SDM: Sequential Deep Matching Model for Online Large-scale Recommender System

Fuyu Lv[1], Taiwei Jin[1], Changlong Yu[2], Fei Sun[1], Quan Lin[1], Keping Yang[1], Wilfred Ng[2]
[1]Alibaba Group, Hangzhou, China
[2]The Hong Kong University of Science and Technology, Hong Kong, China
{fuyu.lfy,taiwei.jtw,ofey.sf,tieyi.lq,shaoyao}@alibaba-inc.com;{cyuaq,wilfred}@cse.ust.hk

ABSTRACT

Capturing users' precise preferences is a fundamental problem in large-scale recommender system. Currently, item-based Collaborative Filtering (CF) methods are common matching approaches in industry. However, they are not effective to model dynamic and evolving preferences of users. In this paper, we propose a new sequential deep matching (SDM) model to capture users' dynamic preferences by combining short-term sessions and long-term behaviors. Compared with existing sequence-aware recommendation methods, we tackle the following two inherent problems in real-world applications: (1) there could exist multiple interest tendencies in one session. (2) long-term preferences may not be effectively fused with current session interests. Long-term behaviors are various and complex, hence those highly related to the short-term session should be kept for fusion. We propose to encode behavior sequences with two corresponding components: multi-head self-attention module to capture multiple types of interests and long-short term gated fusion module to incorporate long-term preferences. Successive items are recommended after matching between sequential user behavior vector and item embedding vectors. Offline experiments on real-world datasets show the superior performance of the proposed SDM. Moreover, SDM has been successfully deployed on online large-scale recommender system at Taobao and achieves improvements in terms of a range of commercial metrics.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Deep Matching; Sequential Recommendation

ACM Reference Format:
Fuyu Lv[1], Taiwei Jin[1], Changlong Yu[2], Fei Sun[1], Quan Lin[1], Keping Yang[1], Wilfred Ng[2]. 2019. SDM: Sequential Deep Matching Model for Online Large-scale Recommender System. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357818

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357818

1 INTRODUCTION

Large-scale recommender systems in industry are required to have both accurate prediction of users' preferences and quick response to their current need. Taobao[1], the largest e-commerce website in China, which supports billions of items and users, firstly retrieves a candidate set of items for a user and then applies a ranking module to generate final recommendations. In the process, the quality of candidates retrieved in the so-called matching module plays a key role in the whole system. Currently, online deployed matching models at Taobao are mainly based on item-based Collaborative Filtering (CF) methods [16, 22]. However, they model static user-item interactions and do not well capture dynamic transformation in users' whole behavior sequences. Such methods usually lead to homogeneous recommendation. To accurately understand interests and preferences of users, sequential order information should be incorporated into the matching module.

In this paper, we consider the dynamic evolution of users' interests by introducing deep sequential recommendation model instead of item-based CF in matching stage. When people begin to use online shopping services at Taobao, their behaviors accumulate to relatively long sequences. The sequences are composed of sessions. A session is a list of user behaviors that occur within a given time frame. A user usually has a clear unique shopping demand in one session [20] while his/her interest can change sharply when he/she starts a new session. Directly modeling the sequences while overlooking such the intrinsic structure would hurt the performance [6]. So we refer to the latest interaction sessions of users as short-term behaviors, other previous as long-term ones. The two parts are modeled separately to encode their inherent information which could be used to represent users' different levels of interests. Our goal is to recall top N items after the user sequences as matching candidates.

