Neural Time-Aware Sequential Recommendation by Jointly Modeling Preference Dynamics and Explicit Feature Couplings

Qi Zhang, *Student Member, IEEE*, Longbing Cao, *Senior Member, IEEE*, Chongyang Shi, and Zhendong Niu

***Abstract*— In recommendation, both stationary and dynamic user preferences on items are embedded in the interactions between users and items (e.g., rating or clicking) within their contexts. Sequential recommender systems (SRSs) need to jointly involve such context-aware user–item interactions in terms of the couplings between the user and item features and sequential user actions on items over time. However, such joint modeling is non-trivial and significantly challenges the existing work on preference modeling, which either only models user–item interactions by latent factorization models but ignores user preference dynamics or only captures sequential user action patterns without involving user/item features and context factors and their coupling and influence on user actions. We propose a neural time-aware recommendation network (TARN) with a temporal context to jointly model 1) stationary user preferences by a feature interaction network and 2) user preference dynamics by a tailored convolutional network. The feature interaction network factorizes the pairwise couplings between non-zero features of users, items, and temporal context by the inner product of their feature embeddings while alleviating data sparsity issues. In the convolutional network, we introduce a convolutional layer with multiple filter widths to capture multi-fold sequential patterns, where an attentive average pooling (AAP) obtains significant and large-span feature combinations. To learn the preference dynamics, a novel temporal action embedding represents user actions by incorporating the embeddings of items and temporal context as the inputs of the convolutional network. The experiments on typical public data sets demonstrate that TARN outperforms state-of-the-art methods and show the necessity and contribution of involving time-aware preference dynamics and explicit user/item feature couplings in modeling and interpreting evolving user preferences.**

***Index Terms*— Convolutional neural network (CNN), coupling learning, explicit features, feature couplings, recommender system, sequential recommendation, time-aware sequential recommendation, user preference dynamics.**

Manuscript received 25 March 2019; revised 30 November 2020; accepted 12 March 2021. Date of publication 14 April 2021; date of current version 6 October 2022. This work was supported in part by the Australian Research Council Discovery Grants under Grant DP190101079 and Grant FT190100734; in part by the National Key Research and Development Program of China under Grant 2018YFB1003903; in part by the National Natural Science Foundation of China under Grant 61502033, Grant 61472034, Grant 61772071, Grant 61272361, and Grant 61672098; and in part by the China Scholarship Council under Grant 201806030191. *(Corresponding authors: Longbing Cao; Chongyang Shi.)*

Qi Zhang is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Data Science Lab, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: qi.zhang-13@student.uts.edu.au).

Longbing Cao is with the Data Science Lab, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: longbing.cao@uts.edu.au).

Chongyang Shi and Zhendong Niu are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: cy_shi@bit.edu.cn; zniu@bit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3069058.

Digital Object Identifier 10.1109/TNNLS.2021.3069058

I. Introduction

RECOMMENDER systems play increasingly important roles in various domains in the age of information explosion, e.g., social media and e-commerce, by suggesting products and services (called items in general) that match people's interest. In general, recommendations are predicted based on the analysis of existing user actions on items such as user clicks and purchases of items and user ratings, which is called user–item interactions and reflects user preferences for items. The user–item interactions can be characterized in terms of observable (explicit) user/item features, contextual factors and their couplings, and user sequential actions on items over time [1]–[3]. During the interactions, users not only maintain their stationary preferences within a certain context (e.g., in browsing a movie website, artists may prefer musical movies while children may be more excited about newly released animation) but also adapt their preferences to new or other items over time or contextual change (such as the successive release of new movies or circumstances change). Sequential recommender systems (SRSs) should cater to both stationary and dynamic user preferences. Fig. 1 illustrates the sequential recommendation of movies by considering the couplings among the user's age, movie genres, and the current date and the relations between movie series.

When explicit user/item features are available, modeling their feature couplings [2] may disclose their driving roles on user preferences on items and improve recommendation effectiveness and comprehensibility [3]. Existing approaches such as factorization machines (FM) [4], statistical recommendation learning [5] and DeepFM [6] embed explicit features into a low-rank latent space and factorize user–item interactions (e.g., ratings and clicks) and feature couplings in terms of the pairwise inner product of the embedded latent vectors or latent variable relation modeling. Involving more features (especially categorical features which are converted to one-hot representations) often makes input feature vectors highly sparse [7]

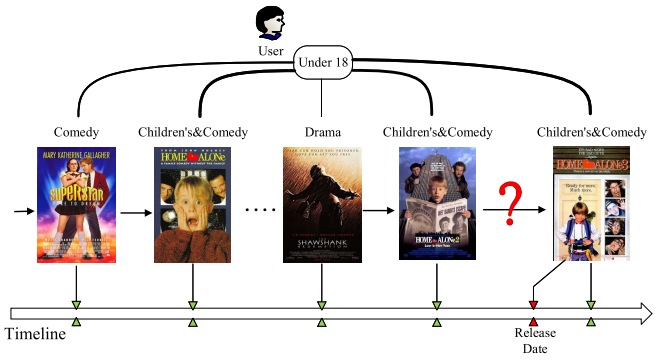

Fig. 1. Example to illustrate the motivation of user's action on movies. A user under 18 years old watched a series of movies (labeled by movie genres). Lower green arrows denote the time points when the movies were watched, and the upper lines indicate the couplings between user feature "Under 18 years old" and movie genres where the thickness of lines refers to the strength of user/item feature couplings. At the current time (the last one in the figure), the film Home Alone 3 may be highly recommended to the user after its release since 1) the user likely prefers children's and comedy movies and 2) there is a sequential evolution from Home Alone 1 to Home Alone 3.

and the computation costly especially on recommendation data that is usually sparse and "long-tailed" as new items or users are continually introduced [5]. It is extremely challenging to model user–item interactions and feature couplings on large, sparse, and evolving data [5], [8].

Although modeling explicit feature couplings is helpful for disclosing some factors driving user-item interactions and user stationary preferences on items, SRSs have to also consider the time influence [9], [10] on user preferences and contextual change. In reality, recommendation data such as user's clicks, check-ins, or e-commerce transactions are stamped with their occurrence timeframes. Such time information reflects user action sequencing and preference dynamics [8], [11]–[13], which should be combined with the above user stationary preferences modeling for a more comprehensive understanding and interpretability of user preference formation and evolution [14] and informing recommendation transition.

Incorporating time dynamics into typical temporal and sequential models is reflected in recent work, such as Markov chain-based models [15], [16], recurrent neural networks (RNNs) [10], [17], convolutional neural networks (CNNs) [18], [19], and attention networks [20], [21]. While these models capture transitional patterns of user action sequences and achieve state-of-the-art performance in user preference modeling, there are still several issues in user action sequences worthy of further exploration. First, time-specific information may not be involved in the modeling, instead they rely on temporal order. Second, multi-fold sequential patterns such as union-level and skip-behavior patterns [18] may exist in sequential user actions, where Markov chain-based and RNN-based models are not suitable. Furthermore, user sequential actions take place in their corresponding temporal context which influences future actions [22], [23]. Lastly, explicit user/item feature couplings not only disclose driving factors of user stationary preferences [1], [3] they also supplement sequential modeling, e.g., in alleviating the cold-start issues.

Accordingly, neural time-aware recommendation networks (TARNs) are introduced to jointly consider user stationary preferences and preference dynamics for sequential recommendation by involving time-specific information, explicit feature couplings, and user action sequences. Specifically, we utilize a feature interaction network to represent user stationary preferences by factorizing the pairwise couplings between the non-zero features of users, items, and context through the inner product of their feature embeddings, which can alleviate the data sparsity issues. By assuming user preference dynamics is attributed to the temporal dynamics within a user's sequential actions and is strongly related to the user's recent action sequence, we introduce a convolutional layer with multiple filter widths to capture multi-fold sequential patterns by taking the advantage of CNNs in extracting multi-granular local features [24], [25]. We further propose a temporal action embedding to represent user actions, which combines the embeddings of items and temporal context as the input of the convolutional network to make the network aware of the temporal context.

Specifically, our main contributions include the following.

1) The proposed TARN jointly models user stationary preferences and preference dynamics for sequential recommendation. Specifically, TARN has a feature factor network to factorize user/item/context feature couplings and a convolutional network fed by user action sequences to model the temporal dynamics of user preferences.
2) To make the convolutional network sensitive to a temporal context, a novel temporal action embedding is proposed to embed items and the temporal context into the same space and represent user actions by combining their embeddings as the inputs of the convolutional network.
3) An attentive average pooling (AAP) is introduced upon the convolutional outputs to obtain significant features and large-span feature combinations, facilitating effective information passing to higher layers.
4) We provide visualization to interpret the rationale and the interesting findings of the model.

The empirical results on three real-world data sets show that: 1) the proposed mechanisms in TARN for modeling feature couplings and user preference dynamics are effective; 2) TARN outperforms the state-of-the-art methods in terms of various verification aspects; and 3) both the temporal action representation and AAP have a positive impact on improving recommendation performance.

II. Related Work

The value of involving observable user/item/context features and modeling their coupling relationships [3], [26], [27] has been increasingly recognized in recommendation research since they reflect the "state of user/item/context appearance" and contribute to the reality and challenges of recommendation [1]. Previous work, such as FM [4] captures latent factors in a low-dimensional space and factorize both first- and second-order feature couplings via the inner product of feature embedding vectors. Recent work such as NFM [7],

DeepFM [6], and DCN [28] further feeds latent feature embeddings into deep networks to capture higher-order and nonlinear feature couplings. Such neural models capture latent relations between ratings and latent features and outperform the aforementioned shallow recommenders. However, these shallow and neural models only learn stationary user preferences but ignore user preference dynamics over time, thus they are impractical for real-life applications.

