Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction

Xiang Li, Chao Wang, Bin Tong, Jiwei Tan, Xiaoyi Zeng, Tao Zhuang
Alibaba Group, Hangzhou & Beijing, China
{leo.lx,xiaoxuan.wc,tongbin.tb,jiwei.tjw,yuanhan,zhuangtao.zt}@alibaba-inc.com

ABSTRACT

For better user satisfaction and business effectiveness, Click-Through Rate (CTR) prediction is one of the most important tasks in E-commerce. It is often the case that users' interests different from their past routines may emerge or impressions such as promotional items may burst in a very short period. In essence, such changes relate to item evolution problem, which has not been investigated by previous studies. The state-of-the-art methods in the sequential recommendation, which use simple user behaviors, are incapable of modeling these changes sufficiently. It is because, in the user behaviors, outdated interests may exist and the popularity of an item over time is not well represented. To address these limitations, we introduce time-aware item behaviors for addressing the recommendation of emerging preference. The time-aware item behavior for an item is a set of users who interact with this item with timestamps. The rich interaction information of users for an item may help to model its evolution. In this work, we propose a CTR prediction model TIEN based on the time-aware item behavior. In TIEN, by leveraging the interaction time intervals, information of similar users in a short time interval helps identify the emerging user interest of the target user. By using the sequential time intervals, the item's popularity over time can be captured in evolutionary item dynamics. Noisy users who interact with items accidentally are further eliminated thus learning robust personalized item dynamics. To the best of our knowledge, this is the first study to the item evolution problem for E-commerce CTR prediction. We conduct extensive experiments on five real-world CTR prediction datasets. The results show that the TIEN model consistently achieves remarkable improvements to the state-of-the-art methods.

CCS CONCEPTS

• **Information systems → Content ranking**; **Online shopping**; **Recommender systems**; **Content analysis and feature selection**; **Data encoding and canonicalization**.

KEYWORDS

sequential recommendation, information dissemination, recurrent neural network, attention, e-commerce recommendation

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411952

ACM Reference Format:
Xiang Li, Chao Wang, Bin Tong, Jiwei Tan, Xiaoyi Zeng, Tao Zhuang. 2020. Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411952

1 INTRODUCTION

Large E-commerce portals such as Amazon and Taobao are serving hundreds of millions of users nowadays with billions of items. For better user satisfaction and business effectiveness, Click-Through Rate (CTR) prediction has been one of the most important tasks in E-commerce. With the rapid progress of deep neural models, most of advanced CTR prediction models [8, 17, 26, 32–34] achieve great success by leveraging sequential user behaviors. For a given user, the user behavior consists of items with which the user interacts, which allows models to predict his/her potential interest according to the historical behaviors. However, users' preferences may change or emerge rapidly, for example breaking news, tweets or photos may occasionally attract users' interests. There are also cases in E-commerce that a user's interest may drift or some items may burst suddenly. For example, during the shopping festival, such as Amazon's Black Friday and Taobao's Double 11, users' preferences different from their daily routines may emerge, and items in promotions can become popular in a very short period. In such situations, using typical user behavior usually fails to predict users' emerging interests. The reason is not only outdated interests may exist in the user behaviors, but also the prediction of user's emerging preference heavily depends on the evolution of items. This problem is substantial but has rarely been studied by previous works. We name it as item evolution problem. Therefore, in this paper we focus on how to recommend emerging preference in E-commerce CTR prediction.

In order to tackle this problem, we introduce item behaviors to the CTR prediction task besides user behaviors. The item behavior for an item is a set of users who interact with this item. It can be viewed as *global* interactive behavior. In contrast, user behavior is viewed as *local* interactive behavior, since interaction information of the other users is unavailable. Item behavior was proposed in social network recommendation [5, 19, 24, 31] to find a group of users similar to the interacted users in the item behaviors. Item behaviors can be applied to our problem, because the rich interaction information of users may help to model the evolution of an item. Previous works either integrate the item behaviors through a simple aggregation, or take the item behavior as a behavioral sequence to capture the item dynamics in a temporal evolution [29, 30]. However, these methods either treat users in item behaviors equally or preserve only the order of users in an item behavior sequence, which ignores

two possible time-sensitive characteristics in recommending the emerging preferences.

First, the interaction time in item behavior is critical to predicting the potential audience of the item. Users may have interacted with an item far from the current, and now they may have changed their interests. Methods that ignore interaction time in item behaviors may diffuse users' outdated interests to their current similar users. For example, a school uniform was purchased by a girl student a few years ago who may now prefer ladies wear or urban fashion dress due to the drifting of her interests, therefore it is not appropriate to recommend a school uniform for her similar users. Instead, the school uniform should be recommended to similar users of ones who have recently purchased it. Second, the interaction time in item behavior is critical to reflect the items' attractiveness to users varying over time. The items even with same item behaviors can have different popular trends and life cycles. For example, item A and item B have the same interactive users, but the interaction time was one day ago and one year ago, respectively. This implies that item A is a more popular item than item B at this time. Thus, an item clicked or purchased far before the current time may have less impact to the current recommendation. Unfortunately, modeling the item behavior as a sequence simply ignores the interaction timestamps, which results in the same temporal dynamics for the two items. The above limitations suggest that an evolutionary item dynamics is important for the emerging preference recommendation. In this paper, we propose time-aware item behavior to address the limitations. Unlike traditional item behavior, the time-aware item behavior for an item is a set of users who interact with this item with timestamps. The time-aware item behaviors are helpful for recommending the emerging preferences since they may imply items' potential preference evolution and impact over time.

Time-awareness in the item behaviors helps to pay attention to users who interact with a specific item in a recent period. However, the emerging preference recommendation can be affected by noisy interactions. It is because noisy behaviors of a similar user to the target one make the recommendation unreliable. For example, in a very recent period, user A who prefers political news accidentally clicks a piece of entertainment news. User B, who is similar to user A, prefers political news as well. Even if time awareness takes effect, entertainment news will still be unnecessarily recommended to user B due to that both user A and B share the same interest in the recent period. Moreover, Guo et al. [6] pointed out that even the same item should be viewed differently for different users. To achieve this, attention mechanism was used to activate the most similar users in item behaviors concerning certain target users [6, 14]. However, the noisy interactions may result in unreliable personalized item dynamics if considering the most similar users only. In order to obtain robust personalized item dynamics, how to eliminate the noisy users in the time-aware item behaviors remains a challenge.