When it comes to short-term sessions modeling, methods based on recurrent neural networks (RNNs) have shown effective performance in session-based recommendation [7]. On top of that, Li et al. [14] and Liu et al. [17] further propose attention models to emphasize the main purpose and the effects of the last clicks respectively in a short-term session so that the models can avoid interest shift caused by users' random actions. However, they ignore that users' points of interest are multiple in a session. We observe that customers care about multiple aspects of items such as categories, brand, color, style and shop reputation, etc. Before making the final decision for the most preferred item, users compare many items repeatedly. Thus using single dot-product attention representations

[1]https://www.taobao.com/

fails to reflect diverse interests happening in different time of purchasing. Instead, multi-head attention [25], firstly proposed for machine translation tasks, allows models to jointly attend to multiple different information of different positions. The multi-head structure could naturally solve the issue of multiple interests by representing preferences from different views. So we propose our multi-interest module to augment the RNN-based sequential recommender using multi-head attention. At the same time, equipped with such self-attention, our module can represent accurate users' preferences by filtering out the causal clicks.

Users' general preferences from long time always influence the decisions at present [2, 5, 15, 29, 33]. Intuitively, if a user is a NBA basketball fan, he may view/click abundant items related to NBA stars. When he chooses to buy shoes now, sneakers of famous stars would attract him more than ordinary ones. Hence it is crucial to consider both long-term preferences and short-term behaviors. Ying et al. [29] and Li et al. [15] both take customers' long-term preferences into account by simple combination with the current session. However, in real-world applications, customers have various and abundant shopping demands and their long-time behaviors are also complex and diverse. Stuffs related with NBA stars only take up a pretty small number of long-term behaviors. The long-term user preference, which is related to current short-term session, can not be significantly revealed in the overall long-term behavior representations. If we simply concatenate long- and short-term representations or sum them up over weighted attention, it is not an effective way to fuse. Information related to current short-term session in the long-term vectors should be kept.

In our matching model, we design a gated fusion module to merge global (long-term) and local (short-term) preference features. The input to the gated fusion module is user profile embedding, long-term and short-term representation vectors. Then a gate vector is learned to control the fusion behaviors like different gates in LSTM so that the model could precisely capture interest correlation as well as users' attention to long/short-term interests. On the one hand, the most relevant information in the long-term vectors is fused with short-term vectors. On the other hand, users could have more accurate attention to long/short term interests. Unlike the scalar weights of attention-like models, the gate vector has more powerful representation ability for decision in our super complex neural networks.

The main contributions of this paper are summarized below:

- We develop a novel sequential deep matching (SDM) model for large-scale recommender system in real-world applications by considering both short- and long-term behaviors. These two parts are modeled separately, which represent different levels of user interests.
- We propose to model short-term session behaviors by multi-head self-attention module to encode and capture multiple interest tendencies. A gated fusion module is used to effectively combine long-term preferences and current shopping demands, which incorporates their correlation information rather than simple combinations.
- Our SDM model is evaluated on two offline datasets in the real world and outperforms the other state-of-the-art methods. To demonstrate its scalability in industrial applications, we successfully deployed it on production environment of recommender system at Taobao. The SDM model has been running online effectively since December 2018 and achieves significant improvements compared to previous online system.

2 RELATED WORK

2.1 Deep Matching in Industry

To develop more effective matching models in industrial recommender system, many researchers adopt deep neural networks which have the powerful representation ability. Models based on Matrix Factorization (MF) [13] try to decompose pairwise user-item implicit feedback into user and item vectors. YouTube [4] uses deep neural network to learn both embeddings of users and items. The two kinds of embeddings are generated from their corresponding features separately. The prediction is made as equivalent to search the nearest neighbors of users' vectors among all the items. Besides, Zhu et al. [34] proposes a novel tree-based large-scale recommender system, which can provide novel items and overcome the calculation barrier of vector search. Recently, graph embedding based methods are applied in many industrial applications to complement or replace traditional methods. Wang et al. [26] proposes to construct an item graph from users' behavior history and then applies the state-of-the-art graph embedding methods to learn the embedding of each item. To address the cold start and sparsity problem, they incorporate side information of items to enhance the embedding procedure. Ying et al. [30] develops and deploys an effective and efficient graph convolutional network at Pinterest[2] to generate embeddings of nodes (items) that incorporates both graph structure as well as node feature information. But these models can't well take the dynamic and evolving of users' preferences into consideration. In this work, we consider this in matching stage by introducing sequential recommendation.

