DMRAN: A Hierarchical Fine-Grained Attention-Based Network for Recommendation

Huizhao Wang,[1] **Guanfeng Liu,**[2] **An Liu,**[1*] **Zhixu Li**[1] and **Kai Zheng**[3*]
[1]Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, China
[2]Department of Computing, Macquarie University, Sydney, NSW, Australia
[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China
guanfeng.liu@mq.edu.au, {anliu, zhixuli}@suda.edu.cn, zhengkai@uestc.edu.cn

Abstract

The conventional methods for the next-item recommendation are generally based on RNN or one-dimensional attention with time encoding. They are either hard to preserve the long-term dependencies between different interactions, or hard to capture fine-grained user preferences. In this paper, we propose a Double Most Relevant Attention Network (DMRAN) that contains two layers, i.e., Item level Attention and Feature Level Self-attention, which are to pick out the most relevant items from the sequence of user's historical behaviors, and extract the most relevant aspects of relevant items, respectively. Then, we can capture the fine-grained user preferences to better support the next-item recommendation. Extensive experiments on two real-world datasets illustrate that DMRAN can improve the efficiency and effectiveness of the recommendation compared with the state-of-the-art methods.

1 Introduction

A user's next choice is relevant to her previous behaviors [Wang *et al.*, 2018; Zhou *et al.*, 2018; Zhu *et al.*, 2018; Zhu *et al.*, 2017]. Then, a typical personalized recommendation process can be naturally divided into the following steps: (1) modeling the users' historical behaviors to capture the potential co-occurrence relationship between different items; and (2) exploiting the co-occurrence relationship to predict the next item that a user is likely to interact with. Here, the co-occurrence relationships are also called sequential patterns or dependencies.

Recently, inspired by the success of the attention mechanism [Bahdanau *et al.*, 2015], the related models equipping RNN with an attention mechanism have been proposed and achieved good results in recommendations [Chen *et al.*, 2018; Khattar *et al.*, 2018]. In contrast to RNN, the attention mechanism can maintain a variable-length memory, and thus provides complementary information to the sequence patterns modeled by RNN. In addition, it can reduce the online service delay and the time cost of offline training. Some studies [Wang *et al.*, 2018; Zhou *et al.*, 2018] adopt a structure that is only composed of the attention mechanism, as the attention computation is parallel compared to the sequential computation of RNN.

Figure 1: An example of the difference between the one-dimensional attention and the multi-dimensional attention

Intuitively, a user interacts with an item, may just like some aspects of the item. For example, as shown in Figure 1, a user may buy a piece of clothing because of its style, rather than the brand. In other words, the impact of different features of an item on the next choice is different.

However, the existing attention based recommendations [Wang *et al.*, 2018; Zhou *et al.*, 2018; Chen *et al.*, 2018; Yu *et al.*, 2019; Yu *et al.*, 2018] do not consider such different preferences. Specifically, this attention-based pooling can be formulated as $\boldsymbol{C_j} = \sum_{i=1}^{n} a_{ij}\vec{\boldsymbol{v_i}}$, where a_{ij} is a scalar and measures the dependency between $item_i$ and $item_j$, or the attention of $item_i$ to $item_j$. $\vec{\boldsymbol{v_i}}$ is an abstract representation of the i-th item in user historical interaction sequence. In one-dimensional attention, each dimensional in $\vec{\boldsymbol{v_i}}$ will multiply by the same value of a_{ij}, and thus, different abstract features/aspects of an item are assigned the same weight.

To this end, we propose Double Most Relevant Attention Network (DMRAN) to learn the sequential patterns for the next-item recommendation. Specifically, different from previous approaches, like ATEM [Wang *et al.*, 2018] and LARN [Pei *et al.*, 2017], this method encodes the time signal by converting continuous temporal features into discrete features, and then performs Item Level Attention and Feature Level Self-attention. By Item Level Attention, the uncorrelated in-

*corresponding author

terference items in the sequence of the historical behaviors are initially filtered out, and thus it can reduce the size of the sequence. By Feature Level Self-attention, the different features of an item can be given different weights, and thus it can capture more fine-grained user preferences, which is a multi-dimensional self-attention. The main contributions of our work are summarized as follows:

- To the best of our knowledge, this is the first method that adopts two different levels of attention mechanisms simultaneously for recommendations, which can capture more fine-grained user preferences and generate a high-level representation of a user.
- Our DMRAN has the advantages of one-dimensional attention and multi-dimensional self-attention at the same time, i.e., the good efficiency and effectiveness.
- We perform extensive experiments on two real-world datasets. The results show our model outperforms the state-of-art methods in terms of Area Under Curve (AUC) and the training time.