There has been an increasing number of models incorporating temporal dynamics into user preference modeling, which can be roughly categorized into two basic approaches: temporal modeling [8], [29], [30] and sequential modeling [10], [17]–[20] which is also our focus in this work. Typical sequential modeling methods include Markov chain-based methods such as [15], [16] which factorize the transition matrix of user successive actions to capture the transitional patterns of a user action, which are not suitable for capturing high-order sequential patterns and the long-term dependence between user actions. The recent neural network-based models, such as RNN-based [10], [17], [31], [32] and CNN-based [18], [19] models prevail over the early sequential models and achieve excellent results in the sequential recommendation. Their success relies on the strong representation capability of deep neural networks and their sequential modeling through capturing precise action transition patterns to properly represent user preferences and dynamics. With the great success of attention mechanism in modeling global dependences between input and output [33]–[35], attention neural networks have been introduced into sequential recommendation and achieve state-of-the-art performance and good interpretability.

However, the above models either ignore user stationary preferences or only model stationary preferences by factorizing latent user-item interactions without utilizing explicit user/item/context features. Models which do not capture feature couplings cannot characterize the intrinsic cause of interactions between users and items and may greatly suffer from cold-start problems.

As our work focuses on neural sequential modeling, we further review the latest progress achieved on SRSs. Recent advanced deep neural models, such as graph neural networks (GNNs) [36], [37], memory networks [38], [39], and deep reinforcement learning [40] have been introduced into SRSs and achieve great success due to their strength for modeling and capturing the comprehensive relations within user action sequences [41]. As these models take different mechanisms and designs from our proposed TARN, we will not compare them with TARN in the experiments.

The models most relevant to ours are recurrent recommendation networks (RRN) [10], convolutional sequence embedding recommendation model (Caser) [18], and sequential temporal context-aware recommendation (STAR) [42]. RRN models both item changes and user preference changes by two separative LSTM auto-regressive models in addition to a low-rank factorization model upon user/item variables to capture stationary preferences. Caser captures sequential patterns by a convolutional network and combines a user latent vector to model user stationary preferences. Both RRN and Caser do not consider the influence of temporal context on user preference dynamics. SITAR based on STAR involves (temporal) context in stacked RNNs for sequential recommendation and captures the dynamics of contexts and temporal gaps. Furthermore, these three models neglect the interactions between explicit users, items, and temporal context and suffer from modeling user stationary preferences and alleviating cold-start issues. Different from the above models, our proposed TARN models user stationary preferences by capturing the interactions between explicit user/item/temporal features and involving temporal context into convolutional networks to model user preference dynamics, which improves model comprehensibility and recommendation performance. Considering that CNNs are used to capture multi-fold sequential patterns in TARN, we also investigate the recent CNN-based methods for fashion recommendation [30], [43]–[45], hashtag recommendation [46], and news and document recommendation [47], [47], [48]. These methods apply CNNs to process images and text, which is quite different from our work utilizing CNNs to model user action sequences, thus we also do not compare TARN with these methods.

III. TARN Design

Here, we introduce the TARN design which incorporates a convolutional network to learn the temporal dynamics of user preferences and a feature factor model to capture user stationary preferences. As shown in Fig. 2, TARN consists of four components: embedding layers, convolutional network, feature interaction layers, and output layers. TARN models the collaborative effect of user/item features, sequential actions, and the temporal context on user actions on items by fusing the three aspects of information to generate sequential recommendations.

Formally, let $\mathcal{I}$ be the set of items, $\mathcal{U}$ be the set of users, and $\mathcal{X}_I$ and $\mathcal{X}_U$ are the corresponding explicit item feature matrix and user feature matrix, respectively. Each user is associated with a sequence of historical actions:

$$\mathcal{S}_u = \{(i_1, t_1), (i_2, t_2), \ldots, (i_{|\mathcal{S}_u|}, t_{|\mathcal{S}_u|})\} \tag{1}$$

where each action (i, t) indicates that item $i \in \mathcal{I}$ is addressed at timestamp t. Assuming that a user's action at a time is influenced by his/her prior actions, we apply a N-sized window sliding over the user's action sequence to construct a clipped action sequence set. We generate the clipped sequence set for a user u as follows:

$$\left\{\mathcal{S}^{N+1}_{u,1:N}, \mathcal{S}^{N+2}_{u,1:N}, \ldots, \mathcal{S}^{|\mathcal{S}_u|}_{u,1:N}\right\}. \tag{2}$$

$\mathcal{S}^{p}_{u,1:N} = \{(i_{p-N}, t_{p-N}), (i_{p-N+1}, t_{p-N+1}), \ldots, (i_{p-1}, t_{p-1})\}$ is the clipped sequence denoting the N successive actions of user u before time step p. Given the item feature matrix $\mathcal{X}_I$, the feature vector $X_u \in \mathcal{X}_U$, and the clipped sequence $\mathcal{S}^{t}_{u,1:N}$ of user u, the recommendation problem is to suggest new items to user $u \in \mathcal{U}$ at time step t with those items that are not in $\mathcal{S}_u$ but are likely to be favored by the user.

A. Modeling Temporal Dynamics

Leveraging the advantages of CNNs in capturing local features and relations [25], [49], our method applies a convolutional layer with multi-width filters to discover multi-fold

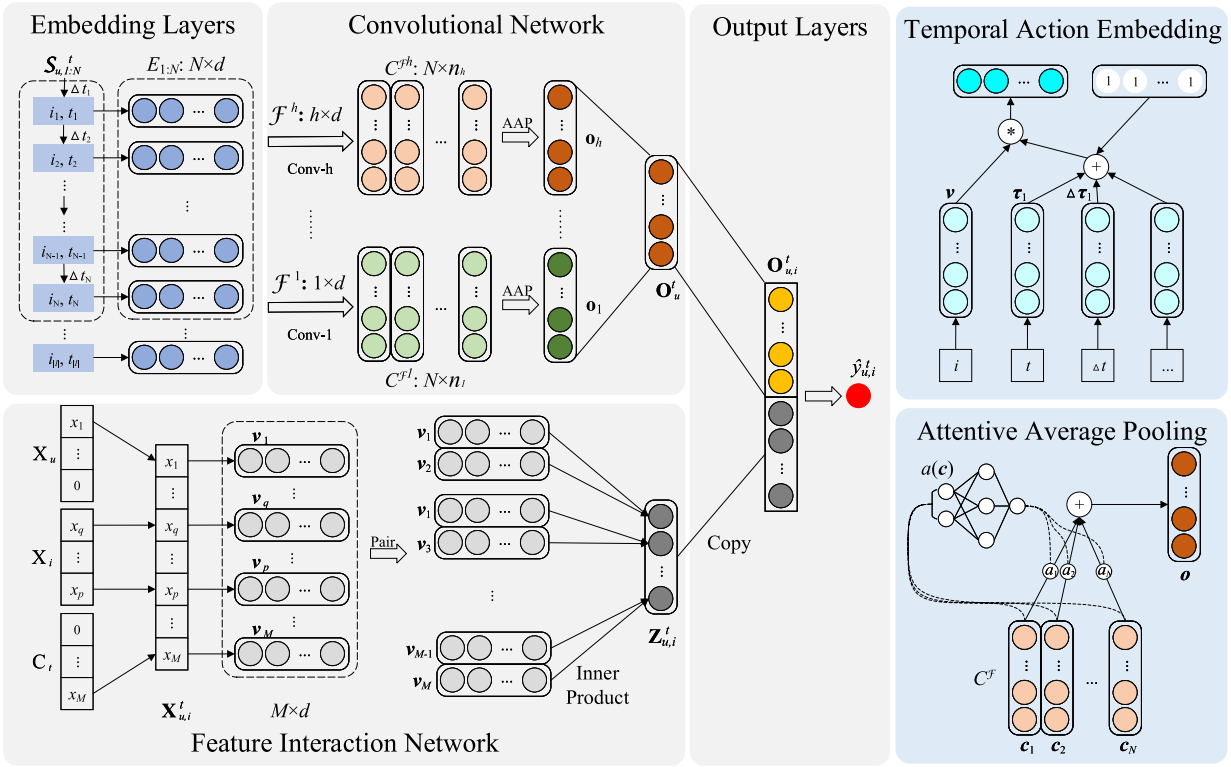

Fig. 2. TARN architecture for the time-aware modeling of user–item interactions by involving user action sequences and user/item/temporal feature couplings. The N-size action sequence $\mathcal{S}^t_{u,1:N}$ of user u before time t is fed into the embedding layers and then the convolutional layers to generate the representation of temporal dynamics. The feature vector $\boldsymbol{X}^t_{u,i}$ containing user features $\boldsymbol{X}_u$, item features $\boldsymbol{X}_i$ and temporal context $\boldsymbol{C}_t$ is fed to the feature interaction network to represent the interactions of user/item/temporal features. These two aspects of representations are combined in the output layers to predict the action at time t. The right two subcomponents show the action embedding method and the pooling strategy used in the embedding layers and convolutional network, respectively.

sequential patterns of user actions on items. First, we introduce the proposed temporal action embedding.