2.2 Sequence-aware Recommendation

Sequential recommendation aims at modeling users' preferences and predicting users' future actions such as next clicks or purchases from observed actions in a sequence manner. Previously, FPMC [21] and HRM [27] model the local sequential behaviors between adjacent items in a sequence by combining Matrix Factorization and Markov Chains. Recently, deep neural networks bring powerful representation and generalization ability for recommender system.

Hidasi et al. [7] firstly applies gated recurrent unit (GRU) to make recommendations based on users' current short sessions. Afterwards, Li et al. [14] leverages attention mechanism to extract users' main purpose especially for longer sessions and achieves better results. Liu et al. [17] subsequently creates a novel short-term attention priority model instead of RNNs and then points out the importance of the last click in a session. Besides RNNs, Tang and Wang [24] and Yuan et al. [31] propose convolutional sequence embedding recommendation models as a solution. Kang and McAuley [12] and Zhang et al. [32] use self-attention only architecture to encode user's action history. Tang et al. [23] builds a M3 model that can combine different methods above by a gating

[2] https://www.pinterest.com/

mechanism. But these methods overlook the multiple interests in one session.

Chen et al. [3] introduces the memory mechanism to sequential recommender systems, which designs a user memory-augmented neural network (MANN) to express feature-level interests. As for more fine-grained user preference, Huang et al. [8, 9] use knowledge base information to enhance the semantic representation of key-value memory network called knowledge enhanced sequential recommender. However, the extra storage, manual feature design and computation of memory network of these methods cannot be accepted in industry on account of the large-scale users and items.

It's also crucial to take customers' long-term stable preferences into consideration. Li et al. [15] proposes BINN model by concatenating users' session behavior representations and stable preferences of historical purchasing behaviors. Ying et al. [29] comes up with a novel two-layer hierarchical attention network to recommend the next item that one user might be interested in. Bai et al. [2] uses multi-time scales to characterize long-short time demands and incorporate them into a hierarchical architecture. Another instance of unifying general and sequential interests is Recurrent Collaborative Filtering [5], which combines RNN sequence model and matrix factorization method in a multi-task learning framework. And Zhao et al. [33] does the same combination by adversarial training. Simple combinations are not effective enough to fuse short/long preferences, while multi-task and adversarial methods are not applicable in industrial applications. In this paper, we propose multi-head self-attention to capture multiple user interests in short-term session behaviors and use gating mechanism to incorporate long-term preferences effectively and efficiently in a real-world application.

3 THE PROPOSED APPROACH

3.1 Problem Formulation

We first formulate the sequential matching problem and our solution as well as mathematical notations for variables in the deep model. Let $\mathcal{U}$ denote a set of users and $\mathcal{I}$ denote a set of items. Our model considers whether a user $u \in \mathcal{U}$ would interact with an item $i \in \mathcal{I}$ at time t. For user u, we can get his/her latest sessions by sorting the interacted items in the ascending order of time. Inspired by session-based recommendation [7, 14], we reformulate the new session generation rules:

- Interactions with the same session ID recorded by the backend system belong to the same one.
- Adjacent interactions with time difference less than 10 minutes (or longer depending on the specific scenario) are also merged into one session.
- Maximum length of a session is set to 50, which means a new session will begin when the session length exceeds 50.

Each latest session of user u is regarded as the short-term behavior, namely $\mathcal{S}^u = [i_1^u, ..., i_t^u, ..., i_m^u]$, where m is the length of the sequence. The long-term behaviors of u that happened before $\mathcal{S}^u$ in past 7 days are denoted by $\mathcal{L}^u$. Based on these preliminaries, we can define our recommendation task. Given the short-term behaviors $\mathcal{S}^u$ and long-term behaviors $\mathcal{L}^u$ of user u, we would like to recommend items for him/her.