2 Related Work

2.1 Sequential Recommendation

Recurrent Neural Networks (RNN) together with its variants LSTM and GRU have been widely applied in sequential recommendation, including session-based GRU [Hidasi *et al.*, 2016], dynamic recurrent model [Yu *et al.*, 2016], and hierarchical personalized RNN model [Quadrana *et al.*, 2017]. 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 parallelized, 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 embedding task [Vieira and Moura, 2017], Caser [Tang and Wang, 2018] 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, in CNN, the fixed-size encoding vector may not support both short and long sequences well.

2.2 Attention and Self-Attention

Attention has been widely used in, such as machine translation task [Bahdanau *et al.*, 2015], and reading comprehension [Cui *et al.*, 2017; Cheng *et al.*, 2016], 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) [Wang *et al.*, 2018] can learn an attentive context embedding that intensifies relevant items but downplays those irrelevant to the next choice. Different from attention, self-attention considers the inner-relations of a sequence, and thus can learn the sequence patterns and internal dependencies. Following the structure of Transformer [Vaswani *et al.*, 2017], ATRank [Zhou *et al.*, 2018] transforms the interaction sequence into a new sequence via self-attention, and has achieved good performance in the next-item recommendation.

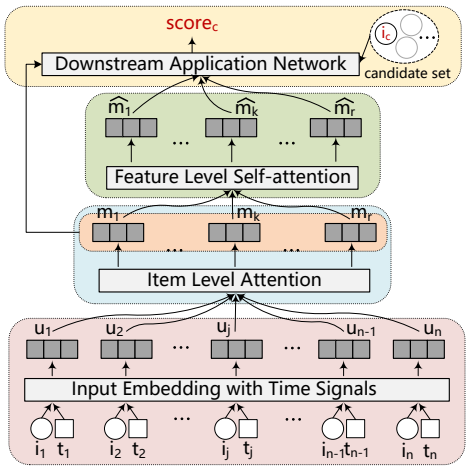

Figure 2: The architecture of DMRAN

3 Problem Formulation

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

Let $\boldsymbol{U} = \{u_1, u_2, \ldots, u_{|U|}\}$ 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, where we concern whether a user $u \in \boldsymbol{U}$ had interacted with an item $i \in \boldsymbol{I}$ at relative time index t. Hence each interaction record can be formulated as a triple $i_t{}^{(u)} = \langle u, i, t \rangle$. By sorting the interaction records in ascending order according to the corresponding time signal, we can form an *interaction sequence* for user u, denoted as $\boldsymbol{S_u} = (i_1{}^{(u)}, i_2{}^{(u)}, \ldots i_n{}^{(u)})$, where n is the length of the *interaction sequence*.

The task of personalized recommendation aims to rank all items in a candidate set based on their probabilities that a user will interacted with at the next time. Formally, the problem can be defined as follows.

Input: The interaction sequences of all users, namely $\boldsymbol{S} = \{\boldsymbol{S_1}, \boldsymbol{S_2}, \ldots \boldsymbol{S}_{|U|}\}$.

Output: A personalized ranking model for recommendation, 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, when entering a user's *interaction sequence* $\boldsymbol{S_u}$.

4 Double Most Relevant Attention Network

The overall architecture of DMRAN is shown in Figure 2. Next, we discuss each of them in detail.

4.1 Input Embedding With Time Signals

Similar to discrete word symbols in natural language processing [Mikolov *et al.*, 2013], the original item IDs have

very limited representation capacity. Therefore, our model first employs a fully connected layer to embed item IDs (i.e., one-hot representations) into a continuous low-dimensional space. Formally, let $\boldsymbol{V} \in \mathbb{R}^{d_e \times |\boldsymbol{I}|}$ be a matrix consisting of the item embedding, where d_e is the dimensionality of the latent embedding spaces. In addition, to compensate for the loss of temporal order information caused by abandoning the sequence model RNN, following the ATRank [Zhou *et al.*, 2018], we split the time signals based on days into multiple granularity, e.g., the continuous time signal in range of $[0,1),[1,2),[2,4),\ldots,[2^k,2^{(k+1)})$ can be mapped to the discrete feature $0,1,2,\ldots,(k+1)$, and then we can get a matrix $\boldsymbol{T} \in \mathbb{R}^{d_e \times (k+2)}$ that includes the time signal embedding. It's worth noting that different behavior groups may have different granularities of time slicing. Finally, we can encode the behavior of a user ${i_t}^{(u)} = \langle u, i, t \rangle$ as

