DMFP: A Dynamic Multi-Faceted Fine-Grained Preference Model for Recommendation

Huizhao Wang[1], Guanfen Liu[2], Yan Zhao[1], Bolong Zheng[3], Pengpeng Zhao[1], and Kai Zheng[4*]
[1]*School of Computer Science and Technology, Soochow University, Suzhou, China*
[2]*Department of Computing, Macquarie University, NSW, Australia*
[3]*School of Computer Science and Technology, Huazhong University of Science and Technology, China*
[4]*University of Electronic Science and Technology of China, Chengdu, China*
guanfeng.liu@mq.edu.au; {zhaoyan, ppzhao}@suda.edu.cn; bolongzheng@hust.edu.cn; zhengkai@uestc.edu.cn

***Abstract*—The time signals behind a user's historical behaviors are important for better inferring what she prefers to interact with at the next time. For the attention-based recommendation methods, relative position encoding and time intervals division are two common ways to model the time signal behind each behavior. They either only consider the relative position of each behavior in the behavior sequence, or process the continuous temporal features into discrete category features for subsequent tasks, which can hardly capture the dynamic preferences of a user. In addition, although the existing recommendation methods have considered both long-term preference and short-term preference, they ignore the fact that the long-term preference of a user may be multi-faceted, and it is difficult to learn a user's fine-grained short-term preference. In this paper, we propose a Dynamic Multi-faceted Fine-grained Preference model (DMFP), where the multi-hops attention mechanism and the feature-level attention mechanism together with a vertical convolution operation are adopted to capture users' multi-faceted long-term preference and fine-grained short-term preference, respectively. Therefore, DMFP can better support the next-item recommendation. Extensive experiments on three real-world datasets illustrate that our model can improve the effectiveness of the recommendation compared with the state-of-the-art methods.**

I. Introduction

The task that how to build a better recommendation system can be turned into that how to characterize and understand users' interests or needs more accurately. In real life, a user's preference is intrinsically dynamic and evolving. In addition, intuitively, the next item or action more likely depends on the items or actions the user engaged recently [1], [2]. As the historical behaviors of a user can naturally form a sequence over the timeline, the approaches based on Recurrent Neural Network (RNN) or Convolutional Neural Networks (CNN) are the most common ways to capture users' dynamic preferences [2]–[8]. However, limited by the recursive structure, RNN-based methods usually have expensive time cost for offline training and online inference. Besides, CNN-based methods usually overemphasize the interaction between several contiguous behaviors, which makes it hard for them to capture the long-term dependencies [9].

Recently, inspired by the great success of the attention mechanism in image captioning [10] and reading comprehension [11], some studies focus on how to effectively integrate the attention mechanism into recommendations [1], [9], [12].

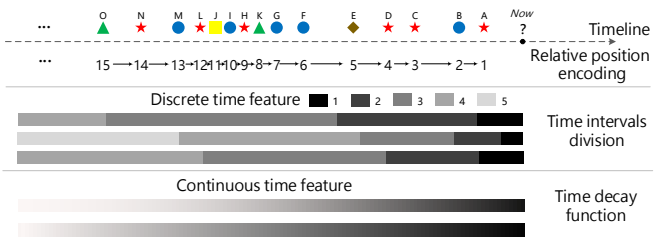

Fig. 1. A schematic diagram of the difference ways of modeling the time signals behind a user's historical behaviors. A basic shape (e.g., circle, triangle, square, diamond and pentagram) represents a historical behavior of the corresponding user. The all historical behaviors of a specific user can be lined up sequentially along the timeline naturally. In addition, the farther the two shapes are apart, the greater the time difference between the two corresponding behaviors.

These attention-based methods are more applicable in parallel processing compared to RNN-based methods, which reduces the time cost of offline training and online prediction. More importantly, the attention mechanism can maintain a variable-length memory, which provides an opportunity to improve the performance of recommendations.

A. *Limitations of the Existing Works*

*1) **Difficult to model the time signals behind users' behaviors**:* As shown in Fig 1, for the attention-based recommendation methods, there are two common ways to model the time signals behind users' behaviors: (1) relative position encoding, and (2) time interval division. The relative position encoding only considers the relative orders like the first, the second and so on [13], [14]; and the time interval division first slices the continuous time signal into intervals of different sizes, then each interval is mapped to a unique discrete value [9], [15]. Compared to the relative position encoding, this method is more in line with the law of the forgetting in human Brain. However, the size of each interval is related to the experimentally selected dataset, which requires a wealth of experience and increases the difficulty of processing time information.

*2) **Difficult to capture the multi-faceted long-term preference and fine-grained short-term preference**:* To characterize and understand users' interests or needs more accurately,

finding a balance between long-term preference and short-term preference is considered as a feasible solution [2], [15], [16].

The long-term preference. The long-term preference refers to a user's general "tastes" hidden in her whole historical behaviors. Caser [2] and HARSAM [15] learn a static embedding for each user to represent her long-term preference. Here, this long-term preference is considered static, because once a recommendation model is trained, the long-term preference embedding will not change before the next retraining, even if some new behaviors occur. Different from these methods described above, SHAN [1] believes learning a static long-term preference representation for each user cannot fully express the dynamics of long-term preference preference, and thus reconstructing long-term user representations from the up-to-date historical behaviors set is more reasonable. However, SHAN ignores a key point, that is, a user may have multi-faceted preferences, which should be treated as a learning goal and explicitly expressed in the loss function. Observing Fig. 1, intuitively, this user is likely to have a long-lasting interest for the two different types of things represented by blue circles and red pentagrams.

The short-term preference. The short-term preference refers to the current interests of a user, which can be reflected by the user's latest L behaviors. Compared to the long-term preference, the short-term preference tends to show a user's demands at the next time. Thus, for better recommendations, a fine-grained analysis of the user's recent behaviors is crucial. SHAN [1] and ATEM [12] adopt an attention mechanism to learn a context embedding that intensifies relevant items but downplays those irrelevant items to the next choice. Specifically, this principle can be formulated as $\vec{\boldsymbol{C}} = \sum_{i=1}^{n} a_i \vec{\boldsymbol{v_i}}$, where $\vec{\boldsymbol{C}}$ is the context embedding that represents the interests of the corresponding user, $\vec{\boldsymbol{v_i}}$ is a vector which can be understood as an abstract representation of one recent behavior, a_i is a scalar where the larger the value, the better the corresponding behavior can reflect the user's short-term preference. However, in the above process, each abstract feature in $\vec{\boldsymbol{v_i}}$ multiplies by the same value of a_i, which makes it hard to preserve only the highly related aspects/features of a historical behavior while discarding those unrelated parts. In other words, many of the existing methods are not good at capturing the fine-grained short-term preference of a user.

B. Contributions of Our Work

To address the limitations of the existing works, we propose a Dynamic Multi-faceted Fine-grained Preference model, called DMFP, for the next-item recommendation. Firstly, this model abandons the customary way of converting continuous time signals into discrete features in many attention-based recommendation models, and adopts a time decay function to preserve the temporal information behind each historical behavior. Secondly, in order to more accurately characterize users' preference, our model extends the existing methods from two different perspectives, i.e., the long-term preference and the short-term preference, respectively. Specifically, inspired by the work in sentence embedding [17], a multi-hops attention mechanism is adopted to mine a user's multi-faceted long-term preference, where a penalization term is designed to encourage to learn different interests of a user. Furthermore, to better understand the users' short-term preference, we model the latest L behaviors of a user in two different ways. One is the natural extension of additive attention [18], [19] at the feature level, the other is the vertical convolution operation [2], [20]. These can capture the relationships among the same dimensional entries of the recent behaviors.