The general network structure is illustrated in Figure 1. Our model takes current session $\mathcal{S}^u$ and $\mathcal{L}^u$ as input. $\mathcal{S}^u$ and $\mathcal{L}^u$ are

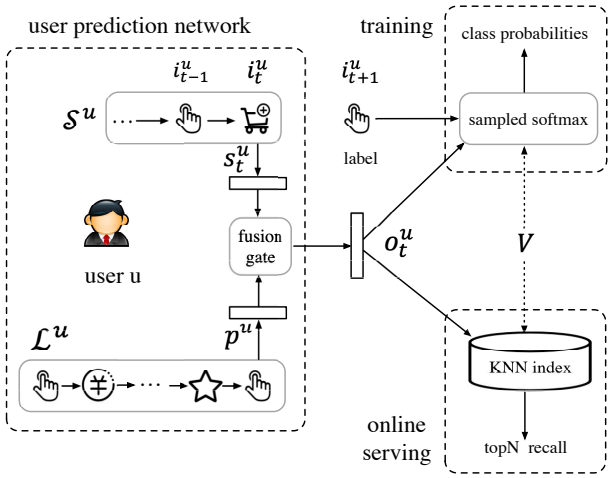

Figure 1: The general network structure of our SDM model. The user prediction network takes user's short-term session $\mathcal{S}^u = [..., i_{t-1}^u, i_t^u]$ and long-term behavior $\mathcal{L}^u$ as input. The target of the network is the next interacted item i_{t+1}^u. s_t^u and p^u denote short-term and long-term representations respectively. o_t^u is the predicted user behavior vector. V is the item embedding vectors.

respectively encoded into short-term session representation $\boldsymbol{s_t^u}$ at time t and long-term behaviors representation $\boldsymbol{p^u}$. The two kinds of representations are combined through a gated neural network. We name this module as *user prediction network* that predicts user behavior vector $\boldsymbol{o_t^u} \in \mathbb{R}^{d \times 1}$ from $\mathcal{S}^u$ and $\mathcal{L}^u$. Let $\boldsymbol{V} \in \mathbb{R}^{d \times |\mathcal{I}|}$ denote *item embedding vectors* of $\mathcal{I}$ where $|\mathcal{I}|$ is the number of all items and d is the embedding size of each vector. Our goal is to predict top N item candidates at time $t+1$ based on the scores of inner product between $\boldsymbol{o_t^u}$ and each column vector $\boldsymbol{v_i}$ in $\boldsymbol{V}$

$$z_i = \text{score}(\boldsymbol{o_t^u}, \boldsymbol{v_i}) = \boldsymbol{o_t^u}^T \boldsymbol{v_i} \tag{1}$$

where $\boldsymbol{v_i} \in \mathbb{R}^{d \times 1}$ is the ith item's embedding vector.

3.2 Training and Online Serving

During training process, the positive label at time t is the next interacted item i_{t+1}^u. Negative labels are sampled from $\mathcal{I}$ excluding i_{t+1}^u by the log-uniform sampler considering the large amount of items in a real-world application. Then the prediction class probabilities are made by a softmax layer. This is called sampled-softmax [10] and we apply cross-entropy as loss function

$$\begin{aligned} \hat{\boldsymbol{y}} &= \text{softmax}(\boldsymbol{z}) \\ L(\hat{\boldsymbol{y}}) &= -\sum_{i \in \mathcal{K}} y_i \log(\hat{y}_i) \end{aligned} \tag{2}$$

where $\mathcal{K}$ is the sampled subset of $\mathcal{I}$ including positive and negative labels, $\boldsymbol{z} = [z_1, ..., z_{|\mathcal{K}|}]$ is the inner product scores between $\boldsymbol{o_t^u}$ and each $\boldsymbol{v_i}$ ($i \in \mathcal{K}$), $\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_{|\mathcal{K}|}]$ is the prediction probability distribution over each sampled item, and y_i is the truly probability distribution of item i.