$$\boldsymbol{u_{it}} = dense\left(\boldsymbol{V_i} \oplus \boldsymbol{T_{i-t}}\right) \tag{1}$$

where i-t is the result of t mapping; i and i-t index the i^{th} and $(i\text{-}t)^{th}$ column of $\boldsymbol{V}$ and $\boldsymbol{T}$ respectively; $\oplus$ is the concatenation operator; $dense()$ refers to a full connected layer. To keep the notations simple, in the rest of this paper, we ignore the subscript t and use $\boldsymbol{u_i}$ to replace $\boldsymbol{u_{it}}$. Thus, the interaction sequence $\boldsymbol{S_u}$ can be encoded as a 2-D matrix $\boldsymbol{H} \in \mathbb{R}^{n \times d_e}$.

$$\boldsymbol{H} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots \boldsymbol{u_n}) \tag{2}$$

4.2 Item Level Attention

After the embedding, we will pick the informative items that can reflect the user's interests or relevant to the next choice from the corresponding *interaction sequence*. Thus, one-dimensional attention can be adopted, like ATEM [Wang *et al.*, 2018], where the items being more informative or relevant to the next choice are given larger weights. Specifically, the attention mechanism takes the *interaction sequence* $\boldsymbol{H}$ as input, and outputs a vector of weights $\boldsymbol{a} \in \mathbb{R}^n$:

$$e\left(u_j\right) = \boldsymbol{w_{s2}} ReLU\left(\boldsymbol{W_{s1}} \boldsymbol{u_j}^{\boldsymbol{T}}\right) \tag{3}$$

$$\alpha_j = \frac{exp\left(e\left(u_j\right)\right)}{\sum_{i=1}^{n} exp\left(e\left(u_i\right)\right)} \tag{4}$$

$$\boldsymbol{a} = (\alpha_1, \alpha_2, \cdots \alpha_n) \tag{5}$$

where $j \in \{1, 2, ..., n\}$, $\boldsymbol{W_{s1}} \in \mathbb{R}^{d_e \times d_e}$, $\boldsymbol{w_{s2}} \in \mathbb{R}^{d_e}$, and the attention score $e\left(u_j\right)$ is a scalar. E.q. 4 ensures the sum of the computed weight α_j equals 1.

Then we sum up the interaction sequence $\boldsymbol{H}$ according to the weight provided by $\boldsymbol{a}$ to get a vector $\boldsymbol{m} \in \mathbb{R}^{d_e}$:

$$\boldsymbol{m} = \sum_{j=1}^{n} \alpha_j \boldsymbol{u_j} \tag{6}$$

where $\boldsymbol{m}$ can be seen as an abstract representation of the user's interests. Inspired by the discovery of sentence embedding task [Lin *et al.*, 2017], this vector $\boldsymbol{m}$ focuses on a special set of related words or phrases, and only reflects an aspect of the overall semantics of a sentence. Analogy to our task, the vector representation $\boldsymbol{m}$ usually focuses a small proportion items in the *interaction sequence*, and thus cannot cover all interests of a user, especially for a long *interaction sequence*. Thus, to represent the multifaceted and overall preferences of a user, we perform multiple hops of attention and get multiple and different vectors that focuses on different component of the *interaction sequence*. For simplicity, the above process is formalized:

$$\boldsymbol{A} = softmax\left(\boldsymbol{W_{s2}} ReLU\left(\boldsymbol{W_{s1}} \boldsymbol{H^T}\right)\right) \tag{7}$$

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{H} = (\boldsymbol{m_1}, \boldsymbol{m_2}, \cdots \boldsymbol{m_r})^{\boldsymbol{T}} \tag{8}$$

where $\boldsymbol{w_{s2}} \in \mathbb{R}^{d_e}$ is extended into $\boldsymbol{W_{s2}} \in \mathbb{R}^{r \times d_e}$, $softmax()$ is performed along the second dimension of its input, thus the vector $\boldsymbol{a} \in \mathbb{R}^n$ becomes a weight matrix $\boldsymbol{A} \in \mathbb{R}^{r \times n}$, and the embedding vector $\boldsymbol{m} \in \mathbb{R}^{d_e}$ becomes a embedding matrix $\boldsymbol{M} \in \mathbb{R}^{r \times d_e}$, where r is the number of components that are expected to be extracted from the interaction sequence.

The above attention makes our model focus on the items that can reflect the users' multifaceted interests and can reduce the interference from the irrelevant items. Thus it can be understood as a high level representation of one query "**which items are informative or relevant to the next choice?**".