1) Temporal Action Embedding: To make the convolutional network sensitive to a temporal context, we introduce a temporal action embedding method to represent user actions. It embeds item variables, the time of the current action, and the time interval between the current action and the previous action in the same low-dimensional space and combines these embeddings as the representation of user sequential actions, i.e., the input of the convolutional layer in TARN.

First, we utilize time encoding schemes to transform a standard timestamp to a unique time id for embedding. Specifically, the schemes are customized per application scenarios to extract time factors, e.g., hour, day, and weekday types, from a timestamp. To choose a proper time encoding scheme, we analyze the relationships between user actions with different time factors and try different schemes to test their performance.[1] Since the schemes cannot extract all the time factors and do not completely depict the temporal shift between successive actions, we further consider the time interval between successive actions which can be interpreted as the period of user absence from the recommender system. The time interval is calculated as follows, where t_j is the timestamp at j-th action:

$$\Delta t_j = \lfloor \log(t_j - t_{j-1} + 1) \rfloor. \tag{3}$$

[1]We extract hour for MovieLens and Last.fm and day types (weekend and weekday) for Tafeng in the experiments after trying different schemes.

Here, Δt_j reflects how long a user has been absent from the recommender system at the j-th action ($\Delta t_1 = \max_j \Delta t_j$). Assuming that the influence of the time interval Δt_j becomes small with the time interval being larger, we use the logarithmic function to rescale the time interval and adopt floor function ($\lfloor \cdot \rfloor$) to convert a scaled time interval to a positive integer for embedding. Note that the converted integers have an upper bound and are enumerable since timestamp t_j is bounded by the max timestamp in each application.

After encoding the timestamps and transferring time intervals into integers, we embed these two aspects of time information and combine their resultant embeddings with item embeddings to obtain temporal action representation as shown in the temporal action embedding in Fig. 2. Specifically, given any N-action sequence clipped at time step t of user u, i.e., $\mathcal{S}^t_{u,1:N} = \{(i_1, t_1), (i_2, t_2), \ldots, (i_N, t_N)\}$, the sequence is represented by the following embedding matrix (for concision, we omit subscript u and time step t in the following):

$$\begin{aligned} &E_{1:N} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_N\}^T \in \mathbb{R}^{N \times d} \\ &\text{s.t. } \boldsymbol{e}_j = \boldsymbol{v}_j * (\boldsymbol{\tau}_j + \Delta\boldsymbol{\tau}_j + \mathbf{1}), \quad j \in \{1, 2, \ldots, N\} \end{aligned} \tag{4}$$

where $\boldsymbol{e}_j$ denotes the embedding vector for j-th action. $\boldsymbol{v}_j, \boldsymbol{\tau}_j, \Delta\boldsymbol{\tau}_j \in \mathbb{R}^{d \times 1}$ are the embedding vectors for item i_j, timestamp t_j and time interval Δt_j, respectively, and $*$ denotes the element-wise product. Inspired by the idea of initializing context embeddings in [31], we initialize $\boldsymbol{\tau}$ and $\Delta\boldsymbol{\tau}$ by 0-mean

Gaussian and thus treat the context term $(\boldsymbol{\tau}_j + \Delta\boldsymbol{\tau}_j + \mathbf{1})$ as a mask over item embedding. Note that the proposed temporal action embedding can easily be extended to integrate other contextual information such as location by summing all context embeddings in the multiplicative item in (4).

2) Multi-Width Convolutional Network: By assuming that a user's action at time t is influenced by his/her recent actions, we represent the previous action sequence to model the temporal dynamics of user preferences. Specifically, we feed $N \times d$ embedding matrix $E_{1:N}$ of the N-action sequence into the convolutional layers. Herein, we utilize d-height convolutional filters to slide over the embedding matrix to capture the action-level local feature combinations. Considering the existence of multi-fold sequential patterns, we leverage a convolutional layer with multiple filter widths which is formulated as follows (shown in the convolutional network in Fig. 2):

$$\begin{aligned} \mathcal{F} &= \mathcal{F}^1 \cup \mathcal{F}^2 \cup \cdots \cup \mathcal{F}^H \\ \mathcal{F}^h &= \{f^l \in \mathbb{R}^{h\times d} | l \in \{1, \ldots, n_h\}\}, \quad h \in \{1, \ldots, H\} \end{aligned} \tag{5}$$

where H is the maximum filter width, h is the width of a filter and n_h denotes the number of filters with a width h. Intuitively, $\mathcal{F}^h$ denotes the set of h width filters, and its l-th filter f^l treats the embedding vector of each action as a whole and covers h actions each step during convolutional calculation. Thus, given the embedding $E_{1:N}$, the j-th convolution feature by the wide convolution[2] of filter $f^l \in \mathcal{F}^h$ is given by:

$$c_j^{h,l} = \phi_a(f^l \cdot E_{j:j+h-1} + b_c), \text{ s.t. } j \in \{1, 2, \ldots, n_h\}. \tag{6}$$

Here, operator $\cdot$ computes the inner-product, b_c is a bias term, and ϕ_a is a non-linear activation function ($ReLU$ in TARN). $E_{1:N}$ is extended to $E_{1:N+h-1}$ with 0 padding for wide convolution when calculating $c_j^{h,l}$. The filter f^l slides along the embedding matrix $E_{1:N}$ to produce a feature map: $C^{h,l} = [c_1^{h,l} c_2^{h,l} \cdots c_N^{h,l}]$ for the l-th filter in $\mathcal{F}^h$. To differentiate different granular features, we stack the convolutional outputs extracted by filters with the same width h, i.e., $\mathcal{F}^h$:

$$C^{\mathcal{F}^h} = \begin{bmatrix} C^{h,1} \\ C^{h,2} \\ \vdots \\ C^{h,n_h} \end{bmatrix}^T = \begin{bmatrix} c_1^{h,1} c_1^{h,2} \cdots c_1^{h,n_h} \\ c_2^{h,1} c_2^{h,2} \cdots c_2^{h,n_h} \\ \vdots \\ c_N^{h,1} c_N^{h,2} \cdots c_N^{h,n_h} \end{bmatrix}. \tag{7}$$

Consequently, for all the filters in $\mathcal{F}$, the convolutional output over the embedding matrix $E_{1:N}$ is,

$$C^{\mathcal{F}} = [C^{\mathcal{F}^1} C^{\mathcal{F}^2} \cdots C^{\mathcal{F}^H}]. \tag{8}$$

3) Attentive Average Pooling: We introduce an AAP over each stacked convolutional output, i.e., $C^{\mathcal{F}^h}$, to capture the large-span feature combinations shown in Fig. 2. Additionally, the attentive pooling learns to distribute high weights to useful features, facilitating effective information passing to high-level networks.

Specifically, we denote $C^{\mathcal{F}^h} = [\mathbf{c}_1 \mathbf{c}_2 \cdots \mathbf{c}_N]^T$ for concision where $\mathbf{c}_j \in \mathbb{R}^{n_h \times 1}$ represents the j-th row in matrix $C^{\mathcal{F}^h}$. Obviously, $\mathbf{c}_j$ consists of the convolutional outputs of all filters $f \in \mathcal{F}^h$ at jth position (action). The weight of each position $j \in \{1, 2, \ldots, N\}$ is formulated as follows:

$$a_{hj} = \frac{exp(\mathbf{L}^T ReLU(\mathbf{W}_a \mathbf{c}_j + \mathbf{b}_a))}{\sum_{\mathbf{c}_j \in C^h} \exp(\mathbf{L}^T ReLU(\mathbf{W}_a \mathbf{c}_j + \mathbf{b}_a))} \tag{9}$$

where $\mathbf{W}_a \in \mathbb{R}^{d_a \times n_h}$, $\mathbf{L} \in \mathbb{R}^{d_a \times 1}$ and $\mathbf{b}_a \in \mathbb{R}^{d_a \times 1}$ are identical for each $C^{\mathcal{F}^h}$ for simplicity and convenience for training, meaning that all convolutional outputs share attention parameters. The attentive average pooling over C_h is built upon the weighted sum of $\mathbf{c}_j$:

$$\mathbf{o}_h = \sum_{j=1}^{N} a_{hj} \mathbf{c}_j^T, \text{ s.t. } \sum_{j=1}^{N} a_{hj} = 1, o_h \in \mathbb{R}^{n_h \times 1}. \tag{10}$$

For any $C^{\mathcal{F}^h} \subset C^{\mathcal{F}}$, we obtain a corresponding new representation $\mathbf{o}_h$ by the attentive average pooling. We then concatenate the series of $\mathbf{o}_h$ and feed them into a fully connected neural network to further extract high-level features:

$$\mathbf{O} = \sigma_o \left(\mathbf{W}_o \begin{bmatrix} \mathbf{o}_1 \\ \mathbf{o}_2 \\ \vdots \\ \mathbf{o}_H \end{bmatrix} + \mathbf{b}_o \right) \tag{11}$$

where $W_o \in \mathbb{R}^{d \times (H * n_h)}$ is a weight matrix, $\mathbf{b}_o \in \mathbb{R}^{d \times 1}$ is the corresponding bias term and σ_o is the activation function (specifically, $ReLU$ is used in experiments). $\mathbf{O}$ as the final output of the convolutional layers stands for the sequence representation reflecting user preference dynamics. Correspondingly, we denote the sequence representations of $\mathcal{S}_{u,1:N}^t$ with $\mathbf{O}_u^t$ which reflects the preference dynamics of user u at time step t. The vector representing the temporal dynamics of user preferences is fed into a concatenation layer along with the feature coupling vector, the details of which are discussed in Section III-C.