We deploy the model on our online recommender system. The item embedding vectors $\boldsymbol{V}$ are imported into an efficient K-Nearest-Neighborhood (KNN) similarity search system [11]. Meanwhile, the user prediction network is deployed on a high-performance

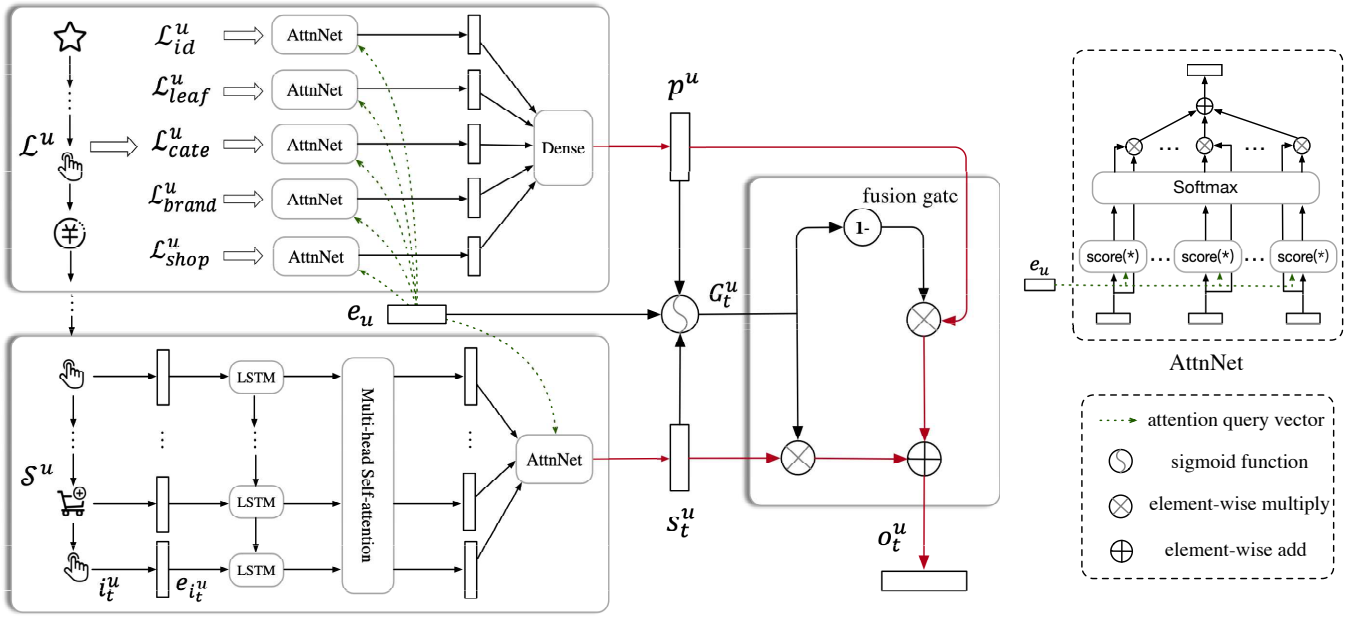

Figure 2: Each $i_t^u \in \mathcal{S}^u$ is embedded into a vector $e_{i_t^u}$. Short-term representation s_t^u is encoded by LSTM and attention mechanism. We describe long-term behaviors $\mathcal{L}^u$ from various side information, i.e., item ID ($\mathcal{L}_{id}^u$), first level category ($\mathcal{L}_{cate}^u$), leaf category ($\mathcal{L}_{leaf}^u$), brand ($\mathcal{L}_{brand}^u$) and shop ($\mathcal{L}_{shop}^u$). Long-term representation p^u is encoded through attention and dense fully-connected networks. s_t^u and p^u are fused into user behavior vector o_t^u through a gate vector G_t^u. e^u is user profile embedding.