4.3 Feature Level Self-attention

The affect of different features of an item to the next choice is different and there are even some irrelevant features which tends to overwhelm the influence of a few truly relevant ones. Thus, to capture more fine-grained user preferences, we give more attention on these relevant features, after picking out the relevant items. Specifically, we propose a feature level self-attention. Instead of computing a one-dimensional attention scalar score for each element of a sequence, the feature level self-attention computes a multi-dimensional alignment vector for any two elements of a sequence.

Suppose $\boldsymbol{m_i}$ and $\boldsymbol{m_j}$ are two vectors in embedding matrix $\boldsymbol{M}$, which represent two different components of the *interaction sequence* $\boldsymbol{H}$ respectively, so the attention $\boldsymbol{f}\left(\boldsymbol{m_i}, \boldsymbol{m_j}\right) \in \mathbb{R}^{d_e}$ between $\boldsymbol{m_i}$ and $\boldsymbol{m_j}$ is:

$$\begin{aligned} &\boldsymbol{f}\left(\boldsymbol{m_i}, \boldsymbol{m_j}\right) = \\ &\boldsymbol{W_{s5}} \tanh(\boldsymbol{W_{s4}} \boldsymbol{m_i} + \boldsymbol{W_{s3}} \boldsymbol{m_j} + \boldsymbol{b_{s2}}) + \boldsymbol{b_{s1}} \end{aligned} \tag{9}$$

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 $\boldsymbol{b_{s1}}$,$\boldsymbol{b_{s2}} \in \mathbb{R}^{d_e}$, and $\tanh()$ is a nonlinear activation function.

Then, the alignment vector $\boldsymbol{\beta_{ij}}$ between $\boldsymbol{m_i}$ to $\boldsymbol{m_j}$ can be computed by normalizing along each dimension by E.q. 10 and E.q. 11 as follows.

$$\left[\beta_{ij}\right]_k = \frac{exp\left(\left[f\left(m_i, m_j\right)\right]_k\right)}{\sum_{t=1}^{r} exp\left(\left[f\left(m_i, m_t\right)\right]_k\right)} \tag{10}$$

$$\boldsymbol{\beta_{ij}} = \left(\left[\beta_{ij}\right]_1, \left[\beta_{ij}\right]_2, \cdots \left[\beta_{ij}\right]_{d_e}\right) \tag{11}$$

where $k \in \{1, 2, ...d_e\}$, $[*]_k$ indexes the k^{th} dimension of the vector $*$, and a large $\left[\beta_{ij}\right]_k$ means that the k^{th} abstract feature of $\boldsymbol{m_j}$ is strongly relevant with $\boldsymbol{m_i}$. Finally, the output of

this attention mechanism is still a matrix and its shape is consistent with the corresponding $\boldsymbol{M}$, denoted as $\widehat{\boldsymbol{M}} \in \mathbb{R}^{r \times d_e}$:

$$\widehat{\boldsymbol{m_i}} = \sum_{j=1}^{r} \boldsymbol{\beta_{ij}} \circ \boldsymbol{m_j} \tag{12}$$

$$\widehat{\boldsymbol{M}} = (\widehat{\boldsymbol{m_1}}, \widehat{\boldsymbol{m_2}}, \cdots \widehat{\boldsymbol{m_r}})^{\boldsymbol{T}} \tag{13}$$

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

To sum up, the feature level self-attention computes a weight vector for each element. The more the relevant features of an item, the larger the weight. This can model more detailed dependencies between two interactions and capture more fine-grained user preferences. The above process can be viewed as another query "**which features are the relevant ones over the relevant items?**".

4.4 Downstream Application Network

After obtaining the matrix $\widehat{\boldsymbol{M}}$ and matrix $\boldsymbol{M}$, we need 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 [Zhou *et al.*, 2018; Ying *et al.*, 2018]. Specifically, following ATRank [Zhou *et al.*, 2018], we employ vanilla attention and inner product to compute the corresponding preference score $score_c$ of the candidate item $c \in \boldsymbol{C_u}$, as follows:

$$\boldsymbol{h} = f_{vanilla}(\boldsymbol{M}, \boldsymbol{i_c}) \tag{14}$$

$$\widehat{\boldsymbol{h}} = f_{vanilla}(\widehat{\boldsymbol{M}}, \boldsymbol{i_c}) \tag{15}$$

$$\boldsymbol{h_c} = f_{fusion}(\boldsymbol{h}, \widehat{\boldsymbol{h}}) \tag{16}$$