Compared to the existing works, our method offers several distinct advantages:

- DMFP adopts a time decay function to model the time signal of each behavior, which is flexible for a variety of scenarios.
- DMFP encourages to capture the multi-faceted long-term preference and fine-grained short-term preference at the same time, which can learn more accurate preference representation for each user and eventually lead to better recommendations.
- Extensive experiments on three real-world datasets illustrate that DMFP outperforms the state-of-the-art methods in terms of Area Under Curve (AUC), Recall, and Normalized Discounted Cumulative Gain (NDCG).

II. Related Work

Sequential Recommendation: Sequential recommendation predicts the next item that a user is likely to be interested in, based on the user's historical behaviors. Recurrent Neural Networks (RNN) together with its variants LSTM and GRU have been widely applied in sequential recommendation, including session-based GRU [3], dynamic recurrent model [16], and hierarchical personalized RNN model [4]. These RNN-based methods encode historical interaction records into a latent state vector representing the preferences of a user. Although the state vector is able to capture sequential patterns, it still suffers from several issues. For example, it can hardly to be paralleled, and has low efficiency. In addition, it can hardly to preserve long-term dependencies, and emphasize the impact of the recent behaviors excessively.

Inspired by the capability of extracting local features and good efficiency, CNN has been used in sequential recommendation. Similar to the sentence classification [21], Caser [2] uses the 1-D convolution layer and the max-over-time pooling layer to encode historical interactions into a vector to represent the preferences of a user. However, for CNN, the fixed-size encoding vector may not support both short and long sequences well.

Attention and Self-Attention: Recently, attention has been widely used in, such as machine translation task [18], and reading comprehension [11], [22], as it can preserve the highly related elements by assigning different weights for each element in a sequence. For the next-item recommendation, the attention-based transaction embedding model (ATEM) [12] can learn an attentive context embedding that intensifies relevant items but downplays those irrelevant ones to the next choice. Different from attention, self-attention considers the

inner-relations of a sequence, and thus can learn the internal dependencies between elements of a sequence. Following the structure of Transformer [23], ATRank [9] transforms the interaction sequence into a new sequence via self-attention, and has achieved good performance in the next-item recommendation.

III. Problem Formulation

We first define notations used throughout the paper, and then formalize the problem to be addressed.

Notations. Let $\boldsymbol{U} = \{u_1, u_2, \ldots, u_{|\boldsymbol{U}|}\}$ denote the set of users and $\boldsymbol{I} = \{i_1, i_2, \ldots, i_{|I|}\}$ denote the set of items, where $|\boldsymbol{U}|$ and $|\boldsymbol{I}|$ denote the number of elements in the set of User $\boldsymbol{U}$ and Item $\boldsymbol{I}$ respectively. Our task focuses on personalized recommendation, so we concern whether a user $u \in \boldsymbol{U}$ interacts with an item $i \in \boldsymbol{I}$ at time t. Hence each historical behavior can be formulated as a triple ${i_p}^{(u)} = \langle u, i, t\rangle$, where p is the relative position of this behavior in all historical behaviors of user u. By sorting the historical behaviors of user u in ascending order according to the corresponding time signal of each behavior, a *historical behaviors sequence* can be formed, denoted as $\boldsymbol{S_u} = ({i_1}^{(u)}, {i_2}^{(u)}, ...{i_n}^{(u)})$, where n is the length of the *historical behaviors sequence*. In addition, taking the latest L ones from $\boldsymbol{S_u}$, namely $\widetilde{\boldsymbol{S_u}} = ({i_{n-L+1}}^{(u)}, {i_{n-L+2}}^{(u)}, ...{i_n}^{(u)})$, it can reflect user u's short-term preference, which is an important factor for predicting the next item that user u will interact with.

Problem Definition. The task of personalized recommendation aims to output the k items from a candidate set based on their probabilities that a user will interact with at the next time. Formally, the problem can be defined as follows:

Input: The *historical behaviors sequence*s of all users $\boldsymbol{S} = \{\boldsymbol{S_1}, \boldsymbol{S_2}, ...\boldsymbol{S}_{|\boldsymbol{U}|}\}$.

Output: A personalized recommendation model, denoted as $\boldsymbol{f_{rec}}$, which can output the k items that the corresponding user is most likely to interact with at the next time based on a user's *historical behaviors sequence* $\boldsymbol{S_u}$.

IV. Proposed Methodology

The overall architecture of DMFP is shown in Fig. 2, it consists of three major parts: (1) *Long-Term Preference*, (2) *Short-Term Preference*, and (3) *Downstream Application Network*. In addition, each major part contains a number of layers. Firstly, we introduce two important components, i.e., items embedding and time decay function, which are shared by *Long-Term Preference* and *Short-Term Preference*.

Items Embedding. The items that a user interacts with are usually identified by some unique digital IDs, whereas these original IDs have a very limited representation capacity. Therefore, our model first converts each ID to a one-hot encoding vector. Specifically, for the item (ID $= i$), only the unit at position i is set to 1 and all others are set to 0. Then our model employs a fully-connected layer to embed the one-hot encoding vector into a continuous low-dimensional space, which is more informative. Formally, let $\boldsymbol{V} \in \mathbb{R}^{d_e \times |\boldsymbol{I}|}$ be the weight matrix of the fully-connected layer, where d_e is the dimensionality of the latent embedding spaces, and the i^{th} column of the weight matrix encodes that item (ID $= i$) to the real-valued embedding $\boldsymbol{V_{:,i}}$.

Time Decay Function. Each behavior may have different action time, which is an important continuous feature and provides a good opportunity to understand how a user's interests drifts over time. Intuitively, the closer the behavior occurs from the current time point, the more it reflects the user's current preference. In order to integrate the time effect into our recommendation model, we design a time decay function f_{time}, as E.q.(1).

$$f_{time}^{\lambda}(t_{now}, t_i) = e^{-\lambda|t_{now} - t_i|} (\lambda \geq 0) \tag{1}$$

where t_{now} is the current time point, t_i is the time point at which the i^{th} behavior in a *historical behavior sequence* occurs, and thus $f_{time}(t_{now}, t_i)$ is a scalar between 0 and 1. In addition, λ is the time decay factor, and its value is different in different systems. If the users' interest of a system change frequently, it should take a larger value of λ, and vice versa.

A. *Multi-faceted long-term preference*

*1) **Long-Term Behaviors Embedding With Time Decay Layer**:* A user's long-term preference can be reflected by her whole historical behaviors, $\boldsymbol{S_u} = ({i_1}^{(u)}, {i_2}^{(u)}, ...{i_n}^{(u)})$. Hence, for a specific user u, we can encode the historical behavior ${i_p}^{(u)} = \langle u, i, t\rangle$ as Eq. (2).

$$\boldsymbol{h_p} = f_{time}^{\lambda_1}(t_{now}, t) \times \boldsymbol{V_{:,i}} \tag{2}$$

where $\boldsymbol{h_p} \in \mathbb{R}^{d_e \times 1}$ and λ_1 is the time decay factor for long-term preference.