real-time inference system of machine learning. This kind of architecture follows the YouTube for video recommendation [4]. When customers use our online service, they will interact with lots of items and their feedback about the items will be recorded by the backend system. These information will be processed and then stored in database as users' behavior logs. Useful information from massive logs are extracted to be constructed into structured data that our model requires. At time t, customer's historical behaviors ($\mathcal{S}^u$ and $\mathcal{L}^u$) are fed into the inference system. Then the user behavior vector $\boldsymbol{o}_t^u$ is predicted. The KNN search system retrieves the most similar items with $\boldsymbol{o}_t^u$ according to their inner products. Top N items are then recommended. Now we elaborate on how $\mathcal{S}^u$ and $\mathcal{L}^u$ are encoded in the network and how the two representations are fused as illustrated in Figure 2.

3.3 Input Embedding with Side Information

In Taobao's recommendation scenario, customers not only focus on a specific item itself, but also concern about the brand, shop and price, etc. For example, some people tend to buy items of the specific brand, and the others would like to buy items from shops that they trust in. Furthermore, due to the sparsity caused by the large-scale online items in industry, encoding items only by item ID feature level is far from satisfaction. So, we describe an item from different feature scales, i.e., item ID, leaf category, first level category, brand and shop, which are denoted as side information set $\mathcal{F}$. Each input item, $i_t^u \in \mathcal{S}^u$, is represented as a dense vector $\boldsymbol{e}_{i_t^u} \in \mathbb{R}^{d\times 1}$ transformed by the embedding layer so that they can be fed into the deep neural network directly

$$\boldsymbol{e}_{i_t^u} = \text{concat}(\{\boldsymbol{e}_i^f | f \in \mathcal{F}\}) \tag{3}$$

where $\boldsymbol{e}_i^f = \boldsymbol{W}^f \boldsymbol{x}_i^f \in \mathbb{R}^{d_f \times 1}$ is item i's input embedding of feature f with embedding size d_f. $\boldsymbol{W}^f$ is the feature f's transformation matrix and $\boldsymbol{x}_i^f$ is a one-hot vector.

Similarly, user profile could describe user u from different feature scales, such as age, gender, and life stage. Input of user u's profile information is represented as a dense vector $\boldsymbol{e}_u \in \mathbb{R}^{d\times 1}$

$$\boldsymbol{e}_u = \text{concat}(\{\boldsymbol{e}_u^p | p \in \mathcal{P}\}) \tag{4}$$

where $\mathcal{P}$ is profile features set and $\boldsymbol{e}_u^p$ is embedding of feature p.

3.4 Recurrent Layer

Given the embedded short-term sequence $[\boldsymbol{e}_{i_1^u}, ..., \boldsymbol{e}_{i_t^u}]$ of u, to capture and characterize the global temporal dependency in the short-term sequence data, we apply Long Short Term Memory (LSTM) network as the recurrent cell following session-based recommendation [7, 14, 15]. The LSTM can be described as

$$\begin{aligned}
\boldsymbol{in}_t^u &= \sigma(\boldsymbol{W}_{in}^1 \boldsymbol{e}_{i_t^u} + \boldsymbol{W}_{in}^2 \boldsymbol{h}_{t-1}^u + b_{in}) \\
\boldsymbol{f}_t^u &= \sigma(\boldsymbol{W}_f^1 \boldsymbol{e}_{i_t^u} + \boldsymbol{W}_f^2 \boldsymbol{h}_{t-1}^u + b_f) \\
\boldsymbol{o}_t^u &= \sigma(\boldsymbol{W}_o^1 \boldsymbol{e}_{i_t^u} + \boldsymbol{W}_o^2 \boldsymbol{h}_{t-1}^u + b_o) \\
\boldsymbol{c}_t^u &= \boldsymbol{f}_t \boldsymbol{c}_{t-1}^u + \boldsymbol{in}_t^u \tanh(\boldsymbol{W}_c^1 \boldsymbol{e}_{i_t^u} + \boldsymbol{W}_c^2 \boldsymbol{h}_{t-1}^u + b_c) \\
\boldsymbol{h}_t^u &= \boldsymbol{o}_t^u \tanh(\boldsymbol{c}_t^u)
\end{aligned} \tag{5}$$

