performance, which is consistent with common sense. However, we should note that more factors mean higher computational complexity. The usual practice is to no longer increase latent factors when the performance improvement is not significant. Therefore, only 64 latent factors are considered in our model.

In most cases, the number of hidden layers in deep neural networks plays a key role in representations learning and matching function learning. In order to show the degree of influence, we evaluate the ranking performance of the model under different numbers of hidden layers.

Figure 4 shows the impact of using different numbers of hidden layers in the representation learning model in the DMF component on ranking performance. We can see from the figure that the ranking performance has been improved initially as the number of hidden layers increases. However, when the number of hidden layers reaches 3, the ranking performance begins to decrease. It demonstrates that the learning ability of the model has been saturated to some extent, and increasing more layers will lead to overfitting. Therefore, in this article, we use two three-layer DNN models as the representation learning models of users and items.

Figure 5 shows the HR and NDCG with different numbers of hidden layer in the matching function learning model in the DNN component on ranking performance. From the figure, we can see that the ranking performance becomes better and better as the number of hidden layers

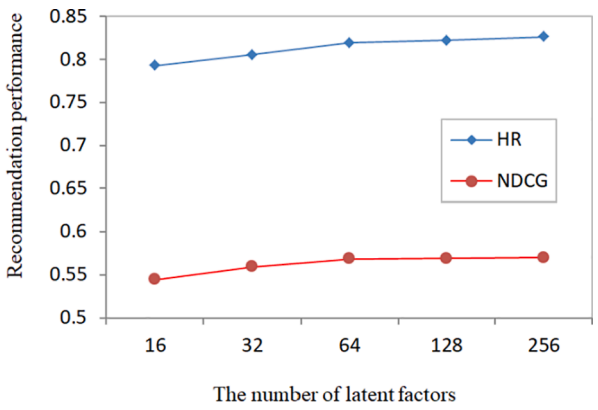

Fig. 3. Impact of the number of latent factors in matching function learning.

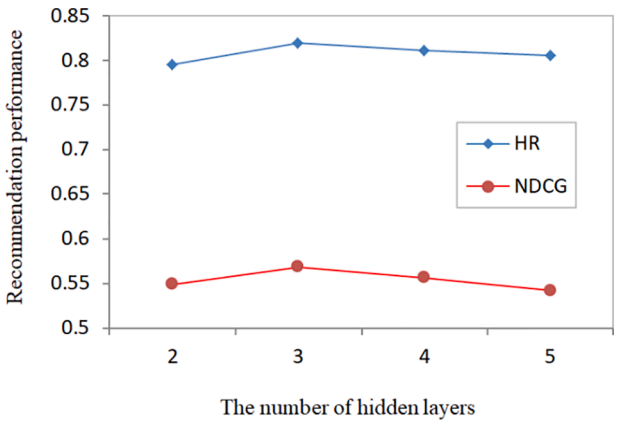

Fig. 4. Impact of the number of hidden layers in representation learning.

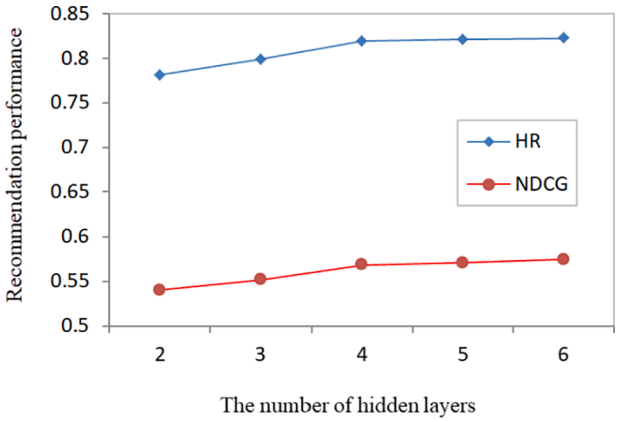

Fig. 5. Impact of the number of hidden layers in feature interaction learning.

increases. Nevertheless, when the number of hidden layers reaches 4, stacking more layers will no longer significantly improve the ranking performance, so we only keep four hidden layers in our model. The possible reason is that the user preference learning model has learned the high-order representations of the user. Therefore, a simple function is enough to model the complex matching function. It indirectly proves that optimizing the input of a DL model is important, which can relieve the burden of the subsequent hidden layers of the model and increase the recommendation accuracy.