Assuming that user u has n historical behaviors, then her *historical behavior sequence* can be encoded as 2-D matrix $\boldsymbol{H} \in \mathbb{R}^{n \times d_e}$, namely:

$$\boldsymbol{H} = [\boldsymbol{h_1}, \boldsymbol{h_2}, \cdots \boldsymbol{h_n}]^{\boldsymbol{T}} \tag{3}$$

*2) **Multi-hops Attention Layer**:* After gaining the embedding of each historical behavior, we hope to get a user's long-term preference from the *historical behavior sequence* embedding matrix $\boldsymbol{H}$. The items that provide more information about users' preference or more relevant to users' next choice should be given larger weights. Thus, the additive attention mechanism can be adopted, like ATEM [12] and SHAN [1]. Specifically, the attention mechanism takes the *historical behavior sequence* embedding matrix $\boldsymbol{H}$ as input, and outputs a vector of weights $\boldsymbol{a} \in \mathbb{R}^{1 \times n}$:

$$e\left(h_j\right) = \boldsymbol{w_{s2}} ReLU\left(\boldsymbol{W_{s1}} \boldsymbol{h_j}^{\boldsymbol{T}}\right) \tag{4}$$

$$\alpha_j = \frac{exp\left(e\left(h_j\right)\right)}{\sum_{i=1}^{n} exp\left(e\left(h_i\right)\right)} \tag{5}$$

$$\boldsymbol{a} = [\alpha_1, \alpha_2, \cdots \alpha_n] \tag{6}$$

where $j \in \{1, 2, ..., n\}$, $\boldsymbol{W_{s1}} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w_{s2}} \in \mathbb{R}^{1 \times d_e}$, $ReLU()$ is an activation function and $e\left(u_j\right)$ is a scalar. In addition, E.q.(5) is also known as the $softmax()$, and it ensures the sum of the weight α_j equals 1.

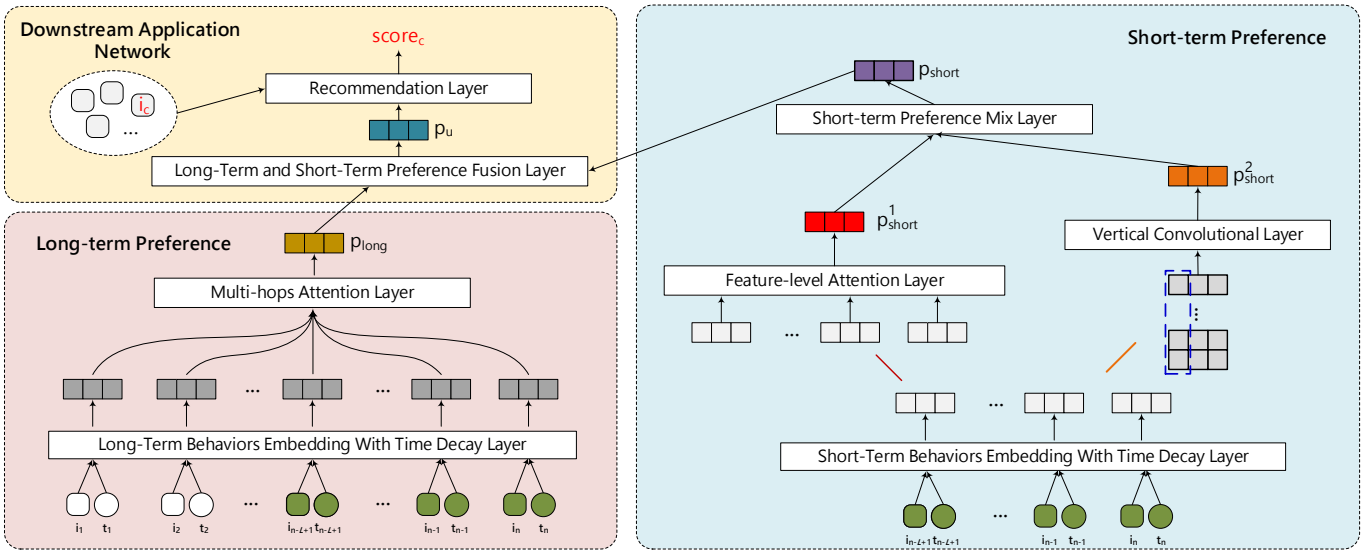

Fig. 2. The overall architecture of DMFP

Then, a vector $\boldsymbol{m} \in \mathbb{R}^{d_e \times 1}$ can be seen as an abstract representation of the user's long-term preference. It can be obtained based on the *historical behavior sequence* embedding matrix $\boldsymbol{H}$ and the attention weight vector $\boldsymbol{a}$. Specifically, this process can be formulated as Eq. (7).

$$\boldsymbol{m} = \sum_{j=1}^{n} \alpha_j \boldsymbol{h_j} \tag{7}$$

However, a user's interests may be multi-faceted. In other words, there may be more than one type of things the user likes, for example, a user may like both love movies and science fiction movies. Inspired by the work of the sentence embedding [17], to represent the overall long-term preference of a user, we perform multiple hops of the attention to get multiple $\boldsymbol{m}$'s. Namely, we will extract r different preferences from the *historical behavior sequence*. Therefore, the $\boldsymbol{w_{s2}} \in \mathbb{R}^{1 \times d_e}$ is extended into $\boldsymbol{W_{s2}} \in \mathbb{R}^{r \times d_e}$, and the weight vector $\boldsymbol{a} \in \mathbb{R}^{1 \times n}$ becomes a weight matrix $\boldsymbol{A} \in \mathbb{R}^{r \times n}$. For simplicity, the above process is formalized as

$$\boldsymbol{A} = softmax\left(\boldsymbol{W_{s2}} ReLU\left(\boldsymbol{W_{s1}} \boldsymbol{H^T}\right)\right) \tag{8}$$

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{H} = [\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots \boldsymbol{m_r}]^{\boldsymbol{T}} \tag{9}$$

Here, the $softmax()$ is performed along the second dimension of its input. As a result of the multi-hops attention, the original vector $\boldsymbol{m} \in \mathbb{R}^{d_e \times 1}$ becomes a matrix $\boldsymbol{M} \in \mathbb{R}^{r \times d_e}$, which can reflect the user's r different long-term preferences.

Finally, like AttRec [24], we take the mean embedding $\boldsymbol{p_{long}} \in \mathbb{R}^{d_e \times 1}$ of the r row vectors in matrix $\boldsymbol{M}$ as an abstract representation of the user's long-term preference.

$$\boldsymbol{p_{long}} = \frac{1}{r} \sum_{t=1}^{r} \boldsymbol{m_t} \tag{10}$$

Penalization Term: The preference matrix $\boldsymbol{M}$ can suffer from redundancy problems, if the attention mechanism always provides similar weights vector a for all the r hops. Thus, we introduce a penalization term to encourage the diversity of weight vectors across different hops of attention, which is equivalent to promoting the multi-faceted preferences. Specifically, like the sentence embedding [17], we use the dot product of $\boldsymbol{A}$ and its transpose, subtracted by an identity matrix, as the measure of redundancy.

$$P = \left\| \boldsymbol{A}\boldsymbol{A^T} - \boldsymbol{I} \right\|_F^2 \tag{11}$$

$$\|\boldsymbol{X}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} {X_{i,j}}^2} \tag{12}$$

where $\|\bullet\|_F$ means the Frobenius norm of a matrix. Similar to the L2 regularization term, the penalization term P will be multiplied by a coefficient, and we minimize it together with the original loss.

Considering that $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$ are any two different weight vectors in $\boldsymbol{A}$, and the sum of all entries within any one weight vector in $\boldsymbol{A}$ is 1, then any non-diagonal elements $a_{ij}(i \neq j)$ in matrix $\boldsymbol{A}\boldsymbol{A^T}$ corresponds to a summation over element-wise product of the two weight vectors:

$$0 < a_{ij} = \sum_{k=1}^{d_e} a_i^k a_j^k < 1 \tag{13}$$

where a_i^k and a_j^k are the k^{th} element in the $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$, respectively.