Based on the above, we propose TIEN (Deep **T**ime-Aware **I**tem **E**volution **N**etwork) to deal with the item evolution problem. In our TIEN, a Time-Interval Attention Layer is proposed to calculate the importance weight for each user in the item behaviors according to the intervals between the historical interaction and the recommendation time. This allows the model to pay attention to valuable users in the item behaviors. Moreover, a Robust Personalized Attention Layer is proposed to eliminate noisy users in the item behaviors thereby achieving robust personalized item dynamics. Last, a Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture the popularity of the items, such that two items with the same item behaviors can be featured in different popularity. Therefore, more accurate recommendation of emerging preferences is achieved by fully leveraging the global item behaviors, that is modelling both personalized item dynamics and popularity varied over time. The contributions of the paper are summarized as:

- To the best of our knowledge, this is the first study to the item evolution problem for E-commerce CTR prediction. To tackle this problem, we introduce time-aware item behavior, an extension of traditional user behavior which has not been used in recommending emerging preference.
- We propose the following two techniques to fully utilize the rich interaction information in the time-aware item behavior. The first is to pay attention to up-to-date interaction in the item behaviors and capture evolutionary item dynamics by using sequential time intervals. The second is to eliminate noisy interaction, which helps realize robust personalized item dynamics.
- We conduct extensive experiments on five real-world CTR prediction datasets. Experimental results verify the proposed model consistently brings remarkable improvements to previous methods and achieves new state-of-the-art results in most the cases.

2 RELATED WORK

2.1 Sequential Recommendations

In recent years, there have been growing numbers of researches on personalized recommendation for music [27], news [18], videos [4], and jobs [2] based on deep neural networks. Personalized search and recommendation is typically based on sequential user behaviors. The RNN model is used in GRU4Rec [8] for inferring users' future intentions based on their historical click behaviors. Several extensions to the RNN model are proposed in [25] for enhancing the performance. In CASER [26], several historical items are regarded as an 'image', and horizontal and vertical convolutional filters are used to capture behavior patterns at different scales. However, these methods can only obtain a fixed trajectory of interest evolution for any candidate item, so they may be disturbed by interest drifting. To resolve the above problem, an attention mechanism is proposed in DIN [34] to capture relative interests to the candidate item and obtain adaptive interest representation. Inspired by Transformer [28] for machine translation, an attention-based framework is invented in ATRANK [32] for the CTR prediction. A two-layer RNN structure with an attention mechanism is proposed in DIEN [33], which uses the attention weights to control the second RNN layer to activate the most relative interests to the candidate item.

Compared to sequence-independent recommendation approaches, such as LR [16] and FM [23], methods based on sequential user behaviors improve the performance of CTR due to users' potential interest being predicted. However, as mentioned in Introduction, personalized recommendation based on the user behavior may fail to predict users' emerging interest, because users' preferences may change or emerge rapidly.

2.2 Information Dissemination

In many social network scenarios, information like news, events, pictures or some fashion elements may spread in a predictable pattern, in which future evolution and potential impact of an item can be predicted with an early-stage observation on it. Item behavior was proposed in social network recommendation [3, 19, 24] to predict users who will be affected in the future, which essentially aims at finding a group of users similar to the interacted users in item behaviors. Early studies [9, 10] use matrix factorization to predict who will be influenced in the future. For further mining the sequential pattern in item behaviors, Du et al. [5] uses RNN model to capture the hidden dynamics in the temporal evolution of the item's affected sequence. By extending this idea, Typo-LSTM [29] uses RNN model to predict the next affected user. Recently, DIB is proposed in [6] to capture personalized item dynamics concerning certain target users through the attention mechanism, so that even the same item may be viewed distinctively for different users. Also, Liu et al. [14] activates users in item behaviors most similar to target users for learning a personalized item representation.

However, previous RNN-based methods ignore timestamps and preserve the order of items only. In other words, these methods implicitly assume that adjacent users in the sequence have the same time intervals. Moreover, previous works, which use the most similar users to learn personalized item representations, may unnecessarily have noisy interactions involved in the item behaviors, which results in an unreliable personalized item dynamics.

3 PRELIMINARY

In a recommender system, the user-item interaction is typically formulated as a matrix $\mathcal{Y} = \{y_{ui}\}_{M\times N}$, where M and N denote the numbers of users and items, respectively. The interaction y is either implicit feedback [1], e.g., click, or explicit user rating [13]. In this work, we focus on the CTR prediction, which implies that the matrix $\mathcal{Y}$ consists of 0 and 1. Specifically, $y_{ui} = 1$ means that u has clicked i, otherwise $y_{ui} = 0$. Moreover, each interaction is associated with a timestamp t that records the time of interaction. Therefore, the data in the recommender system are denoted by a set of quadruplets $\mathcal{T} = \{(u, i, t, y)\}$, each of which includes the user $u \in \mathcal{U}$ interacts with an item $i \in \mathcal{I}$ at a recommendation time t. As to a target triple (u, i, t) for which an interaction probability will be predicted, two historical interaction behaviors can be created for the target user u and candidate item i at the given time t as follows.

Definition 1. (User Behavior): Given a user u, the user behavior is a set of items that the user u interacts in chronological order, as $\mathcal{I}_u^t = \{i \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{i_1, i_2, \cdots\}$.

Definition 2. (Item Behavior): Symmetrically, given an item i, the item behavior is a sequence of users who interacts with the candidate item i, as $\mathcal{U}_i^t = \{u \mid (u, i, t') \in \mathcal{T}, t' < t\} = \{u_1, u_2, \cdots\}$.

In this work, the final CTR prediction aims to estimate the probability of interaction $\hat{y}$ between the target user $u \in \mathcal{U}$ and the candidate item $i \in \mathcal{I}$, with the consideration of both the user's interaction behavior $\mathcal{I}_u$ and the item's interaction behavior $\mathcal{U}_i$, as:

$$\hat{y} = \Pr\left(y \mid u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t\right) = \mathcal{F}\left(u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t; \Theta\right) \tag{1}$$

The notations and the corresponding descriptions have been summarized in Table 1.

Notation	Description
$\mathcal{U}$,$\mathcal{I}$	User and Item set.
u, i, t	The target user and the candidate item at the recommendation time.
$\mathcal{I}_u^t$,$\mathcal{U}_i^t$	Corresponding User behavior and Item behavior of u and i at time t.
$y, \hat{y}$	The indicator and the predicted probability of the user-item interaction.
$\boldsymbol{e}_u, \boldsymbol{e}_i$	Embeddings of a user u and an item i.
$\mathcal{S}_u^t, \mathcal{S}_i^t$	Embeddings of the user behavior $\mathcal{I}_u^t$ and item behavior $\mathcal{U}_i^t$.
$\boldsymbol{e}_{tiv}$	Embeddings of the time interval for each interaction in item behaviors.
$\mathcal{S}_{tiv}^t$	Embeddings of the time interval sequence of item behaviors.
Θ	The parameters of the function $\mathcal{F}()$.

Table 1: Notations and descriptions.