$$score_c = \boldsymbol{h_c} \boldsymbol{i_c} \tag{17}$$

where $\boldsymbol{i_c}$ is the embedding of the candidate item c; the function $f_{vanilla}()$ represents the execution process of vanilla attention [Vaswani *et al.*, 2017; Zhou *et al.*, 2018] that maps the matrix $\boldsymbol{M}$ or $\widehat{\boldsymbol{M}}$ to a vector $\boldsymbol{h} \in \mathbb{R}^{d_e}$ or $\widehat{\boldsymbol{h}} \in \mathbb{R}^{d_e}$ through the embedding $\boldsymbol{i_c}$, and the $\boldsymbol{h}$ or $\widehat{\boldsymbol{h}}$ can be understood as a overlap between the candidate item c and the user's interests. For flexibility and simplicity, the function $f_{vanilla}()$ can be replaced by a mean operation along the second dimension of the matrix $\widehat{\boldsymbol{M}}$ or $\boldsymbol{M}$ in some scenarios; the function $f_{fusion}()$ refers to the fusion operation by a dimension-wise fusion gate [Shen *et al.*, 2018].

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

$$maximize\ L = \prod_{\boldsymbol{S_u} \in \boldsymbol{S}} \prod_{(\boldsymbol{S_c}, i_t) \in \boldsymbol{D_u}} P\left(i_t \mid \boldsymbol{S_c}\right) \tag{18}$$

$$P\left(i_t \mid \boldsymbol{S_c}\right) = \frac{exp(score_{i_t})}{\sum_{j \in I} exp(score_j)} \tag{19}$$

With this definition, achieving the goal defined in E.q. 18 will force the item i_t to be the one that the user is most likely to interact at the next time. Nevertheless, optimising the objective function is non-trivial since each evaluation of the softmax function needs to traverse all items, leading to expensive time cost. To reduce the complexity, we employ the idea of negative sampling, which approximates the costly denominator term of softmax with some sampled negative instances [Mikolov *et al.*, 2013; Yin *et al.*, 2018].

Let $N_{S_u}(S_c)$ denote the negative instance for $\boldsymbol{S_c}$, where $N_{S_u}(S_c) \notin \boldsymbol{S_c}$, we can then approximate the conditional probability $P\left(i_t \mid \boldsymbol{S_c}\right)$ defined in E.q. 19 as:

$$P\left(i_t, N_{S_u}(S_c) \mid \boldsymbol{S_c}\right) = \prod_{z \in \{i_t, N_{S_u}(S_c)\}} P\left(z \mid \boldsymbol{S_c}\right) \tag{20}$$

where the probability $P\left(z \mid \boldsymbol{S_c}\right)$ is defined as:

$$P\left(z \mid \boldsymbol{S_c}\right) = \begin{cases} \sigma\left(score_z\right), & \text{if } z = i_t \\ 1 - \sigma\left(score_z\right), & \text{if } z = N_{S_u}(S_c) \end{cases} \tag{21}$$

where σ denotes the sigmoid function $1/\left(1 + e^{-x}\right)$. By replacing $P\left(i_t \mid \boldsymbol{S_c}\right)$ in E.q. 19 with the definition of $P\left(i_t, N_{S_u}(S_c) \mid \boldsymbol{S_c}\right)$, we can get the approximated objective function to be optimized. Namely, 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, the objective function can be defined as:

$$\begin{aligned} minimize\ L_\Theta = & -log(L) + \lambda_1 \|\Theta_{uv}\|^2 + \lambda_2 \|\Theta_a\|^2 \\ & + \lambda_3 \left\| \left(\boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}} - \boldsymbol{I}\right) \right\|^2 \end{aligned} \tag{22}$$

where Θ is the set of model parameters; $\Theta_{uv} = \{\boldsymbol{U}, \boldsymbol{V}\}$ is the set of embedding of users and items; Θ_a is the set of weight in model; the last item $\left\| \left(\boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}} - \boldsymbol{I}\right) \right\|^2$ is to punish redundancy between different vectors [Lin *et al.*, 2017], and thus can learn multifaceted interests of a user.

5 Experiment

We aim to answer the following questions in our experiments:

Q1. How does DMRAN perform in terms of efficiency and effectiveness, compared to the state-of-the-art methods?

Q2. How do Item Level Attention (ILA) and Feature Level Self-attention (FLSA) affect the performance of DMRAN?

Q3. How does the key hyper-parameter r (i.e., the number of rows in matrix $\boldsymbol{M}$) affect the performance of DMRAN?

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

Dataset	#Users	#Items	#Samples	#Categories
Amazon Electronic	192,403	63,001	1,689,188	801
Amazon Clothing	39,387	23,033	278,677	-

Table 1: The details of the two datasets

5.1 Experimental Settings

Dataset

We perform experiments on two real-world datasets. The details of them are shown in Table. 1.