Another finding in this experiment is that when we delete the DMF-based matching score learning part from the integrated model, the ranking performance will decrease. This means that linear feature interaction modeling is also very important in the recommendations. A good recommendation can only be achieved by using different models, as we have done in this paper.

5. Conclusion and future work

In this article, we propose a novel two-stage CF recommendation model, TADCF, for high-quality top-k recommendations. In the first stage, a time-aware attention mechanism is used to model dynamic user preferences, in which the predicted item, the recently interacted items and their interaction time are fully considered to determine the importance of different historical interacted item for short-term preference modeling. Then, short- and long-term user preferences are combined to construct the dynamic user preference model. In the second stage, a matching function learning model based on two types of DL models, DMF and DNN, are used to learn the linear and nonlinear user-item feature interactions. Finally, in the last layer of the TADCF model, the

feature interaction vectors of the two models are fused to obtain the user-item matching score. Experimental results on five real-world datasets of different sizes show that our method is significantly and consistently superior to the existing time-aware and DL-based recommendation methods in top-n recommendations.

The method we proposed in this paper deserves further study. First, we only use the implicit user-item interaction information as the data source to learn the user preference representation and the matching function. However, in real-world recommendation process, there is much auxiliary information that can be used to avoid the data sparsity problem and the cold start problems, such as the explicit ratings between the user and the item, trust relationships of users in the social network, sematic item features in the knowledge base, and other kinds of context information. Rich side information can usually bring better performance. Second, this article focuses on the role of attention mechanisms and the traditional DL techniques on top-k recommendations. CF recommendation methods based on heterogeneous information network (fusion user-item interactions with social network and knowledge graph) and the advanced DL technology (such as graph neural network) is the focus of our future work.

Declaration of Competing Interest

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

Acknowledgement

This work is supported by the Chinese National Funding of Social Sciences (No. 20BTQ093).

References

Aljunid, M. F., & Manjaiah, D. H. (2020). An efficient deep learning approach for collaborative filtering recommender system. *Procedia Computer Science, 171*, 829–836.

Andoni, A., Panigrahy, R., Valiant, G., & Zhang, L. (2014). Learning polynomials with neural networks. In *In Proceedings of the 31st International Conference on Machine Learning* (pp. 1908–1916).

Badrul, S., George, K., Joseph, K., & John, R. (2000). Application of dimensionality reduction in recommender system - a case study. In *In Proceedings of the Web Usage Analysis and User Profiling, International WEBKDD Workshop* (pp. 1–12).

Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., & Chi, E. H. (2018). In *Latent cross: making use of context in recurrent recommendation systems* (pp. 46–54). ACM.

Bobadilla J., Alonso S., Hernando A. (2020). Deep learning architecture for collaborative filtering recommender systems. Applied Sciences, 10(7): 2441(1-14).

Campos, P. G., Díez, F., & Cantador, I. (2014). Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *Kluwer Academic Publishers, 24*(1–2), 67–119.

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., & Chua, T. (2017). In *Attentive collaborative filtering: multimedia recommendation with item- and component-level attention* (pp. 335–344). ACM.

Cunha, W., Mangaravite, V., Gomes, C., Canuto, S., & Gonalves, M. A. (2021). On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Information Processing & Management, 58*(3), Article 102481.

Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T. D., Aradhye, H., & Shah, H. (2016). Wide & DL for recommendation systems. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys) (pp. 7-10). ACM.

Dacrema M. F., Boglio S., Cremonesi P. Jannach D. (2019). A troubling analysis of reproducibility and progress in recommender systems research. ACM Transactions on Information Systems, 39(2): (20)1-49.

Den Oord, A. V., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. *In NIPS*, 2643–2651.

Deng, Z., Huang, L., Wang, C., Lai, J., & Yu, P. S. (2019). DeepCF: a unified framework of representation learning and matching function learning in recommendation system. In Proceedings of the 33rd Conference on Artificial Intelligence (AAAI), (pp. 61-68). ACM.

Ding Y., Li X., Time weight collaborative filtering categories and subject descriptors (2005). In Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM) (pp. 485-492). ACM.

Du, X., He, X., Yuan, F., Tang, J., Qin, Z., & Chua, T. (2019). *Modeling embedding dimension correlations via convolutional neural collaborative filtering.* Preprint: Information Retrieval.