where $\boldsymbol{in}_t^u, \boldsymbol{f}_t^u, \boldsymbol{o}_t^u$ represent the *input*, *forget* and *output* gates respectively. The LSTM encodes the short-term interaction sequence of u into a hidden output vector $\boldsymbol{h}_t^u \in \mathbb{R}^{d\times 1}$ at time t, which we call sequential preference representation. $\boldsymbol{c}_t^u$ is the cell state vector carrying information from $\boldsymbol{h}_{t-1}^u$ and flows between cells. We pass $\boldsymbol{h}_t^u$ to the attention network to get higher order representation.

3.5 Attention Mechanism

Under online shopping scenario, customers usually browse some unrelated items alternatively, which are called causal clicks. Unrelated actions would somehow influence the representation of $\boldsymbol{h}_t^u$ in the sequence. We use a self-attention network to decrease the effect of those unrelated actions. Attention network [1, 18, 28] can aggregate various vectors into an overall presentation by assigning different weight scores to each component.

3.5.1 Multi-head Self-Attention. Self-attention is a special case of attention mechanism, which takes the sequence itself as query, key and value vectors of d-dimension. $\hat{\boldsymbol{h}}_t^u$, the output vector after self-attention, can be aggregated from previous hidden outputs of LSTM, $\boldsymbol{X}^u = [\boldsymbol{h}_1^u, ..., \boldsymbol{h}_t^u]$.

Users may have multiple points of interest. For instance, when u is browsing a skirt, both the color and novel style would be the key factors of making decisions. Single attention network would naturally be not enough to capture multiple aspect representations. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions [25] and could model user preference $\hat{\boldsymbol{h}}_t^u \in \mathbb{R}^{d\times 1}$ from multiple views of interest. So we import it in our attention mechanism. The output matrix, $\hat{\boldsymbol{X}}^u = [\hat{\boldsymbol{h}}_1^u, ..., \hat{\boldsymbol{h}}_t^u]$, is calculated as

$$\hat{\boldsymbol{X}}^u = \text{MultiHead}(\boldsymbol{X}^u) = \boldsymbol{W}^O \text{concat}(\mathbf{head}_1^u, ..., \mathbf{head}_h^u) \quad (6)$$

where $\boldsymbol{W}^O \in \mathbb{R}^{d\times hd_k}$ denotes the weight matrix of output linear transformation, h represents the amount of heads and $d_k = \frac{1}{h}d$.

In detail, each $\mathbf{head}_i^u \in \mathbb{R}^{d_k\times t}$ represents a single latent interest

$$\mathbf{head}_i^u = \text{Attention}(\boldsymbol{W}_i^Q \boldsymbol{X}^u, \boldsymbol{W}_i^K \boldsymbol{X}^u, \boldsymbol{W}_i^V \boldsymbol{X}^u) \quad (7)$$

where $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{d_k\times d}$ denote linear transformation weight matrices of query, key and value respectively. Let $\boldsymbol{Q}_i^u = \boldsymbol{W}_i^Q \boldsymbol{X}^u$, $\boldsymbol{K}_i^u = \boldsymbol{W}_i^K \boldsymbol{X}^u$ and $\boldsymbol{V}_i^u = \boldsymbol{W}_i^V \boldsymbol{X}^u$. The attention score matrix is defined as follows

$$\begin{aligned} f(\boldsymbol{Q}_i^u, \boldsymbol{K}_i^u) &= \boldsymbol{Q}_i^{uT} \boldsymbol{K}_i^u \\ \boldsymbol{A}_i^u &= \text{softmax}(f(\boldsymbol{Q}_i^u, \boldsymbol{K}_i^u)) \end{aligned} \quad (8)$$