4 PROPOSED MODEL

In this section, we present our model TIEN as shown in Figure 1. TIEN leverages both user behaviors and item behaviors. The item behavior for an item is a set of users who interact with this item. Time-Interval Attention Layer is proposed to calculate the importance weight for each user in the item behaviors according to the intervals between the historical interaction and the recommendation time. This allows the model to pay attention to valuable users in the item behaviors. Then, a Robust Personalized Attention Layer is proposed to eliminate noisy users in the item behaviors thereby achieving the robust personalized item dynamics. Finally, a Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture popularity of the items, such that two items with the same item behaviors can be featured different popularity. Besides the above three components, the Embedding Layer and Downstream Application Network of our model are similar to related CTR prediction models.

4.1 Embedding Layer

To better encode user's and item's features, the embedding layer is applied upon the sparse features to obtain low-dimensional vectors. In our proposed TIEN, there are two kinds of features: user/item feature and user/item sequence feature. The user/item feature is composed of a certain number of fields. For example, an item feature is represented by multiple fields, such as item ID, shop ID and category ID; a user feature represented by user ID, gender, age and so on. Each field is encoded by a one-hot vector. For example, the gender field of user feature is encoded as $[0, 1]$. The one-hot vectors of all the fields in the user and item features are concatenated to obtain multi-hot vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$, respectively. The user/item sequence feature can be represented by collecting user/item features ($\boldsymbol{x}_u$ or $\boldsymbol{x}_i$) into an ordered set.

In the embedding layer, each field of feature is corresponding to one embedding matrix. For example, the embedding matrix of item ID can be represented by $\mathbf{W}_{emb}^{item\_id} \in \mathbf{R}^{d_{item\_id}\times v_{item\_id}}$, where d_{item_id} is the dimension and v_{item_id} is the vocabulary size. We perform embedding lookups to transform the multi-hot vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$ into low-dimensional embeddings $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$, according to:

$$\begin{aligned} \boldsymbol{e}_u &= [\mathbf{W}_{emb_u}^1 \boldsymbol{x}_u^1, \ldots, \mathbf{W}_{emb_u}^F \boldsymbol{x}_u^F], \mathbf{W}_{emb_u}^f \in \mathbf{R}^{d_f \times v_f} \\ \boldsymbol{e}_i &= [\mathbf{W}_{emb_i}^1 \boldsymbol{x}_i^1, \ldots, \mathbf{W}_{emb_i}^F \boldsymbol{x}_i^F], \mathbf{W}_{emb_i}^f \in \mathbf{R}^{d_f \times v_f} \end{aligned} \tag{2}$$

where F is the number of user/item feature, and $\boldsymbol{x}_u^f/\boldsymbol{x}_i^f$ is the f^{th} feature. Then the user behavior embeddings is formed with item embeddings in the sequence, as $\mathcal{S}_u^t = \left\{\boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_{|\mathcal{S}_u|}}\right\}$. Similarly, the item behavior embeddings is formed with user embeddings,

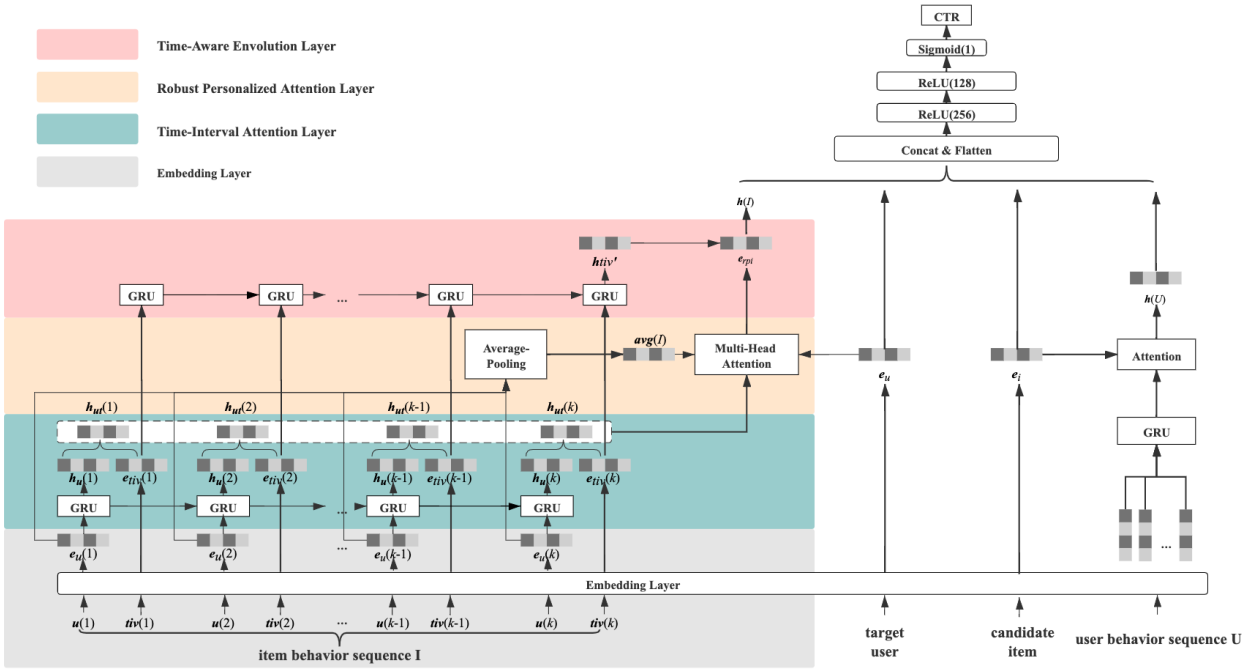


Figure 1: The architecture of TIEN that predicts the probability of a user clicks an item. Besides the target user and candidate item, TIEN uses the information from both user and item behaviors. Embedding Layer transforms user feature $u[k]$ and time interval feature $tiv[k]$ to embedding vectors $\boldsymbol{e}_u[k]$ and $\boldsymbol{e}_{tiv}[k]$, respectively. The Time-Interval Attention Layer leverages the time intervals between the historical interaction and the recommendation time to strength information of certain users in item behaviors. The Robust Personalized Attention Layer uses both the target user and the representative information shared by all users in the item behaviors to build the query of the attention mechanism, thereby eliminating noisy users in the item behaviors. The Time-Aware Evolution Layer models the sequence of interaction time intervals for each item to capture the evolutionary item dynamics. Finally, the evolutionary item dynamics $\boldsymbol{h}(I)$ and embedding vectors of remaining features including user embedding $\boldsymbol{e}_u$, item embedding $\boldsymbol{e}_i$ and user behavior embeddings $\boldsymbol{h}(U)$ are concatenated together and fed into a point-wise fully connected neural network for the CTR prediction.

as $\mathcal{S}_i^t = \left\{\boldsymbol{e}_{u_1}, \cdots, \boldsymbol{e}_{u_{|\mathcal{S}_i|}}\right\}$, where $|\mathcal{S}_u|$ and $|\mathcal{S}_i|$ are the sequence lengths of user and item behaviors, respectively.