B. Modeling Feature Couplings

Recent studies demonstrate that feed-forward neural networks are inefficient in capturing feature couplings [31]. Inspired by obtaining the second-order couplings in FM, we calculate the feature couplings between explicit features by the inner-product of the corresponding feature embedding vectors as shown in the feature interaction network in Fig. 2. Since the feature vector is highly sparse, we only consider the pairwise couplings between non-zero feature vectors. In addition, contextual information also interacts with user/item features, which finally influences user actions. For this reason, we combine the embedded user, item, and context features into a vector and model their coupling relationships.

Given a user u and an item i at time step t, $\boldsymbol{X}_{u,i}^t = [x_1, \ldots, 0, x_q, \ldots, x_p, 0, \ldots, x_M]$ denotes the concatenated feature vectors of user u ($\boldsymbol{X}_u \in \mathcal{X}_U$), item i ($\boldsymbol{X}_i \in \mathcal{X}_I$), and temporal context at timestamp t ($t, \Delta t \in \boldsymbol{C}_t$), shown in Fig. 2. Note that the feature interaction network is flexible so other context features, such as location are easily incorporated in the network by concatenating the feature vectors of other contexts with $\boldsymbol{X}_{u,i}^t$. In $\boldsymbol{X}_{u,i}^t$, categorical features are converted to binary features by one-hot encoding while numerical features remain

[2]Wide convolution guarantees the same heights of convolutional input and output and retains more feature combinations near the end of the sequence.

unchanged. Furthermore, M denotes the number of original categorical and numerical features. We then calculate the pairwise non-zero feature couplings as follows:

$$\begin{aligned}&\mathbf{Z}^t_{u,i} = [z_{1,2}, z_{1,3}, \ldots, z_{M-1,M}]^T\\ &\text{s.t.}\quad z_{p,q} = x_p x_q \mathbf{v}_p \cdot \mathbf{v}_q,\ \ p < q,\ \ x_p, x_q \in \boldsymbol{X}^t_{u,i}\end{aligned} \tag{12}$$

where $\mathbf{v}_p, \mathbf{v}_q \in \mathbf{V}$ denotes the d-dimension position embedding vectors of the p-th and q-th feature within $\boldsymbol{X}^t_{u,i}$, and $\mathbf{V} \in \mathbb{R}^{|\boldsymbol{X}^t_{u,i}| \times d}$ denotes the position embedding matrix. Intuitively, value $z_{p,q}$ measures the correlations between the p-th feature and the q-th feature, and $\mathbf{Z}^t_{u,i}$ captures the correlations between the user, item, and context features in the granularity of their pairwise (second-order) couplings. Note that both the inputs of the convolutional network and feature interaction network (i.e., $\mathcal{S}^t_{u,1:N}$ and $\boldsymbol{X}^t_{u,i}$) involve the same temporal information. We thus use shared embeddings for the temporal information.

Different from common FM-based models, such as DeepFM, NFM, and Deep&Wide [50] which involve both first-order and second-order (pairwise) feature couplings, we are only interested in pairwise feature relations, which is demonstrated to be adequate and effective as shown in Section IV-D. Additionally, the high-order feature relations adopted in recent models like NFM and DCN [28] can model more complex feature relations to improve model performance to some extent. However, introducing high-order feature couplings incurs a high computational burden or makes it difficult to interpret the couplings between features. To this end, we simplify the whole model and only adopt the second-order feature couplings.

In addition, not all feature couplings are equally constructive, and useless couplings may introduce noise and degrade the performance. It is crucial to flexibly distribute weights for different feature couplings. Instead of using the attention mechanism as in AFM [51], we copy the feature coupling vector $\mathbf{Z}^t_{u,i}$ to the output layer in the integration with the convolutional output in (13) and learn the weights directly, which makes our model more flexible than the aforementioned FM-based methods.

C. Prediction and Inference

We combine the sequence representation and feature couplings for prediction. Before feeding the combined vector into the output layers as shown in Fig. 2, we multiply $\mathbf{O}^t_u$ with an item-related vector to specify the sequence representation for each particular item and then stack the resultant vector with the feature coupling vector, which is formulated as:

$$\hat{y}^t_{u,i} = \sigma\left(\mathbf{w}^T \begin{bmatrix}\mathbf{O}^t_{ui}\\ \mathbf{Z}^t_{ui}\end{bmatrix} + b\right) = \sigma\left(\mathbf{w}^T \begin{bmatrix}\mathbf{O}^t_u \circ \widetilde{\mathbf{v}}_i\\ \mathbf{Z}^t_{ui}\end{bmatrix} + b\right) \tag{13}$$

where $\widetilde{\mathbf{v}}_i \in \widetilde{\mathbf{V}} \in \mathbb{R}^{|\mathcal{I}| \times d}$ is a transformation vector specified for item i, $\mathbf{w} \in \mathbb{R}^{(d+|\mathbf{Z}_{ui}|)\times 1}$ and $b \in \mathbb{R}$ are the output weight vector and bias respectively. Operator $\circ$ denotes the element-wise product and $\sigma(\cdot)$ is the Sigmoid function. $\hat{y}^t_{u,i}$ predicts the probability of a target item i for user u at time step t based on the sequence representation $\mathbf{O}^t_u$ and feature coupling vector $\mathbf{Z}^t_{ui}$. Thus, we denote:

$$p\big(i|\mathcal{S}^t_{u,1:N}, \mathcal{X}_U, \mathcal{X}_I\big) = \hat{y}^t_{u,i}. \tag{14}$$

To train the model, an N-width window slides over the action sequence of each user u (padding 0 is performed if the length is smaller than N) to generate a training sequence set, i.e., the clipped sequence set $\{\mathcal{S}^1_{u,1:N}, \mathcal{S}^2_{u,1:N}, \ldots, \mathcal{S}^T_{u,1:N}\}$, and the corresponding next predictive action (precisely item) set of each clipped sequence $\mathcal{S}^t_{u,1:N}$ is treated as $\mathcal{I}^u_t$. We denote the collection of time steps to predict $\mathcal{T}^u = \{1, 2, \ldots, T\}$ for each user u. The likelihood of all the sequences in the training set is given by:

$$p(\mathcal{S}|\Theta, \Omega) = \prod_{u\in U}\prod_{t\in\mathcal{T}_u}\prod_{i\in\mathcal{I}^u_t} \hat{y}^t_{u,i} \prod_{\bar{i}\notin\mathcal{I}^u_t} \big(1 - \hat{y}^t_{u,\bar{i}}\big). \tag{15}$$

Taking the negative logarithm of the likelihood, we have the objective function w.r.t. the cross entropy loss:

$$\mathcal{L}_{\Theta,\Omega} = \sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{T}^u}\sum_{i\in\mathcal{I}^u_t} -\log\big(\hat{y}^t_{u,i}\big) + \sum_{\bar{i}\notin\mathcal{I}^u_t} -\log\big(1 - \hat{y}^t_{u,\bar{i}}\big) \tag{16}$$

where $\bar{i}$ is a sampled negative item (in the experiment we adopt the commonly-used random sampling for efficient training) and $\Theta = \{\mathbf{V}, \boldsymbol{\nu}, \boldsymbol{\tau}, \Delta\boldsymbol{\tau}, \mathbf{W}_a, \mathbf{W}_o, \mathbf{w}, \mathbf{L}, \widetilde{\mathbf{V}}, \mathbf{b_a}, \mathbf{b_o}, b\}$ are model parameters learned by minimizing the objective function. Furthermore, the hyperparameters $\Omega = \{d, N, H, n_h, d_a\}$ are selected via empirical experiments and grid search. After all the parameters are learned, (13) is used to calculate the probability for all items, and the items with the top-K highest probability are recommended to users.

Equation (13) shows that the prediction is made on top of two aspects of information: the previous action sequence and the user, item and context-combined features. Actually, by treating the convolutional output as a "temporal bias" (preference dynamics), user actions are reflected in the stationary feature couplings. Let us rewrite (13):

$$y^{u,t}_i = \sigma\left(\begin{bmatrix}\mathbf{w}_o^T & \mathbf{w}_Z^T\end{bmatrix}\begin{bmatrix}\mathbf{O} \circ \widetilde{\mathbf{v}}_i\\ \mathbf{Z}_{ui}\end{bmatrix} + b\right) \tag{17}$$

$$= \sigma\left(\begin{bmatrix}\mathbf{e}^T & \mathbf{w}_Z^T\end{bmatrix}\begin{bmatrix}\mathbf{O} \circ \widetilde{\mathbf{v}}_i\\ \mathbf{Z}_{ui}\end{bmatrix} + b\right) \tag{18}$$

where $\mathbf{W}_o$ and $\mathbf{W}_Z$ reflect the significance of the two aspects in prediction and $\mathbf{e}$ denotes a unit vector. Equation (18) is obtained since we have $\mathbf{w}_o^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i) = \mathbf{e}^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i \circ \mathbf{w}_o) = \mathbf{e}^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i)$ and parameter $\mathbf{w}_o$ can be absorbed in $\widetilde{\mathbf{v}}_i$. By leaving the "temporal bias" apart, $\mathbf{w}_Z$ is helpful to investigate and interpret the feature-based couplings between users, items and context.