Elkahky, A. M., Song, Y., & He, X. (2015). A multi-view DL approach for cross domain user modeling in recommendation systems. In Proceedings of the 24th International Conference of World Wide Web (WWW) (pp. 278-288). ACM.

Gan, M., & Cui, H. (2021). Exploring user movie interest space: A deep learning based dynamic recommendation model. *Expert Systems with Applications, 173*, Article 114695.

He, X., & Chua, T. (2017). Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International Conference on Research and Development in Information Retrieval (SIGIR) (pp. 1-10). ACM.

He, X., Liao L., Zhang H., Nie, L., Hu, x., & Chua, T.-S. (2017a). Neural collaborative filtering. In Proceedings of the 26th International Conference of World Wide Web (WWW) (pp. 173-182). ACM.

He, Xiangnan, He, Zhankui, Song, Jingkuan, Liu, Zhenguang, Jiang, Yu-Gang, & Chua, Tat-Seng (2018). NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering, 30*(12), 2354–2366.

He X., Zhang H., Kan M. Y., & Chua T. S. (2017). Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 40th International Conference on Research and Development in Information Retrieval (SIGIR) (pp. 1-10). ACM.

Hidasi B., Karatzoglou A. (2017) Recurrent neural networks with top-k gains for session-based recommendations. In Proceedings of the 27th International Conference on Information and Knowledge Management (CIKM) (pp. 843-852). ACM.

Hu, Q.-Y., Zhao, Z.-L., Wang, C.-D., & Lai, J.-H. (2017). An item orientated recommendation algorithm from the multi-view perspective. *Neurocomputing, 269*, 261–272.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* Preprint: Learning.

Kiran, R., Kumar, P., & Bhasker, B. (2019). DNNRec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications, 144*, Article 113054.

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM, 53*(4), 89–97.

Li, S., Kawale, J., & Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In Proceedings of the 25th International Conference on Information and Knowledge Management (CIKM), 811-820. ACM.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural attentive session-based recommendation. In Proceedings of the 27th International Conference on Information and Knowledge Management (CIKM), 1419-1428. ACM.

Li J., Wang Y., McAuley J. (2020). Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 322-330). ACM.

Liu, Q., Chen, T., Cai, J. (2012). Enlister: Baidu's recommendation system for the biggest Chinese Q&A website. In the 6th ACM Recommender Systems Conference (RecSys). (pp. 285-288). ACM.

Liu Q., Zeng Y., Mokhosi R., Zhang H. (2018). STAMP: short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1831-1839).

Ludewig M., Mauro N., Latifi S., Jannach D. (2019). Performance comparison of neural and non-neural approaches to session-based recommendation. In proceedings of the 13th ACM Conference on Recommender Systems (RecSys) (pp. 462-466). ACM.

Luo P., Zhao, Y., Liu, F., Zhuang V. S., Sheng V. S. (2020). Collaborative self-attention network for session-based recommendation. In Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI) (pp. 2591-2597). ACM.

Lyu, Z., Dong, Y., Huo, C., & Ren, W. (2020). Deep match to rank model for personalized click-through rate prediction. In In Proceedings of the 34th Conference on Artificial Intelligence (AAAI) (pp.1-8). ACM.

Song D., Qin L., Jiang M., Liao L. (2018). A temporal and topic-aware recommender model. In Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (pp. 410-417). IEEE.

Yan, S., Wang, H., Li, Y., Zheng, Y., & Han, L. (2021). Attention-aware metapath-based network embedding for HIN based recommendation. *Expert Systems with Applications, 174*, Article 114601.

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *In Proceedings of the 30th Annual Conference on Neural Information Processing Systems* (pp. 550–558).

Vinagre, J., & Jorge, A. M. (2012). Forgetting mechanisms for scalable collaborative filtering. *Journal of the Brazilian Computer Society, 18*(4), 271–282.

Wang, Ruiqin, Cheng, Hsing K., Jiang, Yunliang, & Lou, Jungang (2019). A Novel Matrix Factorization Model for Recommendation with LOD-based Semantic Similarity Measure. *Expert Systems With Applications, 123*, 70–81. In this issue.

Wang, Ruiqin, Jiang, Yunliang, & Lou, Jungang (2020). TDR: Two-stage Deep Recommendation Model based on mSDA and DNN. *Expert Systems With Applications, 145*. In this issue.