4.2 Time-Interval Attention Layer

Item behaviors can be applied to recommend emerging preference in E-commerce CTR prediction, because the rich interaction information of users may help to model the evolution of an item. Previous works either integrate the item behaviors through a simple aggregation, or take the item behaviors as a behavioral sequence to capture the item dynamics in a temporal evolution [29, 30]. However, these methods either treat users in item behaviors equally or preserve only the order of users in an item behavior sequence.

Due to the above reason, the Time-Interval Attention Layer is proposed to calculate the importance weight of each user in the item behaviors using the time intervals between the historical interaction and the recommendation time. Given an item behavior $\mathcal{U}_i^t$ of item i at the timestamp t, each interactive user $u_k \in \mathcal{U}_i^t$ corresponds to a time interval tiv_k, which can be calculated by $tiv_k = |t - t_k|$, where t_k is the interaction timestamp between u_k and i. Since the elapsed time interval tiv follows an exponential distribution, the time intervals are mapped in the ranges $[0, 1), [1, 2), [2, 4), \cdots, [2^j, 2^{j+1})$ to categorical feature $\boldsymbol{x}_{tiv}$ of $0, 1, 2, \cdots, j+1$. We then perform embedding lookups to obtain the time interval embedding $\boldsymbol{e}_{tiv}$ to capture impacts of different time intervals, according to:

$$\boldsymbol{e}_{tiv} = [\mathbf{W}_{emb_{tiv}} \boldsymbol{x}_{tiv}], \mathbf{W}_{emb_{tiv}} \in \mathbf{R}^{d_{tiv} \times v_{tiv}} \tag{3}$$

where d_{tiv} is the dimension and v_{tiv} is the vocabulary size. Then the time interval sequence embedding is formed with each time interval embedding in the item behaviors, as $\mathcal{S}_{tiv}^t = \left\{\boldsymbol{e}_{tiv_1}, \cdots, \boldsymbol{e}_{tiv_{|\mathcal{S}_i|}}\right\}$, where$|\mathcal{S}_i|$ is the sequence length of item behaviors.

In the Time-Interval Attention Layer, we first use GRU to model the dependencies of item behavior embeddings and obtain the behavior hidden state $\boldsymbol{h}_{u_k}$ at each step k. Then, to calculate the importance weight for each user in the item behaviors thus strengthening information of certain users, we build an attention *Query* of the attention mechanism [28] by achieving the time-interval attentive user state. It is produced by the element-wise summation of behavior hidden state and item-interval embedding, according to $\boldsymbol{h}_{ut_k} = \boldsymbol{h}_{u_k} + \boldsymbol{e}_{tiv_k}$.

4.3 Robust Personalized Attention Layer

Time-awareness in the item behaviors helps to pay attention to users who interact with a specific item in a recent period. However, the emerging preference recommendation may be affected by noisy interactions. Moreover, to achieve the personalized item dynamics, attention mechanism was used to activate the most similar users in item behaviors concerning certain target users [6, 14]. However, the noisy interactions may result in an unreliable personalized item dynamics if considering the most similar users only.

Due to the above reason, the Robust Personalized Attention Layer is proposed for eliminating the noisy users in the time-aware item behaviors, thereby improving the robustness of personalized item dynamics. In the Robust Personalized Attention Layer, we represent the user embeddings in an item behavior using average pooling. The average of the user embeddings encodes the most representative information shared by all users in the item behaviors. We use both the target user and the averaged item behavior embeddings to build an attention *Query* of the attention mechanism, according to:

$$\boldsymbol{e}_{u_{smooth}} = \boldsymbol{e}_u + \sum_{k=1}^{K} \boldsymbol{e}_{u_k} \tag{4}$$

where $\boldsymbol{e}_u$ is the embedding of target user, and $u_k \in \mathcal{U}_i^t$. Then, an attentive pooling method is used for generating robust personalized item dynamics according to the target user. It should be noted that Eq. 4 smooths the representation of the target user in an intuitive way, which eliminates the side-effect from the target user. The attention query can be regarded as a trade-off between personalization and noisy elimination.

Inspired by previous works [28, 32], we apply scaled dot-product attention as the attentive pooling method, according to:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \tag{5}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ represent *Query*, *Key* and *Value*, respectively and d_k is the dimension of $\mathbf{K}$. Here, *Query* is the smooth embeddings of the target user $\boldsymbol{e}_{u_{smooth}}$, *Key* is the time-interval attentive user state $\boldsymbol{h}_{ut_k}$, and *Value* is the behavior hidden state $\boldsymbol{h}_{u_k}$. Eq. 5 calculates the weight of each user in the item behaviors by considering both the time interval embedding and the smooth target user embedding, and applies the weight to the hidden state of the item behaviors, thereby achieving robust personalized item dynamics through the weighted summation. Moreover, in order to enable the model to jointly attend to information from different representation subspaces at different positions, we adopts multi-head attention, following [28]. The multi-head attention is defined as follows:

$$\begin{aligned} MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) =& Concat(head_1, head_2, \cdots, head_h)\mathbf{V}^O \\ head_i =& Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{6}$$

where $\mathbf{V}^O$, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are parameters and h is the number of heads. Then, the multi-head attention network employs a feed-forward network for further strengthening the performance. Finally, the robust personalized item dynamics $\boldsymbol{e}_{rpi}$ can be achieved by:

$$\boldsymbol{e}_{rpi} = ReLU((MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V})\mathbf{W}_1 + \boldsymbol{b}_1)\mathbf{W}_2 + \boldsymbol{b}_2 \tag{7}$$

4.4 Time-Aware Evolution Layer

In many scenarios, each item has a different fashion trend and life cycle, which is also the most important characteristic of an item in E-commerce. However, previous methods [6, 14, 29, 30] that model the item behavior as a sequence simply ignore the interaction timestamps of item behaviors, which results in the same temporal dynamics for the two items. Therefore, it is beneficial to capture the evolutionary item dynamics for the emerging preference recommendation.