Complexity Analysis: To make a recommendation for each user u at time step t, the proposed model calculates $\hat{y}^t_{u,i}$ for all items $i \notin \mathcal{S}_u$ and recommends the items with the top-k highest predictive probabilities. Hence, the complexity for making recommendation for all users is $O(|\mathcal{U}||\mathcal{I}|P + k|\mathcal{U}||\mathcal{I}|)$, where U is the user set, $\mathcal{I}$ is the item set, k is the number of recommended items, and P denotes the time complexity of calculating $\hat{y}^t_{u,i}$. Since a user's previous action sequence for a given time step is fixed, the output of the convolutional network can be calculated once and used to calculate $\hat{y}^t_{u,i}$ on all items, which means that the complexity of convolutional operations can be ignored relative to the calculation of feature couplings. The complexity of calculating pairwise feature couplings is $O(Md)$ for each item [4] where M denotes

TABLE I
DATA SET STATISTICS

Dataset	Item#	User#	Interaction#	Feature#
MovieLens	3,706	6,040	1,000,209	10,089
Tafeng	23,071	12,889	657,211	38,004
Last.fm	174,903	983	14,248,823	217,321

the number of features and d is the dimension of latent factors. Accordingly, for all users, the total complexity of recommending top-k items is:

$$O(|\mathcal{U}||\mathcal{I}|P + k|\mathcal{U}||\mathcal{I}|) = O(|U||\mathcal{I}|Md + k|\mathcal{U}||\mathcal{I}|).$$

Practically, recommendation is made based on a candidate item set (denoted as $\mathcal{C}$) rather than all items $\mathcal{U}$ to save time while the corresponding time complexity is $O(|U||\mathcal{C}|Md + k|U||\mathcal{C}|)$ and $|\mathcal{C}| \ll |\mathcal{I}|$. Similar to other neural recommenders, TARN can be trained offline, thus we do not investigate its training efficiency in these experiments.

IV. EXPERIMENTS AND EVALUATION

We perform extensive experiments on three public data sets to investigate the following research problems.

- Q1 What is the effect of integrating feature couplings and preference dynamics on TARN-enabled recommendation?
- Q2 How does TARN perform in comparison with the state-of-the-art recommendation methods?
- Q3 How effective is TARN in capturing user preference dynamics compared with the state-of-the-art methods?
- Q4 Does modeling explicit feature couplings enable TARN to handle the cold-start issue?

A. Experimental Settings

1) Data Sets: The effect of our method in capturing user preference dynamics and feature couplings is tested on three publicly available data sets. Table I summarizes the statistics of the data sets after pre-processing them.

a) MovieLens[3]*:* This data set collects the ratings from users who joined MovieLens in 2000. As this work concerns feature couplings, user demographics (i.e., gender, age, and occupation) and item information (i.e., genre) are selected to train the models. We then convert each rating application (i.e., a pair of user ID and movie ID) along with user/item side information and temporal context into multi-hot representations, resulting in $10,089$ features in total. All the ratings in MovieLens are scaled from $\{1, 2, 3, 4, 5\}$ to a target value 1 to indicate a user has rated a movie.

b) Tafeng[4]*:* This data set contains Chinese grocery store transactional data from November 2000 to February 2001. Customer demographics (i.e., customer ID, age, and pin code) and item attributes (i.e., original ID, sub class, amount, asset, and price) are used in the experiments. Numeric attributes (e.g., amount, asset, and price) are discretized, and each transaction (i.e., a pair of customer ID and original ID) along with customer and item information and temporal context is converted to a feature vector, resulting in $38,004$ features in total.

c) Last.fm[5]*:* This data set records the music listening habits of nearly 1,000 users till May, 5th 2009. User information (i.e., user ID, gender, age, and country) and music information (i.e., track ID and artist ID) are used as features. Regarding all the above categorical features, each record is converted to a 217 321-sized feature vector for model training.

Since all the data sets contain only positive instances, to ensure model generalization in such one-class settings, for each positive instance of a user, we randomly sample two negative instances that have no interactions (i.e., no ratings or no purchase) with the user. All the negative samples are assigned with a target value of 0.

2) Evaluation Metrics: To evaluate the performance in the sequence settings, we employ the widely used leave-one-out protocol. Specifically, we hold the latest item (action) in each user's item sequence in the test set and the remaining items in the training set. After a model is trained, we generate, for each user, a personalized ranking of items that have no interaction with the user in the training set, and we evaluate the performance on the recommended ranking in terms of two ranking-based metrics [18], [52]: HR@K and mean average precision (MAP). Given a top-K ranked item set $\hat{R}_K$ and the target ground-truth item set R, HR@K is calculated as:

$$\text{HR@}N = \frac{|R \cap \hat{R}_N|}{|R|}. \tag{19}$$

MAP is calculated w.r.t. the average precision (AP) on all users, and AP is defined by:

$$\text{AP} = \frac{\sum_{i=1}^{|\hat{R}|} \text{PHR@}i \times \text{rel}(i)}{|\hat{R}|}, \tag{20}$$

where $\hat{R}$ denotes the whole recommendation list and $\text{rel}(i)$ equals 1 if $i \in r$, otherwise 0.

3) Baselines: To investigate the design effectiveness of TARN, our model is customized into the following versions.

1) *T-MP:* This replaces the attentive average pooling in TARN with max pooling in order to investigate the effectiveness of the attentive average pooling. Max pooling is performed over the convolutional output of each filter, i.e., each column of $C^{\mathcal{F}^h}$.
2) *T-IF:* This models user preference dynamics over user action sequences by a convolutional network with an identical-width filter in contrast to the multi-filter design in TARN. In the experiments, we fix the width $h = 4$.
3) *T-NT:* This does not involve any temporal context information in modeling user preference dynamics, which verifies the contribution made by the temporal action embedding.

To make a fair comparison, our experiments use the same settings on other parameters of the variants of TARN. Furthermore, we compare TARN with the following baselines.

[3]https://grouplens.org/datasets/movielens/
[4]https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset
[5]https://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/

1) *POP:* This ranks items based on their popularity determined by the number of user interactions on them in the training set. It is a non-personalized baseline.
2) *FM [53]:* This learns both first- and second-order feature relations in the latent feature space.
3) *NFM [7]:* This combines the linearity of FM and the non-linearity of neural networks to jointly model low-order and high-order feature relations and achieves state-of-the-art performance in sparse prediction.
4) *RRN [10]:* This is state-of-the-art rating prediction method with sequential settings by introducing LSTM and a traditional low-rank factorization to capture user preference dynamics and user-item interactions.
5) *Caser [18]:* This is a CNN-based model for sequential recommendation by leveraging a CNN with multiple horizontal convolutional filters and one vertical convolutional filter to capture sequential patterns and combining user embeddings to capture user global preferences.
6) *SASRec [20]:* State-of-the-art self-attention-based recommendation model uses an attention mechanism to capture the long-term semantics of action prediction.
7) *SITAR [42]:* This is the latest context-aware sequential recommendation model by involving the temporal context into stacked RNNs to capture user preference dynamics that vary with contextual dynamics and temporal gaps.
8) *RCNN [19]:* This is the latest RNN- and CNN-based model with RNN to capture complex long-term dependences and CNN to extract short-term sequential patterns on the recurrent hidden states.

These state-of-the-art methods are deliberately chosen for the following considerations: FM and NFM verify the contributions of capturing the dynamics of user preferences; RRN and SITAR are RNN-based methods; and Caser and RCNN are CNN-based methods. These methods are selected to justify the effectiveness of incorporating feature couplings and to compare RNNs and CNNs in capturing temporal dynamics. Furthermore, SASRec is selected to compare with the typical attention-based method. Markov chain-based methods such as in [15], [16] are not compared since the latest deep neural network-based methods outperform these methods [18].

4) Implementation and Parameter Settings: We implement TARN, its variants, RRN and CRNN in TensorFlow. FM is based on LibFM [4]. NFM, Caser, SASRec, and SITAR are derived from their GitHub versions released by their authors. Furthermore, we perform a grid search of the learning rate over $\{1e^{-4}, 1e^{-3}, \ldots, 1e^{-1}\}$ and L_2 regularization rate over $\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$ to select the best performance for FM and NFM and obtain the parameter settings for RRN, Caser, SASRec, SITAR, and RCNN according to their source codes and recommended settings. To tune the hyperparameters in TARN, we leave out the penultimate actions in each user's sequence for validation. Finally, we set 1) the embedding size: 64; 2) convolutional layer: $n_h = 16$ and $H = 4$, meaning for each $h \in [1, 4]$, we have 16 filters; 3) attentive average pooling: $d_a = 32$; 4) optimizer: Adam [54]; 5) batch size: 256; and 6) learning rate: $1e^{-3}$. For a fair comparison and efficiency, all methods are set with an embedding size of 64 if not specified, although a larger embedding size may perform better.

B. Ablation Study to Verify Q1

To analyze the effectiveness of integrating feature couplings with preference dynamics, we report the ablation test of TARN under embedding sizes of $\{8, 16, 32, 64, 128, 256, 512\}$. Specifically, we discard the feature interaction network in Fig. 2 to test the contribution of modeling feature couplings and compare this with FM to test the performance gain derived by capturing user preference dynamics.

We test HR@10 and MAP of these methods w.r.t. different embedding sizes. The results are shown in Fig. 3 where "TARN w/o FI" denotes TARN without the feature interaction network. Obviously, TARN outperforms the other two methods, indicating that both feature couplings and user preference dynamics contribute to the prediction. The two comparative methods have advantages in different data sets. Specifically, TARN w/o FI achieves significant performance improvement with an increase of embedding size and outperforms FM on MovieLens and Last.fm, but it achieves a small improvement on Tafeng. In contrast, FM performs better when embedding size increases and outperforms TARN w/o FI on Tafeng. The results confirm the fact that user preferences for media (e.g., movies and music) are dynamic and subject to the current context, while users in Tafeng have stable actions and preferences for grocery. This shows that simultaneously modeling user preference dynamics and feature couplings are necessary to deeply understand the intrinsic relations between the user, item, and context features and the user action sequential patterns to enable powerful recommendations.