The Amazon datasets [McAuley *et al.*, 2015] accumulate user behavior log, and we adopt its two subsets: Electronics, and Clothing. For the *interaction sequence* $({i_1}^{(u)}, {i_2}^{(u)}, ...{i_n}^{(u)})$ for user 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 $k = 1, 2, \ldots, n-2$, and we use the first $(n-1)$ behaviors $({i_1}^{(u)}, {i_2}^{(u)}, ...{i_{n-1}}^{(u)})$ to predict the last one ${i_n}^{(u)}$ in the test set $\boldsymbol{U^t}$. In addition, in our experiments, like ATRank, the category information of each item is considered in Amazon Electronic dataset.

Evaluation Metrics

Same as ATRank [Zhou *et al.*, 2018], our model is a ranking framework which aims to further sort the candidate set delivered by a candidate generation model. Therefore, we adopt the AUC metric shown in E.q. 23 to investigate how the positive samples being ranked over negative samples, which has been widely used in evaluating the performance of ranking frameworks in recommendations [Covington *et al.*, 2016].

$$AUC = \frac{1}{|\boldsymbol{U^t}|} \sum_{u \in \boldsymbol{U^t}} \frac{1}{\left|\boldsymbol{I_u^+}\right| \left|\boldsymbol{I_u^-}\right|} \sum_{i \in \boldsymbol{I_u^+}} \sum_{j \in \boldsymbol{I_u^-}} \delta\left(p_{u,i} > p_{u,j}\right) \quad (23)$$

where $\boldsymbol{I_u^+}$ denotes the positive samples set for user u, and $\boldsymbol{I_u^-}$ denotes the corresponding negative samples set. $p_{u,i}$ is the predicted probability that a user u chooses item i in the test set $\boldsymbol{U^t}$, it can be calculated by E.q. 19. $\delta(\cdot)$ is an indicator function which returns 1 if $p_{u,i} > p_{u,j}$, and 0 otherwise. Note that, the higher the value of AUC, the better the quality of the recommendation.

Baselines

We compare DMRAN with the following baseline methods, including one classic recommendation method (i.e., BPR-MF), three NN-based methods (i.e., Bi-LSTM, Bi-LSTM+Attention, and CNN+Pooling), and one attention-based method (i.e., ATRank), that is the most promising recommendation method.

- **BPR-MF** [Rendle *et al.*, 2009]: This method optimizes the matrix factorization (MF) model with a pairwise ranking-aware objective and aims to maximize the difference between positive and negative items. BPR-MF does not model the time signals.
- **Bi-LSTM** [Zhang *et al.*, 2014]: This method implements an improved version of the Bi-LSTM network for the next-item recommendation and employs ranking-based loss functions for learning the model.
- **Bi-LSTM+Attention**: A vanilla attention [Vaswani *et al.*, 2017] is add on the top of the Bi-LSTM method.

	Amazon	
	Electronic	Clothing
BPR-MF	0.8022	0.6069
Bi-LSTM	0.8734	0.6590
Bi-LSTM+Attention	0.8766	0.6612
CNN+Pooling	0.8804	0.6696
ATRank	0.8910	0.6725
DMRAN_No_Time	0.8712	-
DMRAN_Item	0.8896	0.6714
DMRAN_Feature	0.8835	0.6468
DMRAN	**0.8932**	**0.6779**

Table 2: The AUC values of all models in the two datasets

- **CNN+Pooling** [Tang and Wang, 2018]: This method adopts a 1-D convolution structure with max-pooling to extract user preferences from their historical behaviors.
- **ATRank** [Zhou *et al.*, 2018]: Inspired by the great success of Transformer [Vaswani *et al.*, 2017], this method exploits multi-head self-attention mechanism to model the users' interaction sequences for capturing user preferences.

In addition, to evaluate the impact of each part of our model, we compare DMRAN with its three variants, denoted as DMRAN_No_Time, DMRAN_Item, and DMRAN_Feature respectively. DMRAN_No_Time does not consider the time signal behind each historical behavior, DMRAN_Item and DMRAN_Feature remove Feature Level Self-attention and Item Level Attention from DMRAN, respectively.

Hyperparameter

All models are trained with stochastic gradient descent (SGD). The learning rate starts at 1.0. The batch size, L2-loss weight, and the size of all hidden layers are set to 32 or 16, 5e-5 or 1e-4, and 128, respectively. For DMRAN, we apply a gird search in $\{2, 5, 8, 10, 15, 20\}$ for the special hyperparameter r i.e., the number of rows as shown in E.q. 8. In addition, to ensure the robustness, the residual connection, layer normalization and feed forward network [Vaswani *et al.*, 2017] are adopted for implementations.