Due to the above reason, the Time-Aware Evolution Layer is proposed to model the sequence of time intervals in an item behavior to capture the popularity of the items, such that two items with the same item behaviors can be featured in different popularity. We use GRU to learn the relevant time-interval embedding forward step by step to achieve most representative information of the whole time-interval sequence. Each GRU unit takes the corresponding time-interval embedding $\boldsymbol{e}_{tiv_k}$ at each time step and the hidden state from the last time step. We use the output final state as the evolutionary time state of item behaviors, as:

$$\boldsymbol{h}_{tiv}{}' = GRU(\boldsymbol{e}_{tiv_1}, \boldsymbol{e}_{tiv_2}, \cdots, \boldsymbol{e}_{tiv_{|S_i|}}) \tag{8}$$

Finally, the embedding of evolutionary item dynamics $\boldsymbol{h}_{item}$ is obtained by using the combination of evolutionary time state $\boldsymbol{h}_{tiv}{}'$ and robust personalized item dynamics $\boldsymbol{e}_{rpi}$. Note that the combination function can be in a flexible form, such as element-wise summation, multiplication or nonlinear projection.

4.5 Downstream Application Network

The learned evolutionary item dynamics can be applied to various kinds of neural networks according to the downstream task requirement. In this paper, we perform the CTR prediction task and set the downstream application network to be a point-wise fully connected neural network. For this CTR prediction model, besides the information from a target user u and a candidate item i, we use both the user behaviors $\mathcal{I}_u$ and item behaviors $\mathcal{U}_i$ as inputs. A probability is produced that user u clicks item i. The predicted probability of interaction between u and i is calculated as:

$$\hat{y} = \mathcal{F}\left(u, i, \mathcal{I}_u^t, \mathcal{U}_i^t, t; \Theta\right) \tag{9}$$

where $\mathcal{F}$ is a ranking function implemented as a multi-layer deep network with three layers, whose widths are 256, 128 and 1, respectively. The first and second layers use ReLU as activation function while the third layer uses Sigmoid function.

We adopt the widely-used cross entropy [17, 32, 33] as the loss function. The ultimate loss function is formulated as:

$$\arg\min_{\Theta} = -\sum_{k=1}^{N} \left[y_k \log \hat{y}_k + (1 - y_k) \log(1 - \hat{y}_k)\right] \tag{10}$$

where $\boldsymbol{\Theta}$ is the parameters and N is the size of training dataset.

5 EXPERIMENTS

To comprehensively evaluate the proposed model TIEN, we conduct experiments to answer the following research questions:

RQ1 How does TIEN perform, compared to the state-of-the-art models for the CTR prediction task?

RQ2 Does the essential components in TIEN necessarily contribute to the improvement of the performance? Does the incorporation of proposed evolutionary item dynamics contribute to the performance of the state-of-the-art CTR prediction models based on the sequential user behaviors?
RQ3 How do different truncation length K of item behaviors in TIEN and other baselines affect the performances? Is long-term item behaviors well handled by all baselines?
RQ4 How do hyper-parameters in TIEN affect the performance?

5.1 Experimental Setups

5.1.1 Datasets. We evaluate the state-of-the-art methods over popular real-world datasets of the Amazon product data[1] [7, 15].
Amazon Dataset. The dataset contains the temporal users' purchasing behaviors ranging from the May 1996 to July 2014, with 24 categories of different sizes. We choose five categories including Clothing, Beauty, Grocery, Phones and Sports to conduct our experiments. Each category is a set of tuples, and each tuple consists of the information of user, item, rating and timestamp. Since we focus on the CTR prediction tasks, we treat all existing ratings as positive interactions with a label of 1. Following other works [32–34], we filter out users and items whose interactions are less than five. The statistics of each dataset is shown in Table 2.

Datasets	# Users	# Items	# Category	# Interactions	Density	# samples
Clothing	39388	23034	720	278677	0.0003	478580
Beauty	22364	12102	221	198502	0.0007	352278
Grocery	14682	8714	129	151254	0.0011	273146
Phones	27880	10430	51	194439	0.0006	333120
Sports	35599	18358	1073	296337	0.0004	521478

Table 2: Statistics of the datasets.

Dataset Preprocessing. We sort the interactions of each user and items by the timestamp to build the user behaviors and item behaviors. Assuming there are user behaviors $\{i_1, \ldots, i_{j-1}, i_j, \ldots, i_{|\mathcal{S}_u|}\}$ of the user u, where $|\mathcal{S}_u|$ is the sequence length. Let u be the target user and i_j be the candidate item, which is the j^{th} behavior of u. For the item i_j, there exist item behaviors $\{u_1, \ldots, u_{t-1}, u_t, \ldots, u_{|\mathcal{S}_i|}\}$, where $|\mathcal{S}_i|$ is the sequence length. Let u_t be the target user u who interacted with item i_j at timestamp t. In the CTR prediction task, we predict the user response probability between u and i_j according to the first $j-1$ user behaviors, i.e., $\{i_1, \ldots, i_{j-1}\}$, and the item behaviors before the timestamp t, i.e., $\{u_1, \ldots, u_{t-1}\}$. Note that, in order to build the negative samples, 50% candidate items at the prediction time in each dataset have been replaced with another item from the non-clicked item set for each user. This replacement is often used in related works [32–34]. The sample statistics of each dataset is shown in Table 2.
Training & Test Splitting. Following works [17, 21], we split the entire dataset into the training and test parts according to the timestamp of the prediction behavior, effectively avoiding feature leakage. We set a cut time within the time range covered by the full dataset. If the prediction behavior of a sequence took place before the cut time, the sequence is put into the training set. Otherwise it would be put into the test set. In this way, the training set is about 85% of the whole dataset and the left 15% of data is the test set.

[1] http://jmcauley.ucsd.edu/data/amazon/

5.1.2 Evaluation Metrics. We adopt three widely used metrics for the CTR prediction task [17, 20, 22, 32–34], which show a model's performance from different perspectives. The first metric is area under ROC curve (**AUC**) which assesses the pairwise ranking performance of the classification results between the clicked and non-clicked samples. AUC is formulated as:

$$\text{AUC} = \frac{1}{\left|U^{Test}\right|} \sum_{u \in U^{Test}} \frac{1}{|I_u^+|\,|I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \delta(\hat{p_{u,i}} > \hat{p_{u,j}}) \quad (11)$$

where $\hat{p_{u,i}}$ is the predicted probability that a user $u \in U^{Test}$ may click on the item i in the test set and $\delta(\cdot)$ is the indicator function. I_u^+ is the item set clicked by the user, and I_u^- is the item set that the user does not click. The second metric is **Logloss** (cross entropy), which is to measure the overall likelihood of the whole test data. Different from AUC which measures a pairwise relation, Logloss measures predicted probability that can be used to estimate the benefit of a ranking strategy. For example, in the Cost-Per-Click (CPC) advertising system, advertisements are ranked by the effective Cost Per Mille (eCPM), which is the product of the bid price and CTR. The last metric is **F1-score** which is the harmonic mean of precision and recall. It reaches its best value with 1, which means perfect precision and recall, and with 0 at the worst case.

5.1.3 Baselines. To illustrate the effectiveness of TIEN, we compare it with the state-of-the-art methods, which can be grouped into non-sequential and sequential categories.