Wang, H., Wang, N., & Yeung, D.-Y. (2015). Collaborative DL for recommendation systems. In Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining (SIGKDD), 1235-1244. ACM.

Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions. In Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (SIGKDD) (pp. 1-7). ACM.

Wang, N., Wang, H., Jia, Y., & Yin, Y. (2018). Explainable recommendation via multi-task learning in opinionated text data. In Proceedings of the 41st International Conference on Research and Development in Information Retrieval (SIGIR), (pp.165-174). ACM.

Wang Y., Zhang L., Dai Q., Sun F., Zhang B., He Y., Yan W., Bao Y. (2019). Regularized adversarial sampling and deep time-aware attention for click-through rate

prediction. In Proceedings of the 29th International Conference on Information and Knowledge Management (CIKM) (arXiv:1911.00886v1). ACM.
Wang C., Zhang M., Ma W., Liu Y., Ma S. (2020). Make it a chorus: knowledge- and time-aware item modeling for sequential recommendation. In Proceedings of the 43rd International Conference on Research and Development in Information Retrieval (SIGIR) (pp. 1-10). ACM.
Wang D., Jiang M., Syed M., Conway O., Chawla N. V. (2020) Calendar graph neural networks for modeling time structures in spatiotemporal user behaviors. In Proceedings of the 26th International Conference on Knowledge Discovery and Data Mining (SIGKDD) (pp. 2581-2589). ACM.
Wang, Ruiqin, Jiang, Yunliang, & Lou, Jungang (2021). ADCF: Attentive representation learning and deep collaborative filtering model. *Knowledge-Based Systems, 227*. In press.
Wang, Ruiqin, Jiang, Yunliang, & Lou, Jungang (2021). Attention-based dynamic user preference modeling and nonlinear feature interaction learning for collaborative filtering recommendation. *Applied Soft Computing, 6*. In this issue.
Wang, X. & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In Proceedings of the International Conference on Multimedia (pp. 627-636). ACM.
Wu S., Tang Y., Zhu Y., Wang L., Tan T. (2019). Session-based recommendation with graph neural networks. In Proceedings of the 33rd Conference on Artificial Intelligence (AAAI) (pp. 346-353). IEEE.
Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., & Chua, T. (2017). Attentional factorization machines: learning the weight of feature interactions via attention networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI) (pp.3119-3125). ACM.
Xue, H.-J.; Dai, X.-Y.; Zhang, J.; Huang, S.; & Chen,J. (2017). Deep matrix factorization models for recommendation systems. In In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI) (pp. 3203-3209). ACM.
Yu L., Zhang C., Liang S., Zhang X. (2019). Multi-order attentive ranking model for sequential recommendation. In Proceedings of the 33rd Conference on Artificial Intelligence (AAAI) (pp. 5709-5716). IEEE.
Zhang, C., Fan, Y., Xie, Y., Yu, B., Li, C., & Pan, K. (2021). Dynamic network embedding via structural attention. *Expert Systems with Applications, 176*, 114895. https://doi.org/10.1016/j.eswa.2021.114895
Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommendation systems. In Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining (SIGKDD) (pp. 353-362). ACM.
Zhang, H., Ni, W., Li, X., & Yang, Y. (2018). Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-Markov approach. *IEEE Transactions on Systems Man & Cybernetics Systems, 48*(2), 177–194.
Zhang, J., Wang, D., & Yu, D. (2021). TLSAN: Time-aware long- and short-term attention network for next-item recommendation. *Neurocomputing, 444*, 179–191.
Zhao, Z.-L.; Huang, L.; Wang, C.-D.; and Huang, D. (2018). Low-order and sparse cross-domain recommendation algorithm. In Proceedings of the 24th International Conference on Database Systems for Advanced Applications (pp. 150-157).
Zheng, N., Li, Q., & Liao, S. (2010). Which photo groups should I choose? A comparative study of recommendation algorithms in Flickr. *Journal of Information Science, 36*(6), 733–750.
Zhou, C., Bai, J., Song, J., Liu, X., Zhao, Z., Chen, X., & Gao, J. (2018). ATRank: an attention-based user behavior modeling framework for recommendation. In Proceedings of the 32nd Conference on Artificial Intelligence (AAAI) (pp. 4564-4571). IEEE.
Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems, 22*(1), 143–177.