5.2 Comparison Of Performance (Q1)

Table. 2 lists AUC values by the baseline methods, DMRAN and its variants in Amazon Electronic and Clothing datasets. From the table, we can observe:

(1) DMRAN always achieves the best performance, compared with all other methods. Specifically, DMRAN improves 0.24% and 0.80% of AUC compared with the second best method (i.e., ATRank) in Amazon Electronic and Clothing datasets, respectively. This demonstrates the effectiveness of DMRAN in capturing the fine-grained user preferences.

(2) DMRAN outperforms DMRAN_No_Time in Amazon Electronic dataset. Specifically, the relative performance improvement is 2.52% of AUC. This fact indicates the good effectiveness of modeling the time signals in some scenarios.

Figure. 3 shows the evolution of AUC values in the Amazon Electronic dataset along with the training procedure. We can see that DMRAN converges fast, and compared to ATRank, DMRAN does not lead to overfitting that causes a rapid drop in AUC. This indicates the efficiency and effectiveness of DMRAN in training the model parameters.

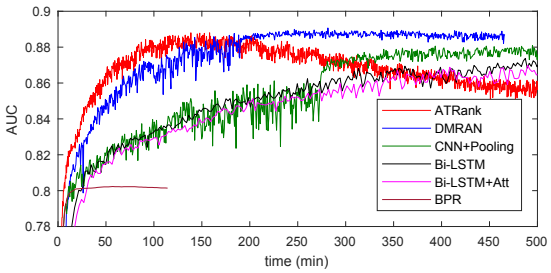

Figure 3: The process of the AUC computation of all methods in Amazon Electronic dataset

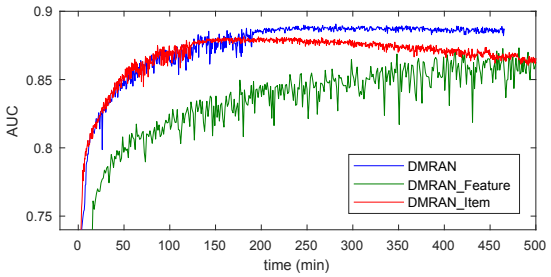

Figure 4: The process of the AUC computation of DMRAN together its variants in Amazon Electronic dataset

To sum up, compared to all baseline methods, DMRAN can get the best ranking result with a good efficiency.

5.3 The Impact of ILA and FLAT (Q2)

Item Level Attention and Feature Level Self-attention are two main parts of our model. We compare DMRAN with its two variants, i.e., DMRAN_Item and DMRAN_Feature.

From the Table. 2, we can see that DMRAN_Item achieves better performance compared with DMRAN_Feature. Based on the statistics, the AUC value of DMRAN_Feature is 0.69% and 3.80% less than that of DMRAN_Item in Amazon Electronic and Clothing datasets, respectively. In addition, the AUC value of DMRAN_Item is only 0.15% and 0.16% less than that of the second best method (i.e., ATRank) respectively. This indicates that Item Level Attention is essential for our model and guarantees the effectiveness of DMRAN.

Figure. 4 shows the convergence process of DMRAN, DMRAN_Item, and DMRAN_Feature in Amazon Electronic dataset. We can see that DMRAN_Item converges faster than DMRAN, and DMRAN converges faster than DMRAN_Feature. The reasons are as follows.

(1) Item Level Attention only computes a weight for each item, but Feature Level Self-attention computes different weights for each features of each item. Thus DMRAN_Feature converges slowly than DMRAN_Item.

(2) Item Level Attention makes the *interaction sequence* (i.e., $\boldsymbol{H}$) into a shorter sequence (i.e., $\boldsymbol{M}$), while extracting the relevant items. Thus, compared to DMRAN_Feature, DMRAN converges faster.

In addition, as shown in Table. 2, the AUC value of DMRAN is 0.40% and 0.96% more than that of DMRAN_Item in Amazon Electronic and Clothing datasets, respectively. This indicates that Item Level Attention can reduce the interference of items that are irrelevant to users' interests on Feature Level Self-attention.