In the extreme case, there is no overlapping between the two weight vectors $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$, the correspond a_{ij} will be 0. Otherwise, it will have a positive value. On the other extreme case, if the two weight vector $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$ are identical and only one unit is 1 while all others are 0, a_{ij} will have a maximum value of 1. An identity matrix $\boldsymbol{I}$ is subtracted from $\boldsymbol{A}\boldsymbol{A^T}$, so that the diagonal elements and no-diagonal elements of $\boldsymbol{A}\boldsymbol{A^T}$

are forced to approximate to 1 and 0 respectively, where the former encourages each weight vector to focus on different historical behaviors, the latter punishes redundancy between different weight vectors, thus it can learn the multi-faceted long-term/general preference.

B. *Fine-grained short-term preference*

As discussed in Section I, a user's recent behaviors are more representative of the user's current interests. Therefore, we need to analyze the recent behaviors of a user to obtain the fine-grained short-term preference, thus ultimately leads to better recommendations.

*1) **Short-Term behaviors Embedding With Time Decay Layer**:* Following Caser [2], in our model, the latest L behaviors $\widetilde{\boldsymbol{S_u}} = (i_{n-L+1}{}^{(u)}, i_{n-L}{}^{(u)}, ...i_n{}^{(u)})$ of user u are selected from her *historical behavior sequence* $\boldsymbol{S_u}$. Then, like the long term behaviors, with a time decay factor λ_2, the short term behaviors $\widetilde{\boldsymbol{S_u}}$ can be encoded as a 2-D matrix $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{L \times d_e}$, namely:

$$\widetilde{\boldsymbol{H}} = \left[\widetilde{\boldsymbol{h_1}}, \widetilde{\boldsymbol{h_2}}, \cdots \widetilde{\boldsymbol{h_L}}\right]^{\boldsymbol{T}} \tag{14}$$

where $\widetilde{\boldsymbol{h_1}} \in \mathbb{R}^{d_e \times 1}$, and it should be noted that λ_1 and λ_2 used to calculate $\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}$ can be different.

*2) **Feature-Level Self-Attention Layer**:* The existing attention based recommendation methods calculate a scalar weight for each historical behavior of a user, and the items that are more relevant to the next choice are given larger weights. However, the user preferences obtained by the above methods are coarse-grained, as they do not consider the fact that a user may have different preferences for each part of an item. For example, we might score 10 points for a movie's special effects, but only 5 points for its story line. Thus, to capture more fine-grained short-term preference, we adopt a feature-level self-attention mechanism that can calculate a scalar weight for each abstract feature of each item. Then the method can only preserve the highly related aspects of a historical behavior while discarding those unrelated parts.

Suppose $\widetilde{\boldsymbol{h_i}}$ and $\widetilde{\boldsymbol{h_j}}$ are two behaviors embedding vectors in matrix $\widetilde{\boldsymbol{H}}$ respectively, so the attention score of vector $\boldsymbol{f}\left(\widetilde{\boldsymbol{h_i}}, \widetilde{\boldsymbol{h_j}}\right) \in \mathbb{R}^{d_e \times 1}$ between $\widetilde{\boldsymbol{h_i}}$ and $\widetilde{\boldsymbol{h_j}}$ is:

$$\begin{aligned}&\boldsymbol{f}\left(\widetilde{\boldsymbol{h_i}}, \widetilde{\boldsymbol{h_j}}\right) = \\ &\boldsymbol{W_{s5}} \tanh(\boldsymbol{W_{s4}}\widetilde{\boldsymbol{h_i}} + \boldsymbol{W_{s3}}\widetilde{\boldsymbol{h_j}} + \boldsymbol{b_{s2}}) + \boldsymbol{b_{s1}}\end{aligned} \tag{15}$$

where all the parameter matrices $(\boldsymbol{W_{s3}}, \boldsymbol{W_{s4}}, \boldsymbol{W_{s5}}) \in \mathbb{R}^{d_e \times d_e}$, the two bias terms $(\boldsymbol{b_{s1}}, \boldsymbol{b_{s2}}) \in \mathbb{R}^{d_e \times 1}$, and $\tanh()$ is a nonlinear activation function.

Then, the alignment vector $\boldsymbol{\beta_{ij}}$ between $\widetilde{\boldsymbol{h_i}}$ and $\widetilde{\boldsymbol{h_j}}$ can be computed by normalizing each dimension as shown in E.q.(16) and E.q.(17).

$$[\beta_{ij}]_k = \frac{exp\left(\left[\boldsymbol{f}\left(\widetilde{\boldsymbol{h_i}}, \widetilde{\boldsymbol{h_j}}\right)\right]_k\right)}{\sum_{t=1}^{L} exp\left(\left[\boldsymbol{f}\left(\widetilde{\boldsymbol{h_i}}, \widetilde{\boldsymbol{h_j}}\right)\right]_k\right)} \tag{16}$$

$$\boldsymbol{\beta_{ij}} = [[\beta_{ij}]_1, [\beta_{ij}]_2, \cdots [\beta_{ij}]_{d_e}]^{\boldsymbol{T}} \tag{17}$$

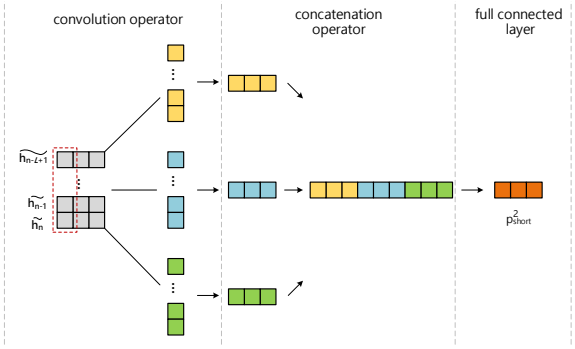

Fig. 3. A schematic diagram of process involved in vertical convolutional layer

where $k \in \{1, 2, \cdots d_e\}$, $[\boldsymbol{x}]_k$ indexes the k^{th} dimension of the vector $\boldsymbol{x}$, and a large $[\beta_{ij}]_k$ means that the k^{th} abstract feature of behavior embedding $\widetilde{\boldsymbol{h_i}}$ is strongly relevant with the behavior embedding $\widetilde{\boldsymbol{h_j}}$. Finally, the output of this feature-level self-attention mechanism is still a matrix and its shape is consistent with the corresponding input $\widetilde{\boldsymbol{H}}$, denoted as $\widetilde{\boldsymbol{M}} \in \mathbb{R}^{L \times d_e}$:

$$\widetilde{\boldsymbol{m_i}} = \sum_{j=1}^{L} \boldsymbol{\beta_{ij}} \circ \widetilde{\boldsymbol{h_j}} \tag{18}$$

$$\widetilde{\boldsymbol{M}} = [\widetilde{\boldsymbol{m_1}}, \widetilde{\boldsymbol{m_2}}, \cdots \widetilde{\boldsymbol{m_L}}]^{\boldsymbol{T}} \tag{19}$$

Here, following the definition of Hadamard product[1], we use "$\circ$" to represent the element-wise product between two vectors with the same shape.

In addition, like the long-term preference, in order to learn a single embedding $\boldsymbol{p^1_{short}} \in \mathbb{R}^{d_e \times 1}$ to represent a user's short-term preference, an average operator is executed on the L rows of the matrix $\widetilde{\boldsymbol{M}}$.

$$\boldsymbol{p^1_{short}} = \frac{1}{L}\sum_{k=1}^{L} \widetilde{\boldsymbol{m_k}} \tag{20}$$

*3) **Vertical Convolutional Layer**:* Inspired by the work of the knowledge graph embedding [20], to further capture a user's short-term preferences, we perform a vertical convolution operation on matrix $\widetilde{\boldsymbol{H}}$. Suppose that a filter $\boldsymbol{\omega} \in \mathbb{R}^{L \times 1}$ is repeatedly operated over each column of $\widetilde{\boldsymbol{H}}$ to finally generate a feature map $\boldsymbol{v} = [v_1, v_2, \cdots v_{d_e}] \in \mathbb{R}^{1 \times d_e}$ as:

$$v_i = ReLU(\sum_{t=1}^{L} \omega_t \widetilde{H}_{t,i} + b) \tag{21}$$

where ω_t indexes the t^{th} dimension of the filter vector $\boldsymbol{\omega}$, $\widetilde{H}_{t,i}$ denotes the element of the t^{th} row and the i^{th} column of the matrix $\widetilde{\boldsymbol{H}}$, and $b \in \mathbb{R}$ is a bias term.