With an increasing embedding size, TARN achieves a more stable performance than the other two methods, and it outperforms the other two methods consistently. The results indicate that: 1) a larger embedding size is beneficial to improve performance due to the increased modeling capability; 2) intrinsically, feature couplings and preference dynamics can work collaboratively and complement each other; and 3) the design of TARN effectively captures and integrates the two aspects for the prediction. Furthermore, the performance worsens when the embedding size becomes large, which is attributed to the fact that complex models easily overfit training data.

C. Performance Comparison to Verify Q2

We investigate the recommendation performance of TARN against the state-of-the-art baselines w.r.t. HR@10 and MAP. The results are summarized in Table II, where the best result in each row is highlighted in bold and the best baseline results are underlined for each data set. Table II enables the following key observations.

First, Table II shows that TARN achieves the best results. Specifically, TARN achieves an improvement of 1.7% and 2.5% in terms of HR@10 and MAP over SASRec on MovieLens where SASRec has the best performance among the state-of-the-art methods, and it also performs better than the baselines on Tafeng; it especially outperforms NFM, the best

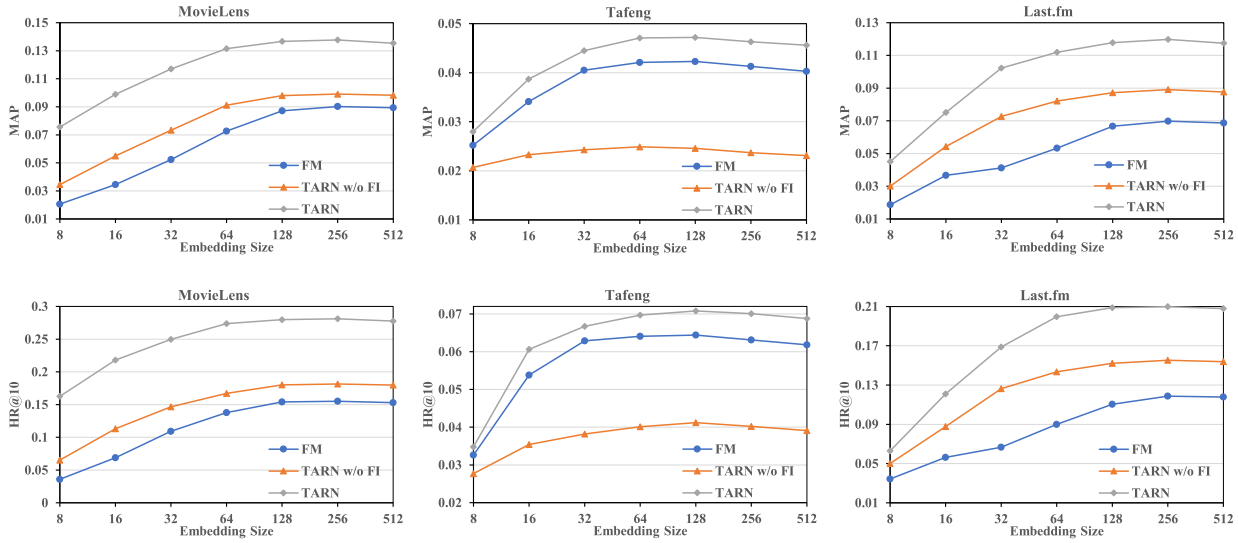

Fig. 3. Ablation test performance: HR@10 and MAP of FM, TARN w/o feature couplings and TARN w.r.t. embedding sizes.

TABLE II
TARN RECOMMENDATION PERFORMANCE COMPARISON WITH BASELINES ON MOVIELENS, TAFENG, AND LAST.FM

Dataset	Metric	POP	FM	NFM	RRN	Caser	SASRec	SITAR	CRNN	T-MP	T-IF	T-NT	TARN
MovieLens	MAP	0.0201	0.0727	0.1028	0.1138	0.1231	0.1283	0.1198	0.1275	0.1269	0.1119	0.126	**0.1315**
	HR@10	0.0363	0.1379	0.1889	0.2402	0.2627	0.2692	0.2531	0.2689	0.2695	0.2397	0.2664	**0.2738**
Tafeng	MAP	0.0284	0.0421	0.0435	0.0334	0.0326	0.0391	0.0407	0.0389	0.0484	0.0437	0.0463	**0.0471**
	HR@10	0.0427	0.0629	0.0647	0.0461	0.0472	0.0566	0.0582	0.0554	0.0694	0.0651	0.0687	**0.0697**
Last.fm	MAP	0.0044	0.0533	0.0672	0.0952	0.0989	0.1029	0.0973	0.1037	0.1052	0.0991	0.1058	**0.1109**
	HR@10	0.0051	0.0899	0.1139	0.1782	0.1818	0.1907	0.1791	0.1911	0.1948	0.185	0.1951	**0.1997**

baseline on Tafeng, by up to 7.7% and 8.3% in terms of HR@10 and MAP. Furthermore, on Last.fm, TARN respectively exhibits an improvement of 4.5% and 6.9% in terms of HR@10 and MAP over the state-of-the-art CRNN. The results indicate that TARN achieves state-of-the-art performance.

Second, the aforementioned result verifies the effectiveness of integrating feature couplings and user preference dynamics. Methods like FM, NFM, SASRec, and CRNN which only involve either of the two aspects cannot fully capture user preferences and degrade the recommendation performance. Similarly, RRN and Caser fail to exploit the available explicit features and do not work well on a data set with strong feature couplings, e.g., Tafeng. SITAR performs well on Tafeng but obtains bad results on the other two, indicating our design of interaction layers and convolutional layers is more effective to capture context influence and sequential patterns.

Furthermore, compared to RNN- and CNN-based methods, the variants of TARN, i.e., T-IF, T-MP, and T-NT, also achieve desirable and even better performance. This improvement is attributed to modeling the coupling relationships between explicit features. The results confirm that explicit features may be coupled to drive user preferences and action on an item [1].

Lastly, Table II shows that TARN works better than T-NT, T-IF, and T-MP as expected, indicating that the proposed temporal action embedding, the setting of multi-width filters, and the attentive average pooling contribute to improving recommendation performance. Specifically, the multi-width filter design facilitates the extraction of different granularities of features and is suitable for capturing multi-fold sequential patterns. The attentive average pooling is beneficial to extract more information and learn a more effective aggregation (i.e., weighted average) in the setting of pooling over the sequence.

D. Influence of Sequence Length to Verify Q3

To further verify TARN's capability in modeling user preference dynamics, we compare RRN, Caser, SASRec, SITAR, CRNN, and TARN under different clipped sequence lengths N. The comparison provides a comprehensive observation of sequential recommendation using these methods. The results in terms of MAP and HR@10 are shown in Fig. 4.

Fig. 4 reports that TARN achieves higher MAP than the baselines over different sequence lengths. This result is partially attributable to the contribution of capturing feature couplings and demonstrates the positive effect of integrating feature couplings and preference dynamics. Except on Tafeng where user preference dynamics are not obvious, the performance of the other methods is strongly influenced by sequence length. Specifically, the best sequence length for the methods are 12 and 20 on MovieLens and Last.fm, respectively (note that RRN is less influenced by sequence length since RRN embeds all previous actions at each step for input to capture global information and is more stable). The results show that the comparative methods achieve their best performance under different sequence lengths on two data sets, which is intuitively reasonable listening to music may occur more frequently than watching movies, and there exist stronger

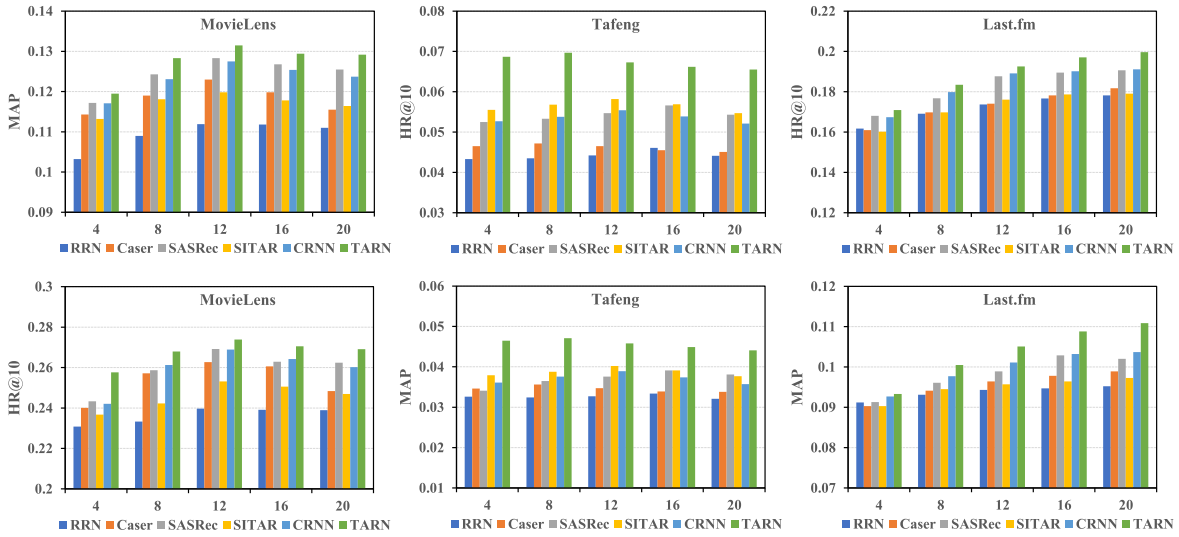

Fig. 4. HR@10 and MAP comparison over different sequence lengths between TARN and the sequential baselines.

long-term dependences within music sequence data. It is worth noting that TARN works much better and more stable than Caser, which demonstrates again that attentive average pooling is effective and more useful to "summarize" the convolutional output. Furthermore, we observe that all methods perform worse on MovieLens when the sequence length becomes larger than 12. This may be because noise (irrelevant actions) is introduced with an increase in sequence length.