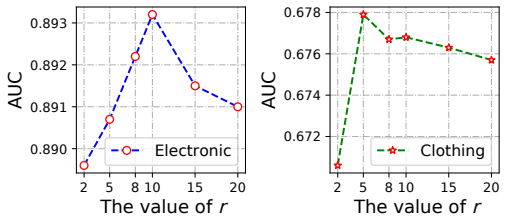

Figure 5: The AUC values by DMRAN based on different r values in the two datasets

5.4 The Impact of Multiple Vectors (Q3)

Figure. 5 shows the AUC values by DMRAN based on different r values in Amazon Electronic and Clothing datasets. We can observe:

(1) When $r = 2$, the AUC value is the minimal in all the cases. Specifically, the AUC values with $r = 2$ are 0.40% and 1.08% less than that of the best performing DMRAN in Amazon Electronic and Clothing datasets, respectively. This indicates that multiple rows in the *interaction sequence* embedding matrix $\boldsymbol{M}$ can provide complementary information about the users preferences for better recommendations.

(2) The AUC value of DMRAN with $r = 2$ is 0.8896 in Amazon Electronic test set. With the increase of the r value to 8 and 10, the AUC value increases to 0.8921 and 0.8932, respectively. While when r reach to 15 and 20, the corresponding AUC values drop to 0.8915 and 0.8910, respectively. This is because: (i) At first, with the increase of r value, there will be more information about users' interests to be embedded in the matrix $\boldsymbol{M}$, and thus the AUC value increases. (ii) However, when r reaches a large value, some useless information that does not match the users' interests will be embedded in the matrix $\boldsymbol{M}$ as well. Thus, the AUC value decreases.

6 Conclusion

In this paper, we have proposed a hierarchical fine-grained Attention-based network (DMRAN) for the next-item recommendation. Based on the two levels of attention mechanism with time encoding, DMRAN not only intensifies relevant items and downplays those irrelevant to the next choice, but also picks out the relevant aspects of the relevant items. Finally, the fine-grained and dynamic user preferences can be captured. Extensive validations on two real-world datasets have demonstrated the superiority of DMRAN against other state-of-the-art methods.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (NSFC) (Grant No. 61572336, 61836007, 61832017, 61532018), 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

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Chen *et al.*, 2018] Jingwu Chen, Fuzhen Zhuang, Xin Hong, Xiang Ao, Xing Xie, and Qing He. Attention-driven factor model for explainable personalized recommendation. In *SIGIR*, pages 909–912, 2018.

[Cheng *et al.*, 2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *EMNLP*, pages 551–561, 2016.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[Cui *et al.*, 2017] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In *ACL*, pages 593–602, 2017.

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[Khattar *et al.*, 2018] Dhruv Khattar, Vaibhav Kumar, Vasudeva Varma, and Manish Gupta. HRAM: A hybrid recurrent attention machine for news recommendation. In *CIKM*, pages 1619–1622, 2018.

[Lin *et al.*, 2017] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[McAuley *et al.*, 2015] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Pei *et al.*, 2017] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David M. J. Tax. Interacting attention-gated recurrent networks for recommendation. In *CIKM*, pages 1459–1468, 2017.

[Quadrana *et al.*, 2017] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*, pages 130–137, 2017.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Shen *et al.*, 2018] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455, 2018.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[Vieira and Moura, 2017] Joao Paulo Albuquerque Vieira and Raimundo Santos Moura. An analysis of convolutional neural networks for sentence classification. In *CLEI*, pages 1–5, 2017.

[Wang *et al.*, 2018] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *AAAI*, pages 2532–2539, 2018.

[Yin *et al.*, 2018] Hongzhi Yin, Lei Zou, Quoc Viet Hung Nguyen, Zi Huang, and Xiaofang Zhou. Joint event-partner recommendation in event-based social networks. In *ICDE*, pages 929–940, 2018.

[Ying *et al.*, 2018] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention networks. In *IJCAI*, pages 3926–3932, 2018.

[Yu *et al.*, 2016] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *SIGIR*, pages 729–732, 2016.

[Yu *et al.*, 2018] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. In *AAAI*, pages 2596–2603, 2018.

[Yu *et al.*, 2019] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *AAAI*, 2019.

[Zhang *et al.*, 2014] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, pages 1369–1375, 2014.

[Zhou *et al.*, 2018] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. Atrank: An attention-based user behavior modeling framework for recommendation. In *AAAI*, pages 4564–4571, 2018.

[Zhu *et al.*, 2017] Jie Zhu, Wei Jiang, An Liu, Guanfeng Liu, and Lei Zhao. Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web Journal*, 20(1):111–134, 2017.

[Zhu *et al.*, 2018] Feng Zhu, Yan Wang, Chaochao Chen, Guanfeng Liu, Mehmet A. Orgun, and Jia Wu. A deep framework for cross-domain and cross-system recommendations. In *IJCAI*, pages 3711–3717, 2018.