Our model uses different filters $\boldsymbol{\omega}$'s to generate different feature maps $\boldsymbol{v}$'s. Specifically, let Ω and τ denote the set of filters and the number of filters, respectively, i.e. $\tau = |\Omega|$, resulting in τ feature maps. In addition, in order to facilitate

[1]https://en.wikipedia.org/wiki/Hadamard_product_(matrices)

the subsequent operation, these τ feature maps are concatenated into a single vector $\in \mathbb{R}^{1\times\tau d_e}$ which then is converted into a vector $\boldsymbol{p^2_{short}} \in \mathbb{R}^{1\times d_e}$ by a full-connected layer. Fig. 3 illustrates the above computation process in the vertical convolutional layer. Formally, this process can be defined as follow:

$$\boldsymbol{p^2_{short}} = ReLU(concat(ReLU(\widetilde{\boldsymbol{H}} * \Omega))\boldsymbol{W_{s5}} + \boldsymbol{b_{s2}}) \quad (22)$$

where $*$ and $concat$ denote a convolution operator and a concatenation operator respectively, and $\boldsymbol{W_{s5}} \in \mathbb{R}^{\tau d_e \times d_e}$ is the weight matrix in a full-connected layer. Finally, the vector $\boldsymbol{p^2_{short}}$ can be considered as an abstract representation of a user's short-term preference at the feature level.

To sum up, the vertical convolution layer is used to exam the global relationships between the same dimensional entries of the latest L behaviors $\widetilde{\boldsymbol{S_u}}$ of a user, which can capture the fine-grained short-term preference.

*4) **Short-Term Preference Mixed Layer**:* Two abstract representations of a user's short-term preference have been obtained by the *Feature-Level Self-Attention Layer* and *Vertical Convolutional Layer* respectively. However, there may be overlapping between $\boldsymbol{p^1_{short}}$ and $\boldsymbol{p^2_{short}}$. Thus, we first concatenate the two abstract representations of a user's short-term preference, and then feed the result into a fully-connected neural network layer to get a more high-level and abstract feature vector $\boldsymbol{p_{short}} \in \mathbb{R}^{d_e \times 1}$, namely:

$$\boldsymbol{p_{short}} = ReLU(\boldsymbol{W_{s6}}\, concat([\boldsymbol{p^1_{short}}, \boldsymbol{p^2_{short}}^{\boldsymbol{T}}]) + \boldsymbol{b_{s3}}) \quad (23)$$

where $\boldsymbol{W_{s6}} \in \mathbb{R}^{d_e \times 2d_e}$ is the weight matrix in the full-connected layer, $\boldsymbol{b_{s3}} \in \mathbb{R}^{d_e \times 1}$ is the bias term, and $\boldsymbol{p_{short}}$ is viewed as a user's fine-grained short-term preference.

C. Downstream Application Network

*1) **Long-Term and Short-Term Preference Fusion Layer**:* After obtaining a user's long-term preference vector $\boldsymbol{p_{long}}$ and short-term preference vector $\boldsymbol{p_{short}}$, unlike SHAN [1] which gives a scalar weight for the long-term preference manually, in our model, a dimension-wise fusion gate is used to accomplish the combination of $\boldsymbol{p_{long}}$ and $\boldsymbol{p_{short}}$, which can learn the nonlinear relationship between the two types of preferences. Formally,

$$\boldsymbol{F} = \sigma(\boldsymbol{W_{s7}}\, \boldsymbol{p_{long}} + \boldsymbol{W_{s8}}\, \boldsymbol{p_{short}} + \boldsymbol{b_{s4}}) \quad (24)$$

$$\boldsymbol{p_u} = \boldsymbol{F} \circ \boldsymbol{p_{long}} + (\boldsymbol{1} - \boldsymbol{F}) \circ \boldsymbol{p_{short}} \quad (25)$$

where $\boldsymbol{p_u} \in \mathbb{R}^{d_e \times 1}$, $\boldsymbol{W_{s7}}, \boldsymbol{W_{s8}} \in \mathbb{R}^{d_e \times d_e}$ and $\boldsymbol{b_{s4}} \in \mathbb{R}^{d_e \times 1}$ are the learnable parameters of the fusion gate, σ denotes the sigmoid function $f(x) = 1/(1+e^{-x})$, and like E.q.(18), "$\circ$" refers to the element-wise product between two vectors with the same shape. As a result, a user's holistic preference, including long-term and short-term preferences, can be represented by the vector $\boldsymbol{p_u}$.

*2) **Recommendation Layer**:* For the personalized recommendation task, our goal is to sort the candidate set $\boldsymbol{C_u}$ according to the similarity between the user's interests and the corresponding item. The similarity is usually expressed by a score of scalar, and the higher the score, the higher the similarity [1], [9]. Specifically, we employ the inner product to compute the corresponding preference score $score_c$ of the candidate item (ID = c) $\in \boldsymbol{C_u}$, as follows:

$$score_c = \boldsymbol{p_u}\, \boldsymbol{V_{:,c}} \quad (26)$$

where $\boldsymbol{V_{:,c}}$ is the embedding of the candidate item (ID = c).

The $score_c$ can be understood as the overlapping between the candidate item (ID = c) and the user's holistic interests. A large $score_c$ means that the user has a high probability to interact with the corresponding item at the next time. Finally, there is a preference score for each candidate item, and thus the top-k items can be identified accordingly.

D. Optimizing the Framework

Based on the above steps, we have built up the personalized recommendation model $\boldsymbol{f_{rec}}$. Then we consider how to train this model.

Given the *historical behavior sequence* $\boldsymbol{S_u}$ of user u, we take the first (t-1) items and the t^{th} item from it, denoted $\boldsymbol{S_{t-1}}$ and i_t, respectively. Then, we can construct such a set $\boldsymbol{D_u} = \left\{\left(\boldsymbol{S_{t-1}}, i_t^{(u)}\right) \mid t = 2, 3, ..., n\right\}$. Recall our task is to predict the most likely item which will be interacted with by the user at the next time, for the corpus $\boldsymbol{S} = \left\{\boldsymbol{S_1}, \boldsymbol{S_2}, ...\boldsymbol{S_{|U|}}\right\}$, a natural optimization objective is:

$$maximize\ L = \prod_{\boldsymbol{S_u} \in \boldsymbol{S}} \prod_{\left(\boldsymbol{S_{t-1}}, i_t^{(u)}\right) \in \boldsymbol{D_u}} P\left(i_t^{(u)} \mid \boldsymbol{S_{t-1}}\right) \quad (27)$$

$$P\left(i_t^{(u)} \mid \boldsymbol{S_{t-1}}\right) = \frac{exp(score_{i_t^{(u)}})}{\sum_{k \in I} exp(score_k)} \quad (28)$$

With this definition, achieving the goal defined in E.q.(27) will force the item i_t to be the one that the user is most likely to interact with at the next time. Nevertheless, optimizing the objective function is non-trivial since each evaluation defined by E.q.(27) needs to traverse all items in the candidate set, which is very time-consuming.