For non-sequential methods, three models are compared. For these three models, we label them as Group 1.

SVD++ [12] is a hybrid method of latent factor model and neighbor based model. It regards all the sequential behaviors as a whole and ignores the temporal dynamics.
YoutubeNet [4] is proposed to recommend videos in YouTube, which gets user representations by simply averaging the item embeddings in the user behavior sequence.
PNN [20] uses a product layer to capture interactive patterns between inter-field categories on the basis of **YoutubeNet**.

For sequential methods, five models that use user behaviors only are compared. For these five models, we label them as Group 2.

GRU4Rec [8] uses RNN and it is the first work using recurrent cell to model sequential user behaviors.
CASER [26] is a CNN-based model, using horizontal and vertical convolutional filters to capture behavior patterns at different scales.
ATRANK [32] is an attention-based framework modeling the dependencies between user behaviors.
DIEN [33] is a two-layer RNN structure with an attention mechanism. It uses the calculated attention values to control the second RNN layer to activate the most relative interests to the candidate item.
UB-GRUA is a sub-model of **TIEN**, which uses GRU to model the dependencies of user behaviors and uses multi-head attention to activate the most interest hidden state with respect to the candidate item.

For sequential methods, another three models are also compared, which use both user and item behaviors to capture the temporal dependencies in user's states and item's states. Note that these

Group	Method	Clothing			Beauty			Grocery			Phones			Sports		
		AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
1	SVD++	0.7332	0.6271	0.6964	0.7410	0.6343	0.7112	0.6379	0.6992	0.6707	0.7014	0.6664	0.6758	0.7842	0.5988	0.7445
	YoutubeNet	0.7363	0.6061	0.6763	0.7081	0.6773	0.6472	0.5805	0.9637	0.5761	0.6367	0.7110	0.5881	0.7717	0.5758	0.7133
	PNN	0.7435	0.5972	0.6656	0.7838	0.6598	0.7140	0.6364	0.8058	0.6071	0.7094	0.7765	0.6495	0.7967	0.5514	0.7346
2	GRU4Rec	0.7554	0.5885	0.6764	0.7973	0.6418	0.7179	0.6134	0.9850	0.6007	0.6980	0.8135	0.6342	0.8021	0.5492	0.7329
	ATRANK	0.7521	0.5883	0.6858	0.7834	0.6744	0.7129	0.6562	0.8152	0.5732	0.6824	0.7757	0.6222	0.8026	0.5451	0.7301
	CASER	0.7504	0.5935	0.6803	0.7930	0.7068	0.7170	0.6183	0.9499	0.5031	0.7222	0.7638	0.6507	0.7989	0.5489	0.7347
	DIEN	0.7564	0.5876	0.6792	0.7933	0.6966	0.7047	0.6402	0.8579	0.6027	0.7005	0.8276	0.6420	0.8054	0.5427	0.7318
	UB-GRUA	0.7590	0.5828	0.6791	0.7975	0.6498	0.7154	0.6442	0.8078	0.5413	0.6982	0.8248	0.6378	0.8039	0.5466	0.7328
3	Topo-LSTM	0.7682	0.5748	0.6740	0.8458	0.5111	0.7657	0.7927	0.5299	0.7313	0.8089	0.5287	0.7392	0.8156	0.5267	0.7523
	DIB	0.7577	0.5820	0.6816	0.8464	0.4848	0.7680	0.7717	0.5751	0.7114	0.7949	0.6797	0.7317	0.8120	0.5316	0.7474
	IB-GRUA	0.7657	0.5763	0.6817	0.8478	0.4842	0.7703	0.7762	0.5768	0.7036	0.8043	0.6360	0.7401	0.8139	0.5275	0.7492
	TIEN	**0.7962***	**0.5476***	**0.6980***	**0.8701***	**0.4479***	**0.7840***	**0.8252***	**0.5019***	**0.7524***	**0.8389***	**0.4949***	**0.7427***	**0.8266***	**0.5101***	**0.7543***

Table 3: Performance comparisons of TIEN with baseline models on AUC, Logloss and F1-score metrics. The bold value marks the best one in one column, while the underlined value corresponds to the best one among all baselines in each group. Here, * indicates statistical significance improvement compared to the best baseline measured by t-test at p-value of 0.05.

methods model user behaviors in the same way as TIEN. For these three models, we label them as Group 3.

Topo-LSTM [29] considers the item behaviors as a sequence, and adopts LSTM to capture the item dynamics.

DIB [6] proposes a dynamic item block, which activates users in item behaviors most similar to target users for learning personalized item dynamics.

IB-GRUA integrates the technique in Topo-LSTM and DIB, which models the dependencies of sequential item behaviors and further learns personalized item dynamics.

TIEN is our proposed model described in Sec. 4.

5.1.4 Reproducibility. All datasets and code of our approach can be found here[2]. We implement TIEN with Tensorflow 1.4. The code of ATRANK[3], CASER[4], DIEN[5], DIB[6], and Typo-LSTM[7] are from the authors. For GRU4Rec[8] and SVD++[9], we also use code from public resources. The code of YoutubeNet and PNN are from DIEN[10]. All neural network models were trained using NVIDIA Tesla P4 GPU with 8 GB memory.

Parameter Settings. For our TIEN and baseline methods, all hyper-parameters are tuned on the validation dataset, where early stopping strategy is applied such that the training is terminated if validation performance does not improve over 10 iterations. Batch size is fixed on 128 for all methods. The dimensions of the item ID, user ID and category ID are set to 32, which refers to the settings in the work [33]. Adam [11] is used as the optimizer with the learning rate of 0.001. Since we mainly discuss sequential models based on user and item behaviors in this paper, the hidden units of the prediction layer is fixed for all models with [256, 128]. For all methods, the truncation length of user behaviors is 100. For dual behavior models, the truncation length K of item behaviors is chosen from $\{5, 10, 20, 30, 40, 50\}$. We report the results of each method under its optimal hyper-parameters settings.

Significance Test. The experiments are repeated 10 times by changing the random seed for TIEN and the baseline models. The two-tailed pairwise t-test is performed to detect significant differences between TIEN and the baseline models.

5.2 Experimental Results: RQ1

All experiments are repeated 10 times and averaged metrics are reported in Table 3. The influence of random initialization to results is less than 0.0002. From Table 3, we can see that TIEN improves the performance significantly against all the baselines and achieves the state-of-the-art performance.

In most cases, the models in Group 1, which do not consider the user behaviors as a sequence, have degraded performances compared with models in Group 2 and 3. Moreover, in all cases, models in Group 3 outperform models in Group 2, which verifies the effectiveness of both user- and item-side information.