E. Cold-Start to Verify Q4

To investigate the way TARN handles cold-start users, we compare TARN to the sequential baselines RRN, Caser, SASRec, SITAR, and CRNN on users with few training samples. The results are shown in Table III which reports the average HR@1 over ten experiments. Tafeng is not included since RRN and Caser perform badly on Tafeng. From Table III, we can make the following observations.

1) TARN achieves obviously higher hit rates (highlighted in bold) over the baselines even for users who have few training samples.
2) SITAR performs slightly better than the other baselines.
3) Caser and RRN are comparable with SASRec and CRNN on users with quite a few samples, and SASRec and CRNN perform worst.

This is because both TARN and SITAR involve context features into the preference modeling, but TARN considers the explicit features, such as user profile and item features and captures the couplings between the user, item, and context features. However, Caser and RRN model user stationary preferences by factorizing only user-item interactions in a latent space as matrix factorization does but do not consider explicit features, and SASRec and CRNN do not even consider user stationary preferences but strongly rely on user action sequences. The results indicate that TARN achieves better performance than state-of-the-art methods in the cold-start settings and demonstrates that modeling explicit feature couplings is beneficial to capture user stationary preferences in predicting user actions.

TABLE III

COLD-START TEST OF TARN OVER THE SEQUENTIAL BASELINES IN TERMS OF HR@1. THE INTERVALS ($[a, b]$) IN THE TABLE HEADER DENOTE THOSE USERS WITH THE NUMBER OF TRAINING SAMPLES IN $[a, b]$

Dataset	Method	$[1, 5]$	$[6, 10]$	$[11, 20]$
MovieLens	RRN	0.0313 ±0.0039	0.0464 ±0.003	0.0481 ±0.0032
	Caser	0.0342 ±0.0032	0.0497 ±0.0028	0.0536 ±0.0029
	SASRec	0.0288 ±0.0021	0.0414 ±0.0029	0.0485 ±0.0035
	SITAR	0.0352 ±0.003	0.0514 ±0.0033	0.0561 ±0.0035
	CRNN	0.0282 ±0.0031	0.0397 ±0.0023	0.0476 ±0.0025
	TARN	**0.0408** ±0.0038	**0.0574** ±0.0039	**0.0615** ±0.0045
Last.fm	RRN	0.0287 ±0.0024	0.0453 ±0.0032	0.0454 ±0.0027
	Caser	0.0302 ±0.0026	0.0449 ±0.0029	0.0461 ±0.0033
	SASRec	0.0264 ±0.0026	0.0418 ±0.0031	0.0443 ±0.0032
	SITAR	0.0311 ±0.0032	0.0464 ±0.0029	0.0491 ±0.0034
	CRNN	0.0252 ±0.0031	0.0437 ±0.0024	0.0456 ±0.0029
	TARN	**0.0408** ±0.0038	**0.0574** ±0.0039	**0.0615** ±0.0045

F. Visualization and Interpretability

In this section, we explore the rationale and interpretability of our model and discuss interesting findings from the model. Due to space limitations, we only analyze the visualization on MovieLens as an example and the results on Tafeng and Last.fm show the same findings.

1) Explicit Feature Couplings: To investigate the predictive effects of couplings between user, item and context features, we design the following experiment: following FM in capturing both first-order and second-order feature couplings as DeepFM [6], NFM [7], and AFM [51], we involve the first-order information into the feature interaction network by

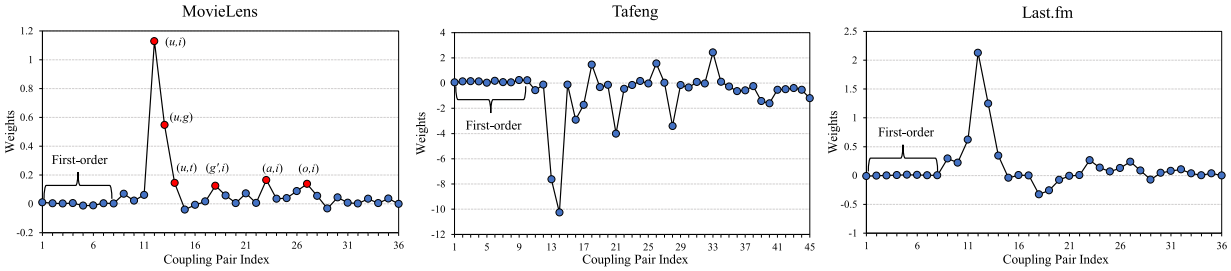

Fig. 5. Visualization of weight for MovieLens, Tafeng, and last.fm. Using MovieLens as an example, the first-order feature couplings are numbered from 1 to 8 and the second-order feature couplings are numbered larger than 8. There are a total of 36 couplings resulting from the eight features in MovieLens.

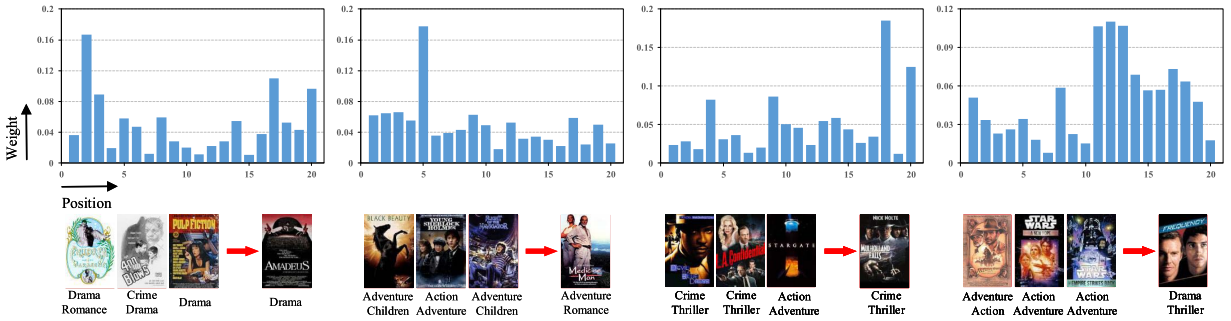

Fig. 6. Illustration of the influence of preference dynamics on prediction on the MovieLens data set. The histograms show the statistics of attention weights on the latest 20 movies of each selected user. The three highest weighted movies (left of the red arrow) and the target movie (right of the red arrow) are listed below which are labeled with their genres.

modifying Eq. (12) as follows:

$$\begin{aligned} \mathbf{Z}^t_{u,i} &= [z_{0,1}, z_{0,2}, \ldots, z_{M-1,M}]^T \\ \text{s.t. } z_{p,q} &= x_p x_q \mathbf{v}_p \cdot \mathbf{v}_q, \quad p < q, \ x_p, x_q \in \widetilde{\mathbf{X}}^t_{u,i}. \end{aligned} \tag{21}$$

Here, $\widetilde{\mathbf{X}}^t_{u,i} = [x_0, x_1, \ldots, 0, x_q, \ldots, x_p, 0, \ldots, x_M]$ denotes the extended feature vector from $\mathbf{X}^t_{u,i}$ where the additional feature $x_0 = 1$ is a constant and the corresponding embedding vector $\mathbf{v}_0 = \mathbf{e}$ is a unit vector. Such a model is trained till convergence on the same parameter settings as TARN, and we visualize the well-trained output weights on the feature coupling vector, i.e., W^Z in (18), to reveal the contribution of first-order and second-order feature couplings in the setting of our model. The absolute values of the resultant weights are high, meaning the corresponding feature couplings have an effect on the prediction.

The weight visualization on MovieLens is shown in Fig. 5 where the y-axis denotes the weight values and the x-axis denotes the coupling pair indices calculated by a mapping function $\Phi : z_{p,q} \mapsto \mathbb{R}$ and $\Phi(z_{p,q}) = p + q$ where $z_{p,q} \in \mathbf{Z}^t_{u,i}$. Fig. 5 shows that the learned weights for all the first-order couplings are around 0 and most of the weights for the second-order couplings are positive. This observation indicates that the first-order couplings adopted in methods like NFM and DeepFM do not contribute to the prediction of user actions in our case. In other words, there is no need to consider the first-order features, and the pairwise feature couplings are suitable and sufficient to capture the coupling relations between user, item, and context features and model user stationary preferences.

In addition, we select six feature pairs (highlighted by the red nodes) whose weights are higher than those of the other pairs. Here, we denote u for user ID, i for item ID, g for genre of items, t for timestamp, g' for gender, a for age, and o for occupation. The six feature pairs are: (user ID, item ID), (user ID, genre), (user ID, time), (gender, item ID), (age, item ID), and (occupation, item ID), which represent inter-feature couplings [27] between users, items and time context features. In contrast, the intra-feature couplings [27] within user features, item features and context features have relatively small weights. These findings indicate that inter-feature couplings between users, items, and context are more productive to the prediction.