To reduce the complexity, we employ the idea of negative sampling, which approximates the costly denominator term softmax with some sampled negative instances [25]. Let $j_t^{(u)}$ denote the negative instance for $\boldsymbol{S_{t-1}}$, where $j_t^{(u)} \notin \left\{\boldsymbol{S_{t-1}}, i_t^{(u)}\right\}$, we can then approximate the conditional probability $P\left(i_t^{(u)} \mid \boldsymbol{S_{t-1}}\right)$ defined in E.q.(28) as:

$$P\left(i_t^{(u)}, j_t^{(u)} \mid \boldsymbol{S_{t-1}}\right) = \prod_{z \in \left\{i_t^{(u)}, j_t^{(u)}\right\}} P\left(z \mid \boldsymbol{S_{t-1}}\right) \quad (29)$$

where the probability $P\left(z \mid \boldsymbol{S_{t-1}}\right)$ is defined as:

$$P\left(z \mid \boldsymbol{S_{t-1}}\right) = \begin{cases} \sigma\left(score_z\right), & \text{if } z = i_t^{(u)} \\ 1 - \sigma\left(score_z\right), & \text{if } z = j_t^{(u)} \end{cases} \quad (30)$$

where like E.q.(25), σ denotes the sigmoid function $f(x) = 1/(1+e^{-x})$.

By replacing $P\left(i_t^{(u)} \mid \boldsymbol{S_{t-1}}\right)$ in E.q.(28) with the definition of $P\left(i_t^{(u)}, j_t^{(u)} \mid \boldsymbol{S_{t-1}}\right)$ in E.q.(29), we can get the approximated objective function to be optimized. In other words, the probability that the ground-truth sample appears as the next should be maximized, whereas the probability that the negative sample appears as the next should be minimized. Finally, this model can be learned by optimizing a point-wise classification loss, and the objective function $Loss$ can be defined as:

$$L = -\sum_{u} \sum_{i_t^{(u)}, j_t^{(u)}} (log(\sigma(score_{i_t^{(u)}})) + log(1 - \sigma(score_{j_t^{(u)}}))) \tag{31}$$

$$minimize\ Loss = L + \eta_1 \|\boldsymbol{\Theta}\|^2 + \eta_2 P \tag{32}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{V}, \boldsymbol{W_{s1}}, \boldsymbol{W_{s2}}, \cdots, \boldsymbol{W_{s8}}, \boldsymbol{b_{s1}}, \cdots, \boldsymbol{b_{s4}}\}$ is the set of weights, and P is a penalization term which is defined in E.q.(11), and its effectiveness will be discussed in detail in the experiments. In addition, η_1 and η_2 control the strength of the regularization and the penalization item respectively.

V. EXPERIMENTS

In the experiments, we aim to answer the following several questions:

Q1: How does DMFP perform in terms of several common recommendation metrics, compared to other state-of-the-art methods?

Q2: How do *Long-Term Preference* (LTP) and *Short-Term Preference* (STP) affect the performance of DMFP?

Q3: How does the time decay function, compared to time intervals division?

A. *Experimental Settings*

*1) **Datasets**:* We perform experiments on three real-world datasets. The details of them are shown in Table. I.

TABLE I
THE DETAILS OF THE THREE DATASETS

Dataset	#Users	#Items	#Actions	#Avg. actions /user	#Avg. actions /item
Amazon Electronic	127,562	54,205	1,329,336	10.42	24.52
Amazon Clothing	39,387	23,033	278,677	7.07	12.09
Movielens_1m	6,027	3,062	574,026	95.24	187.46

- Amazon Dataset[2]. This is a user purchase and rating dataset collected from *Amazon.com* by [26], and notable for its high sparsity and variability. In this work, we adopt its two subsets: Electronic and Clothing.
- Movielens[3]. This is a popular benchmark dataset for evaluating the performance of recommendation models. In our experiment, the Movielens_1m version is adopted.

In addition, in Amzaon Electronic dataset and Movielens_1m dataset, the items that have been observed by less than 6 users and the users who have interacted with less than 6 items are removed. Let the *historical behaviors sequence* for user u be $\boldsymbol{S_u} = ({i_1}^{(u)}, {i_2}^{(u)}, ...{i_n}^{(u)})$, we use the first k interaction behaviors $({i_1}^{(u)}, {i_2}^{(u)}, ...{i_k}^{(u)})$ to predict the $(k+1)^{th}$ behavior ${i_{k+1}}^{(u)}$ in the training set, where $k = 2, \cdots (n-2)$, and we use the first $(n-1)$ behaviors $({i_1}^{(u)}, {i_2}^{(u)}, \cdots {i_{n-1}}^{(u)})$ to predict the last one ${i_n}^{(u)}$ in the test set. Following the setting in Caser [2], for the short-term behaviors, L is set to 4, i.e., $\widetilde{\boldsymbol{S_u}} = ({i_{n-3}}^{(u)}, {i_{n-2}}^{(u)}, {i_{n-1}}^{(u)}, {i_n}^{(u)})$. When n is less than or equal to L (i.e., $n \leq L$), $\widetilde{\boldsymbol{S_u}}$ and $\boldsymbol{S_u}$ are the identical (i.e., $\widetilde{\boldsymbol{S_u}} = \boldsymbol{S_u}$). In addition, in our experiments, for user u, the candidate set $\boldsymbol{C_u}$ is defined as the collection of items that she has never interacted with before.

*2) **Baselines**:* We compare DMFP with two groups of recommendation baseline methods.

The first group includes one CNN-based recommendation method and an attention-based recommendation method as follows:

- **CNN+Pooling** [27]: This method takes a user's *historical behaviors sequence* as a sentence, and then adopts a 1-D convolution structure with max-pooling to extract user preferences.
- **ATRank** [9]: Inspired by the great success of Transformer [23], this method exploits multi-head self-attention mechanism to model the users' *historical behaviors sequence* for capturing user preferences. In addition, this method adopts time intervals division to preserve temporal information.

The second group contains two typical methods that consider the long-term preference and the short-term preference at the same time as follows:

- **Caser** [2]: This method learns a static embedding for each user as the long-term preference, and applies convolution operations on the embedding matrix of the latest L items of a user to capture the short-term preference. Finally, the long-term preference embedding and the short-term preference embedding are concatenated together as the whole preference of a user for downstream network.
- **SHAN* [1]**: This method is a modified version of SHAN. It takes the items in a session as a basic input unit. Specifically, this method adopts two same attention mechanisms (like additive attention) to model the long-term and short-term historical behaviors, respectively. In our experiments, we use the latest L behaviors instead of the last session/transaction as the short-term behaviors set. In addition, the time decay function is used to model the time signal for each behavior, which does not considered in the origin version of SHAN.

In addition, to evaluate the contribution of the long-term preference and the short-term preference for forming a user's holistic preference, we also compare DMFP with its two variants, denoted as DMFP_L and DMFP_S, respectively. DMFP_L only considers the long-term preference, while DMFP_S only considers the short-term preference.