Compared with models in Group 3, i.e., Topo-LSTM, DIB and IB-GRUA, which use both user behaviors and item behaviors, our TIEN shows its superior performance. The possible reasons are shown below: Topo-LSTM can capture the item dynamics in a temporal evolution, but the item dynamics is fixed for all users, lacking personalization; DIB activates the most similar users in item behaviors concerning certain target users for learning personalized item representations, but it ignores the dependency between adjacent item behaviors; Though IBGRUA allows the integration of sequential and personalized characteristics of item behaviors, it does not take the crucial time-sensitive characteristics of item behaviors into consideration and thus cannot capture the evolutionary item dynamics. Moreover, it cannot eliminate the noisy users in the item behaviors to achieve robust personalized item dynamics.

Finally, the improvement of TIEN over the best baseline model are 2.80%, 2.23%, 3.25%, 3.00% and 1.10% AUC scores. As reported in [34], compared with YoutubeNet, DIN improves AUC scores by 1.13% and the improvement of online CTR is 10.0%, which means a small improvement in offline AUC is likely to lead to a significant increase in online CTR. In our practice, for cutting-edge CTR prediction models, even one point of AUC improvement is substantial and achieves significant online promotion.

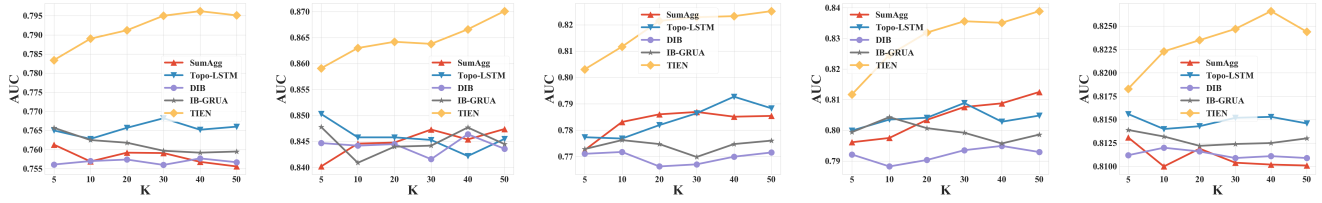
[2]https://github.com/itemevolutionnet/ItemEvolutionNet
[3]https://github.com/jinze1994/ATRank
[4]https://github.com/graytowne/caser_pytorch
[5]https://github.com/mouna99/dien
[6]https://github.com/ouououououou/DIB-PEB-Sequential-RS
[7]https://github.com/vwz/topolstm/
[8]https://github.com/Songweiping/GRU4Rec_TensorFlow
[9]https://github.com/WindQAQ/tf-recsys
[10]https://github.com/mouna99/dien

Method	Clothes			Beauty			Grocery			Phones			Sports		
	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
BaseModel	0.7590	0.5828	0.6791	0.7975	0.6498	0.7154	0.6442	0.8078	0.5413	0.6982	0.8248	0.6378	0.8039	0.5466	0.7328
BaseModel+SumAGG	0.7613↑	0.5794	0.6787	0.8474↑	0.4932	0.7687	0.7869↑	0.5787	0.7123	0.8125↑	0.6100	0.7416	0.8131↑	0.5274	0.7469
BaseModel+TAL	0.7824↑	0.5613	0.6801	0.8499↑	0.4792	0.7557	0.8026↑	0.5225	0.7294	0.8303↑	0.5070	0.7318	0.8203↑	0.5189	0.7490
BaseModel+TAL+RAL	0.7864↑	0.5589	0.6798	0.8598↑	0.4656	0.7790	0.8040↑	0.5241	0.7206	0.8301↓	0.5083	0.7349	0.8200↓	0.5215	0.7420
TIEN (TAL+RAL+TEL)	**0.7962**	**0.5476**	**0.6980**	**0.8701**	**0.4479**	**0.7840**	**0.8252**	**0.5019**	**0.7524**	**0.8389**	**0.4949**	**0.7427**	**0.8266**	**0.5101**	**0.7543**

Table 4: Ablation study of TIEN on AUC, Logloss and F1-score metrics on Amazon datasets. ↑ and ↓ indicates whether adding key components of TIEN boost or degrade the AUC performance.

Model	Clothing			Beauty			Grocery			Phones			Sports		
	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score	AUC	Logloss	F1-score
GRU4Rec	0.7554	0.5885	0.6764	0.7973	0.6418	0.7179	0.6134	0.9850	0.6007	0.6980	0.8135	0.6342	0.8021	0.5492	0.7329
GRU4Rec+TIEN	0.7966	0.5528	0.6811	0.8668	0.4537	0.7686	0.8261	0.5040	0.7357	0.8378	0.4986	0.7322	0.8266	0.5113	0.7562
ATRANK	0.7521	0.5883	0.6858	0.7834	0.6744	0.7129	0.6562	0.8152	0.5732	0.6824	0.7757	0.6222	0.8026	0.5451	0.7301
ATRANK +TIEN	0.7902	0.5619	0.6668	0.8675	0.4498	0.7737	0.8246	0.5037	0.7437	0.8307	0.5110	0.7222	0.8234	0.5154	0.7406
CASER	0.7504	0.5935	0.6803	0.7930	0.7068	0.7170	0.6183	0.9499	0.5031	0.7222	0.7638	0.6507	0.7989	0.5489	0.7347
CASER+TIEN	0.7920	0.5486	0.7089	0.8638	0.4581	0.7653	0.8235	0.5059	0.7445	0.8307	0.5110	0.7222	0.8206	0.5217	0.7366
DIEN	0.7564	0.5876	0.6792	0.7933	0.6966	0.7047	0.6402	0.8579	0.6027	0.7005	0.8276	0.6420	0.8054	0.5427	0.7318
DIEN +TIEN	0.7956	0.5516	0.6851	0.8675	0.4523	0.7846	0.8273	0.5023	0.7400	0.8379	0.4974	0.7399	0.8255	0.5115	0.7526

Table 5: Practicability Study of TIEN.



(a) Amazon Clothing (b) Amazon Beauty (c) Amazon Grocery (d) Amazon Phones (e) Amazon Sports

Figure 2: Performance comparisons of TIEN with baseline models on AUC with respect to different item behavior length K.

5.3 Ablation Study and Practicability Study: RQ2

In this section, we design ablation experiments to study how each component in TIEN contributes to the final performance.

BaseModel: A sub-model of TIEN that uses user behaviors only.

BaseModel+SumAGG: A simple aggregation model which uses sum-pooling to integrate the item behavior embeddings.

BaseModel+TAL: A sub-model of TIEN with the Time-Interval Attention Layer (TAL).

BaseModel+TAL+RAL: A sub-model of TIEN with both TAL and Robust Personalized Attention Layer (RAL).