In the observation, the coupling pair (user ID and item ID) has the largest weight, which is equivalent to factorizing user-item interactions as MF does. This is reasonable because user ID and item ID are representative and informative. Furthermore, user/item features are often partially available and cannot reflect the comprehensive relations between users and items. In addition, the pairs (user ID, genre) and (user ID and time) have higher weights, indicating that the genre of a movie and the temporal context are influential to user actions as expected. Similarly, the other red nodes reflect that the corresponding feature pairs have an impact on the prediction.

2) Preference Dynamics: To further investigate the intuition of modeling user preference dynamics, Fig. 6 illustrates the influence of previous items on prediction for four users. The figure shows the collected attention weights (y-axis) on each position (corresponding to the x-axis, denoting the position of items in a user's action sequence) collected from TARN

with $H = 1$ and $n_h = 32$ on the MovieLens data set. Note that a higher attention weight on an item reflects the item is more influential to the prediction of the target item. We then present the three highest weighted items and the target item and investigate the influence of user preference dynamics in terms of explicit item feature movie genres.

From Fig. 6, we observe that previous movies are assigned different weights according to the attention mechanism, which is acceptable since user actions are differently influenced by previous actions, e.g., watching series movies. In addition, we observe that the most influential (highest weighted) movies are of similar/same genre as the target movies. The results demonstrate that the learned attention weights are meaningful and explainable and show that users prefer similar movies (e.g., movies of the same genre) in the short term. The 4th selected case is exceptional which is explainable since the user action on movies "Drama" and "Thriller" are determined by user stationary preferences for "Drama" or "Thriller" although the previous action sequence presents an obvious pattern of choosing "Action" and "Adventure" movies.

V. Conclusion

To understand how the explicit features of users, items, and context influence user preferences and how user preference dynamics influence user actions over time, we propose a time-aware recommendation neural network, TARN. TARN learns the couplings between explicit user features, item features, and time-related contextual information by a feature interaction network and models the temporal dynamics of user preferences by a convolutional network. To enhance the model's capability of preference modeling, we further propose a temporal action embedding and attentive average pooling. Extensive experiments verify the advantages of TARN and the effectiveness of integrating feature couplings and preference dynamics in capturing intrinsic driving factors of recommendation. Significantly, TARN provides a way to explain user stationary preferences in terms of explicit feature couplings. The visualization results support the rationale of our design and show interesting findings that inter-feature couplings between users, items, and context are more highly contributive than the intra-feature couplings (i.e., first-order couplings within user features, item features, or context features).

The design of TARN may be further expanded by considering more suitable and explainable methods to learn and integrate user preference dynamics and user stationary preferences. Intuitively, these two aspects of preferences are closely related during the evolution of user actions on items. In addition, further research is expected on the more effective coupling learning of the heterogeneous features of users, items, and context; explainable memory neural networks may be used to model historical user action sequences; and more explorations such as the non-linear integration of user, item and context feature couplings with user preference dynamics may be valuable.

References

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

[2] L. Cao, "Coupling learning of complex interactions," *Inf. Process. Manage.*, vol. 51, no. 2, pp. 167–186, Mar. 2015.

[3] Q. Zhang, L. Cao, C. Zhu, Z. Li, and J. Sun, "CoupledCF: Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3662–3668.

[4] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.

[5] T. D. T. Do and L. Cao, "Metadata-dependent infinite Poisson factorization for efficiently modelling sparse and large matrices in recommendation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 5010–5016.

[6] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1725–1731.

[7] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 355–364.

[8] T. D. T. Do and L. Cao, "Gamma-Poisson dynamic matrix factorization embedded with metadata influence," in *Proc. 32nd Ann. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5829–5840.

[9] Q. Cui, S. Wu, Q. Liu, W. Zhong, and L. Wang, "MV-RNN: A multi-view recurrent neural network for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 2, pp. 317–331, Feb. 2020.

[10] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 495–503.

[11] J. L. Moore, S. Chen, D. Turnbull, and T. Joachims, "Taste over time: The temporal dynamics of user preferences," in *Proc. 14th Int. Soc. Music Inf. Retr. Conf.*, 2013, pp. 401–406.

[12] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[13] R. Warlop, A. Lazaric, and J. Mary, "Fighting boredom in recommender systems with linear reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 1764–1773.

[14] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 271–302, Jun. 2005.

[15] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 811–820.

[16] R. He and J. McAuley, "Fusing similarity models with Markov chains for sparse sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 191–200.

[17] J. Wang and J. Caverlee, "Recurrent recommendation with local coherence," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 564–572.

[18] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 565–573.

[19] C. Xu *et al.*, "Recurrent convolutional neural network for sequential recommendation," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 3398–3404.

[20] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*. Washington, DC, USA: IEEE Computer Society, Nov. 2018, pp. 197–206.

[21] W. Yuan, H. Wang, X. Yu, N. Liu, and Z. Li, "Attention-based context-aware sequential recommendation model," *Inf. Sci.*, vol. 510, pp. 122–134, Feb. 2020.

[22] Q. Zhao *et al.*, "Calendar-aware proactive email recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 655–664.

[23] W. Ye, S. Wang, X. Chen, X. Wang, Z. Qin, and D. Yin, "Time matters: Sequential recommendation with complex temporal information," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1459–1468.

[24] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1746–1751.

[25] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252.

[26] C. Zhu, L. Cao, Q. Liu, J. Yin, and V. Kumar, "Heterogeneous metric learning of categorical data with hierarchical couplings," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1254–1267, Jul. 2018.
[27] C. Zhu, L. Cao, and J. Yin, "Unsupervised heterogeneous coupling learning for categorical representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jul. 21, 2020, doi: 10.1109/TPAMI.2020.3010953.
[28] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, pp. 12:1–12:7.
[29] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 447–456.
[30] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 507–517.
[31] A. Beutel *et al.*, "Latent cross: Making use of context in recurrent recommender systems," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 46–54.
[32] J. Huang, Z. Ren, W. X. Zhao, G. He, J.-R. Wen, and D. Dong, "Taxonomy-aware multi-hop reasoning networks for sequential recommendation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 573–581.
[33] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[34] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 1419–1428.
[35] Q. Zhu, X. Zhou, Z. Song, J. Tan, and L. Guo, "DAN: Deep attention neural network for news recommendation," in *Proc. AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2019, pp. 5973–5980.
[36] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2019, pp. 346–353.
[37] R. Qiu, Z. Huang, J. Li, and H. Yin, "Exploiting cross-session information for session-based recommendation with graph neural networks," *ACM Trans. Inf. Syst.*, vol. 38, no. 3, pp. 22:1–22:23, 2020.
[38] X. Zhou, C. Mascolo, and Z. Zhao, "Topic-enhanced memory networks for personalised point-of-interest recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 3018–3028.
[39] X. Chen *et al.*, "Sequential recommendation with user memory networks," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 108–116.
[40] P. Wang, Y. Fan, L. Xia, W. X. Zhao, S. Niu, and J. Huang, "KERL: A knowledge-guided reinforcement learning model for sequential recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 209–218.
[41] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 6332–6338.
[42] L. Rakkappan and V. Rajan, "Context-aware sequential recommendations withStacked recurrent neural networks," in *Proc. World Wide Web Conf.*, May 2019, pp. 3172–3178.
[43] W.-C. Kang, C. Fang, Z. Wang, and J. McAuley, "Visually-aware fashion recommendation and design with generative image models," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*. Washington, DC, USA: IEEE Computer Society, Nov. 2017, pp. 207–216.
[44] X. Chen *et al.*, "Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 765–774.
[45] W. Wang, W. Wang, Y. Xu, J. Shen, and S.-C. Zhu, "Attentive fashion grammar network for fashion landmark detection and clothing category classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4271–4280.
[46] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 2782–2788.
[47] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 233–240.
[48] Y. Tay, A. T. Luu, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2309–2318.
[49] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.
[50] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.
[51] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proc. 26th Int. J. Conf. Artif. Intell. (IJCAI)*, 2017, pp. 3119–3125.
[52] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King, "STELLAR: Spatial-temporal latent ranking for successive point-of-interest recommendation," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 315–322.
[53] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR (Poster)*, 2015.

Qi Zhang (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science and Technology enrolled in the dual Ph.D. program of the Beijing Institute of Technology, Beijing, China, and the University of Technology Sydney, Sydney, NSW, Australia.

His current research interests focus on collaborative filtering, learning to hash, and sequential recommendation.

Longbing Cao (Senior Member, IEEE) received one Ph.D. degree in pattern recognition and intelligent systems from Chinese Academy of Science in 2002 and another Ph.D. degree in computing science from the University of Technology Sydney in 2005.

He is currently a Professor and an ARC Future Fellow with the University of Technology Sydney, Sydney, NSW, Australia. His research interests include AI, data science, analytics and machine learning, behavior informatics, and their enterprise applications.

Chongyang Shi received the Ph.D. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2010.

He is currently an Associate Professor with the School of Computer Science, BIT. His research areas focus on information retrieval, knowledge engineering, personalized service, and sentiment analysis.

Zhendong Niu received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 1995.

He is currently a Professor and the Deputy Dean of the School of Computer Science and Technology, Beijing Institute of Technology. His current research interests include information retrieval, software architecture, digital libraries, and Web-based learning techniques.