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

[3]https://grouplens.org/datasets/movielens/

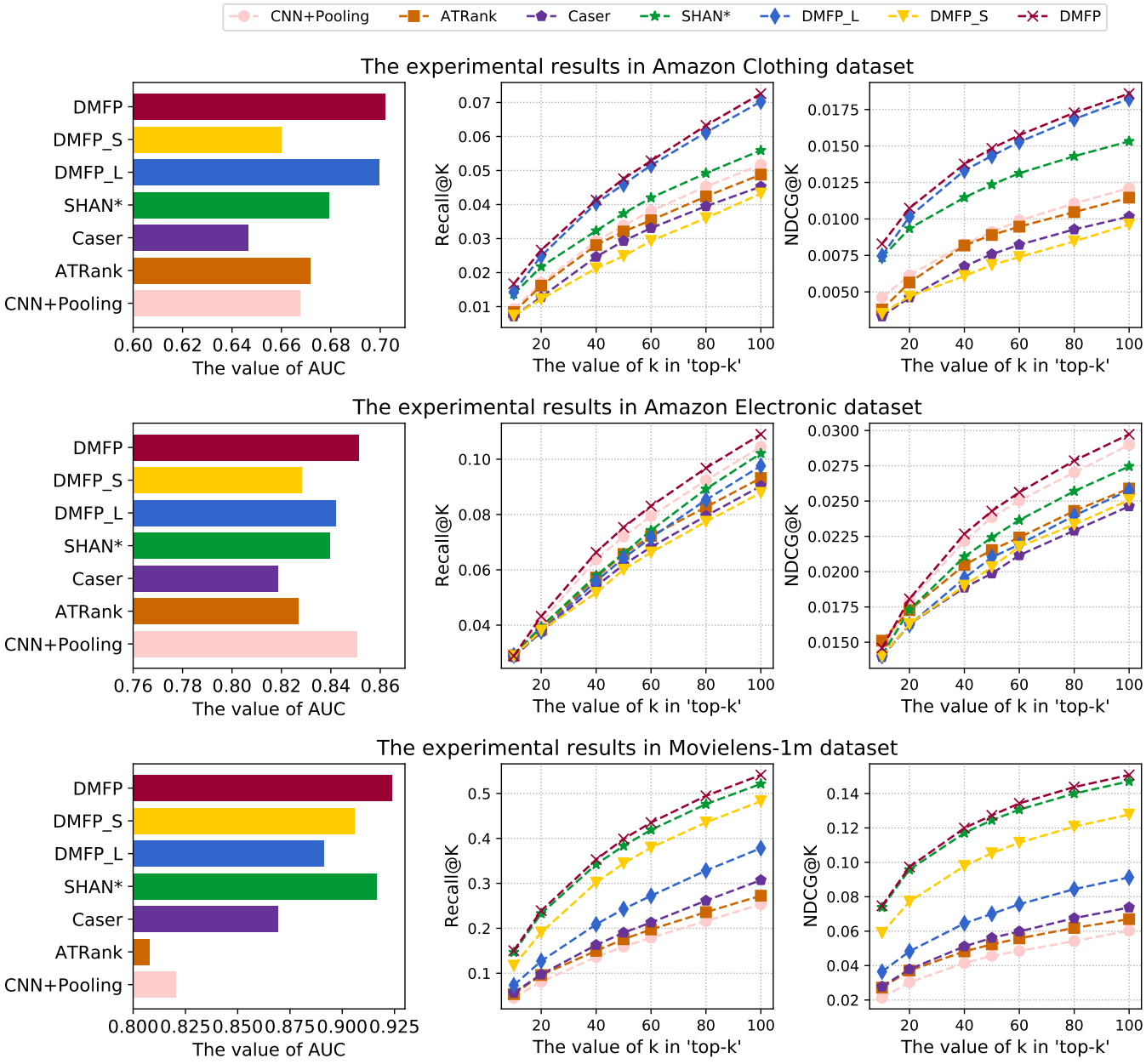

Fig. 4. Performance Comparison of All Methods in Amazon Clothing, Amazon Electronic and Movielens_1m Datasets

*3) **Evaluation Metrics**:* We evaluate all the methods by the following three metrics:

- **Area Under Curve (AUC)**. The AUC metric shown in E.q.(33) is used to investigate how the positive samples are ranked over negative samples, which has been widely used in evaluating the performance of ranking frameworks in recommendations [9], [28].

$$AUC = \frac{1}{|\boldsymbol{U}|} \sum_{u \in \boldsymbol{U}} \delta(socre_{i_t^{(u)}} > socre_{j_t^{(u)}}) \tag{33}$$

where $\boldsymbol{U}$ denotes the user set in the test set, $i_t^{(u)}$ denotes the corresponding ground-truth item for user u, and $j_t^{(u)}$ denotes the corresponding negative sample. $socre_{i_t^{(u)}}$ is a preference score that can be calculated by E.q.(26), and $\delta(*)$ is an indicator function which returns 1 if $*$ is true, and 0 otherwise. The floor of AUC from random guess is 0.5 and the best result is 1.

- **Recall and Normalized Discounted Cumulative Gain (NDCG)**. Recall@K and NDCG@K are two popular metrics for measuring the performance of the Top-K personalized ranking recommendation list [2], [13], [14], [29]. In our experiments, we report Recall@K and NDCG@K with K $\in \{10, 20, 40, 50, 60, 80, 100\}$. Given a list of Top-K predicted items for user u, denoted $\boldsymbol{R_u^K}$, and the corresponding ground-truth item $i_t^{(u)}$ in the test

set, Recall@K and NDCG@K are computed by E.q.(34) to E.q.(36).

$$Recall@K = \frac{1}{|\boldsymbol{U}|} \sum_{u \in \boldsymbol{U}} \delta(i_t^{(u)} \in \boldsymbol{R_u^K}) \tag{34}$$

$$NDCG@K = \frac{1}{|\boldsymbol{U}|} \sum_{u \in \boldsymbol{U}} \frac{2^{index_u} - 1}{log_2(index_u + 1)} \tag{35}$$

$$index_u = \begin{cases} index_u, & \text{if } i_t^{(u)} = \boldsymbol{R_u^K}[index_u] \\ 0, & \text{if } i_t^{(u)} \notin \boldsymbol{R_u^K} \end{cases} \tag{36}$$

where $\boldsymbol{R_u^K}[index_u]$ is the ${index_u}^{th}$ element in list $\boldsymbol{R_u^K}$. In addition, in our test set, since each user only contains one ground-truth item, IDCG@K for each user is equal to 1, which has been removed in E.q.(35). Note that, the higher the value of metrics, the better the quality of the recommendation.

*4) **Hyperparameter Setting**:* Our method and all baseline methods use the common hyperparameters as follows.

All models are learned by optimizing the sigmoid cross entropy loss for a fair comparison, as shown in E.q.(31). All models are trained with Stochastic Gradient Descent (SGD). The batch size for Amazon Clothing and MovieLens_1m is set to 16, but for Amazon Electronic, it is set to 32. The dimension size of each item embedding and the size of all hidden layers are set to 128. In addition, the l2-loss weight (i.e., η_1) is set to 5e-4, the learning-rate starts at 1.0, and decay rate is set to 0.01. For DMFP, the value of r in E.q.(9) is selected from {6, 8, 10, 15, 20}, and the punish weight (i.e., η_2) for P is selected from {0, 3e-4, 3e-3, 3e-2, 3e-1}.

B. Experimental Analysis

*1) **Comparison of Performance (Q1)**:* Fig. 4 depicts the performance of all methods under three metrics, i.e., AUC, Recall@K, and NDCG@K in three real-world datasets. From this figure, we can observe that:

(1) DMFP consistently outperforms all baselines under all evaluation metrics in two sparse datasets (i.e., Amazon Clothing and Electronic) and one dense dataset (i.e., Movielens_1m). Specifically, for Amazon Clothing dataset, DMFP improves 3.36%, 29.83%, and 21.34% in terms of AUC, Recall@100, and NDCG@100, compared with the second best method (i.e., SHAN*) in the three metrics respectively. For Amazon Electronic dataset, DMFP improves 4.33% and 2.55% in terms of Recall@100 and NDCG@100 respectively, compared with the second best method (i.e., CNN+Pooling), respectively. For Movielens_1m dataset, DMFP improves 0.79%, 3.78%, and 2.49% in terms of AUC, Recall@100, and NDCG@100, compared to the second best methods (i.e., SHAN*) respectively. The above experimental results illustrate the effectiveness of DMFP in learning a holistic preference for each user in both sparse and dense datasets.