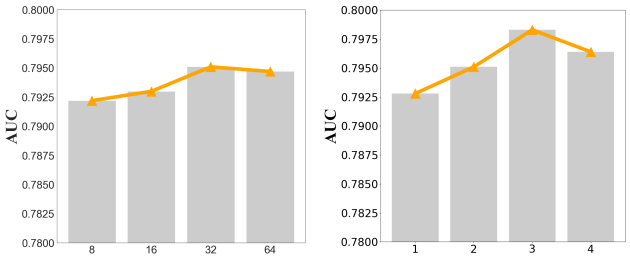
TIEN: The entire proposed model that includes TAL, RAL and Time-Aware Evolution Layer (TEL).

As shown in Table 4, removing any component in TIEN leads to a drop in performance. Moreover, some further observations can be made. First, by leveraging item behaviors, SumAGG alleviates the emerging preference issues and outperforms the baseline model. Second, TAL outperforms SumAGG about 2.11%, 0.25%, 1.57%, 1.78% and 0.72% AUC scores. It shows that leveraging the time intervals between the historical interaction and the recommendation time to strengthen information of certain users in an item behavior can lead to better performance. Third, in many cases, RAL further improves performance, which implies that eliminating the side effects of target users by smoothing the target user's representation can effectively improve the robustness of personalized item dynamics. Last, TEL contributes to the improvements of 0.98%, 1.03%, 1.20%, 0.88%, and 0.66% in AUC. It shows that modeling the sequence of time intervals for each item to capture the evolutionary item dynamics can effectively resolve the item evolution problem.

To verify the utility of evolutionary item dynamics proposed by TIEN, we select several models using user behaviors as base models, including GRU4Rec, ATRANK, CASER, and DIEN. As shown in Table 5, the performance of all based models are improved with the help of the generated evolutionary item dynamics, which proves the effectiveness of the evolutionary item dynamics and shows the practicability of TIEN. Thus, our TIEN can be viewed as a general framework to enhance existing user behavior based CTR prediction models.

5.4 Influence of Truncation Length of Item Behaviors: RQ4

The truncation length K of item behaviors is changed in a range of $\{5, 10, 20, 30, 40, 50\}$ for further investigating the performance of our TIEN and other dual behavior baselines. The results on five Amazon datasets are shown in Figure 2. It can be observed that TIEN improves the performance of the AUC when K increases. In



(a) dimension (b) hidden layer

Figure 3: Performance of TIEN on Amazon Clothing dataset with respect to different hyper-parameters.

most cases, the absolute improvement of the AUC scores is greater than 1%. This implies that TIEN is capable of handling long-term item behavior thereby achieving remarkable improvements. On the contrary, the performances of the baseline models decrease when K increases. This indicates that, without the time-interval attention layer and robust personalized layer, the baseline models cannot handle long-term item behaviors since noisy and useless information is unnecessarily introduced.

5.5 Parameter Sensitivity: RQ4

Impact of Embedding Size: We let the embedding size in TIEN varies in a range of $\{8, 16, 32, 64\}$. The results on Amazon Clothing datasets are shown in Figure 3(a). It can be observed that the AUC scores improve at first and decays latter. The possible reason is that small embedding size may deprive the model of enough expressiveness, and too large embedding size would make the representation vector too sparse, which leads to performance decline.

Impact of Network Depth: We let the number of fully connected layers follow the settings: 1 layer-$[256]$; 2 layers-$[256, 128]$; 3 layers-$[512, 256, 128]$; 4 layers-$[1024, 512, 256, 128]$. The results on Amazon Clothing dataset are shown in Figure 3(b). It is observed that increasing the number of fully connected layers can improve the AUC in the beginning, but the benefit diminishes when more layers are added. Adding more layers may even result in slight performance degradation, possibly due to more model parameters and increased difficulty of training deeper neural networks.

6 CONCLUSION

In this paper, we study how to recommend emerging preferences in E-commerce CTR prediction. We define it as the item evolution problem that has rarely been studied by previous works. In order to tackle this problem, we propose time-aware item behaviors for the CTR prediction task. The time-awareness helps to represent the drift of user interest and the item popularity over time. With the time-aware item behaviors, we propose TIEN (Deep **T**ime-Aware **I**tem **E**volution **N**etwork). Time-Interval Attention Layer is used to identify the emerging user interest by leveraging the interaction time intervals. This allows the model to pay attention to the information of similar users in a short time interval. By using the sequential time intervals, Time-Aware Evolution Layer captures the item's popularity over time, which is represented in evolutionary item dynamics. Robust Personalized Attention Layer helps to eliminate the noisy interaction in the item behaviors, thereby producing robust personalized item dynamics. The extensive experiments on the real-world datasets show the superiority of TIEN over the state-of-the-art methods.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2009. Spatio-temporal models for estimating click-through rate. In *WWW*. ACM, 21–30.
[2] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. 2017. LiJAR: A system for job application redistribution towards efficient career marketplace. In *SIGKDD*. 1397–1406.
[3] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *WWW*. 925–936.
[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Recsys*. 191–198.
[5] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 1555–1564.
[6] Guibing Guo, Shichang Ouyang, Xiaodong He, Fajie Yuan, and Xiaohua Liu. 2019. Dynamic item block and prediction enhancing block for sequential recommendation. In *IJCAI*. AAAI Press, 1373–1379.
[7] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.
[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).
[9] Bo Jiang, Jiguang Liang, Ying Sha, Rui Li, Wei Liu, Hongyuan Ma, and Lihong Wang. 2016. Retweeting behavior prediction based on one-class collaborative filtering in social networks. In *SIGIR*. 977–980.
[10] Bo Jiang, Jiguang Liang, Ying Sha, and Lihong Wang. 2015. Message clustering based matrix factorization model for retweeting behavior prediction. In *CIKM*. 1843–1846.
[11] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[12] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.
[13] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. ACM, 447–456.
[14] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. 2019. Real-time Attention Based Look-alike Model for Recommender System. In *KDD*. ACM.
[15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.
[16] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *KDD*. 1222–1230.
[17] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *KDD*. 596–605.
[18] Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. 2014. Personalized news recommendation using classified keywords to capture user preference. In *ICACT*. 1283–1287.
[19] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2011. RT to Win! Predicting Message Propagation in Twitter. In *ICWSM*.
[20] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *ICDM*. IEEE, 1149–1154.
[21] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*. 565–574.
[22] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *CIKM*. ACM, 679–688.
[23] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
[24] Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H Chi. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *ICSC*. IEEE, 177–184.
[25] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Recsys*. 17–22.
[26] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.
[27] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[29] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological recurrent neural network for diffusion prediction. In *ICDM*. IEEE, 475–484.
[30] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. ACM, 495–503.
[31] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *KDD*. ACM, 1513–1522.
[32] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *AAAI*.
[33] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.
[34] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. ACM, 1059–1068.