(2) Compared with the other two methods (i.e., Caser and SHAN*) that consider both long-term preference and short-term preference simultaneously, DMFP always achieves the best performance in the three datasets. Specifically, compared with SHAN*, DMFP can improve 1.37%, 6.75%, and 8.26% in terms of AUC, Recall@100, and NDCG@100 in Amazon Electronic dataset, respectively. This indicates that DMFP can learn the representation of preference for each user, and this representation can combine the multi-faceted long-term preference and fine-grained short-term preference, captured by multi-hops attention and feature-level self-attention together with vertical convolution operation, respectively. While SHAN* can hardly capture users' preferences by only adopting the simple additive attention mechanism. In addition, DMFP outperforms Caser under all evaluation metrics in the three datasets. This is because that capturing the long-term preferences from the up-to-date historical behaviors set can be more effective than learning a static preference embedding for each user.

*2) **The Impact of LTP and STP (Q2)**:* From the experimental results of DMFP, DMFP_L and DMFP_S shown in Fig. 4, we can observe:

(1) DMFP consistently outperforms DMFP_L and DMFP_S in the three datasets. For Amazon Electronic dataset, compared with DMFP_L, DMFP can improve 1.10%, 11.68%, and 15.36% in terms of AUC, Recall@100, and NDCG@100, respectively. Compared with DMFP_S, DMFP improves 6.33%, 67.87%, and 93.24% in terms of AUC, Recall@100,and NDCG@100, respectively. This illustrates that fusing long-term and short-term preferences can more accurately understand users' needs or interests, and thus can make better recommendations.

(2) DMFP_L outperforms DMFP_S in two sparse datasets (i.e., Amazon Clothing and Electronic). Specifically, for Amazon Clothing dataset, DMFP_L improves 6.27%, 65.93%, and 94.80% in terms of AUC, Recall@100, and NDCG@100, respectively. For Amazon Electronic dataset, DMFP_L improves 1.67%, 11.10%, and 2.79% at AUC, Recall@100, and NDCG@100, respectively. In contrast, for the dense dataset (i.e., Movielens_1m), the performance of DMFP_S is much better than that of DMFP_L. Specifically, DMFP_S can improve 1.63%, 27.57% and 39.97% in terms of AUC, Recall@100, and NDCG@100, respectively. From the experimental results, we can see that the contributions of the long-term preference and the short-term preference are data depended. For sparse datasets, there is a long time interval between two adjacent behaviors, thus it is hard to capture the short-term preference from the latest L behaviors. In contrast, for dense datasets that contain sufficient users' feedback, it is possible to mine a user's current preferences from the latest L behaviors. DMFP can be used in both sparse and dense datasets, as it can capture multi-faceted long-term preferences and fine-grained short-term preferences simultaneously.

*3) **The Impact of Time Decay Function (Q3)**:* We choose ATRank [9] as the carrier to investigate the effect of the time decay function in Amazon Clothing and Electronic datasets, as ATRank has clearly pointed out the detail of slicing the continuous time signal into intervals of different sizes in the two datasets. We compare ATRank with its variant ATRank_T. Specifically, in ATRank_T, the time intervals division in ATRank is replaced with time decay function, and other

TABLE II
PERFORMANCE COMPARISON OF ATRANK AND ATRANK_T IN AMAZON CLOTHING, AMAZON ELECTRONIC DATASETS

		AUC	Recall@10	Recall@50	Recall@100	NDCG@10	NDCG@50	NDCG@100
Clothing	ATRank	**67.18%**	0.838%	3.207%	4.880%	0.379%	0.890%	1.146%
	ATRank_T	67.05%	**0.939%**	**3.313%**	**5.149%**	**0.465%**	**0.941%**	**1.225%**
Electronic	ATRank	82.67%	**2.908%**	6.569%	9.319%	1.511%	2.153%	**2.589%**
	ATRank_T	**83.31%**	2.816%	**6.745%**	**10.073%**	**1.525%**	**2.171%**	2.581%

settings are kept as in [9]. Table II lists the performance comparison by ATRank and ATRank_T in Amazon Clothing and Electronic datasets. From the table, we can observe that ATRank_T outperforms ATRank under the most of the metrics. On average, ATRank_T can improve 7.22% in terms of Recall and 6.61% in terms of NDCG. This is because the time decay function is a smooth function. In addition, the AUC delivered by ATRank_T is not always better than ATRank. This is because that the properties of the time decay function can hardly improve the modeling of the time information when many behaviors are concentrated into a small time interval.

C. Conclusion

In this paper, we have proposed a Dynamic Multi-faceted Fine-grained Preference Model (DMFP) for the next-item recommendation. A time decay function is adopted to preserve the temporal information behind each historical behavior. In addition, in order to learn more accurate user preference representations, DMFP adopts a multi-hops attention mechanism to capture the multi-faceted long-term preference of a user, and adopts a feature-level self-attention mechanism and a vertical convolutional operation to mine the fine-grained short-term preference of a user. Extensive validations on three real-world datasets have demonstrated the superiority of DMFP compared with other state-of-the-art methods.

ACKNOWLEDGEMENTS

This work was supported in part by National Natural Science Foundation of China (NSFC) (Grant No. 61972069, 61836007, 61832017, 61532018, 61572336), Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010), and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

REFERENCES

[1] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention networks," in *IJCAI*, 2018, pp. 3926–3932.

[2] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018, pp. 565–573.

[3] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.

[4] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *RecSys*, 2017, pp. 130–137.

[5] T. Zhang, P. Zhao, Y. Liu, V. Sheng, J. Xu, D. Wang, G. Liu, and X. Zhou, "Feature-level deeper self-attention network for sequential recommendation," in *IJCAI*, 2019, pp. 4320–4326.

[6] H. Wang, G. Liu, A. Liu, Z. Li, and K. Zheng, "Dmran:a hierarchical fine-grained attention-based network for recommendation," in *IJCAI*, 2019, pp. 3698–3704.

[7] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, D. Wang, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI*, 2019, pp. 2629–2636.

[8] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng, "Profit-driven task assignment in spatial crowdsourcing," in *IJCAI*, 1914-1920, pp. 2629–2636.

[9] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *AAAI*, 2018, pp. 4564–4571.

[10] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *CVPR*, 2016, pp. 4651–4659.

[11] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-over-attention neural networks for reading comprehension," in *ACL*, 2017, pp. 593–602.

[12] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *AAAI*, 2018, pp. 2532–2539.

[13] L. Yu, C. Zhang, S. Liang, and X. Zhang, "Multi-order attentive ranking model for sequential recommendation," in *AAAI*, 2019.

[14] W. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018, pp. 197–206.

[15] D. Peng, W. Yuan, and C. Liu, "HARSAM: A hybrid model for recommendation supported by self-attention mechanism," *IEEE Access*, pp. 12 620–12 629, 2019.

[16] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *SIGIR*, 2016, pp. 729–732.

[17] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*, 2017.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[19] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," in *ACL*, 2015, pp. 1577–1586.

[20] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Q. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *NAACL-HLT*, 2018, pp. 327–333.

[21] J. P. A. Vieira and R. S. Moura, "An analysis of convolutional neural networks for sentence classification," in *CLEI*, 2017, pp. 1–5.

[22] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *EMNLP*, 2016, pp. 551–561.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[24] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next item recommendation with self-attention," *CoRR*, 2018.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[26] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*, 2015, pp. 43–52.

[27] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014, pp. 1746–1751.

[28] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *RecSys*, 2016, pp. 191–198.

[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Www, Perth, Australia, April 3-7*, 2017, pp. 173–182.