Finally, by weighted sum pooling, we get

$$\mathbf{head}_i^u = \boldsymbol{V}_i^u \boldsymbol{A}_i^{uT} \quad (9)$$

3.5.2 User Attention. For different users, they usually show various preferences even to similar item sets. Therefore, on top of self-attention network, we add a user attention module to mine more fine-grained personalized information, where $\boldsymbol{e}_u$ is used as the query vector attending to $\hat{\boldsymbol{X}}^u = [\hat{\boldsymbol{h}}_1^u, ..., \hat{\boldsymbol{h}}_t^u]$. The short-term behavior representation $\boldsymbol{s}_t^u \in \mathbb{R}^{d\times 1}$ at time t is calculated as

$$\begin{aligned} \alpha_k &= \frac{\exp(\hat{\boldsymbol{h}}_k^{uT} \boldsymbol{e}_u)}{\sum_{k=1}^{t} \exp(\hat{\boldsymbol{h}}_k^{uT} \boldsymbol{e}_u)} \\ \boldsymbol{s}_t^u &= \sum_{k=1}^{t} \alpha_k \hat{\boldsymbol{h}}_k^u \end{aligned} \quad (10)$$

3.6 Long-term Behaviors Fusion

From long-term view, users generally accumulate interests of different level in various dimensions. A user may often visit a group of similar shops and buy items that belong to the same category repeatedly. Therefore, we also encode long-term behaviors $\mathcal{L}^u$ from different feature scales. $\mathcal{L}^u = \{\mathcal{L}_f^u | f \in \mathcal{F}\}$ consists of multiple subsets: $\mathcal{L}_{id}^u$ (item ID), $\mathcal{L}_{leaf}^u$ (leaf category), $\mathcal{L}_{cate}^u$ (first level category), $\mathcal{L}_{shop}^u$ (shop) and $\mathcal{L}_{brand}^u$ (brand) as illustrated in Figure 2. For example, $\mathcal{L}_{shop}^u$ contains shops that u has interacted in past one week. Entries in each subset are embedded and aggregated into an overall vector through an attention-based pooling considering the quick response under online environment.

Each $f_k^u \in \mathcal{L}_f^u$ is transformed to a dense vector $\boldsymbol{g}_k^u \in \mathbb{R}^{d\times 1}$ by $\boldsymbol{W}^f$ as in section 3.3. Then we use user profile embedding $\boldsymbol{e}_u$ as the query vector to calculate the attention scores and acquire the representation of $\mathcal{L}_f^u$ as

$$\begin{aligned} \alpha_k &= \frac{\exp(\boldsymbol{g}_k^{uT} \boldsymbol{e}_u)}{\sum_{k=1}^{|\mathcal{L}_f^u|} \exp(\boldsymbol{g}_k^{uT} \boldsymbol{e}_u)} \\ \boldsymbol{z}_f^u &= \sum_{k=1}^{|\mathcal{L}_f^u|} \alpha_k \boldsymbol{g}_k^u \end{aligned} \quad (11)$$

$\{\boldsymbol{z}_f^u | f \in \mathcal{F}\}$ are concatenated and fed into a fully-connected neural network

$$\begin{aligned} \boldsymbol{z}^u &= \text{concat}(\{\boldsymbol{z}_f^u | f \in \mathcal{F}\}) \\ \boldsymbol{p}^u &= \tanh(\boldsymbol{W}^p \boldsymbol{z}^u + b) \end{aligned} \quad (12)$$

where $\boldsymbol{p}^u \in \mathbb{R}^{d\times 1}$ is the long-term behavior representation.

To combine with the short-term behaviors, we elaborately design a gated neural network that takes $\boldsymbol{e}_u$, $\boldsymbol{s}_t^u$ and $\boldsymbol{p}^u$ as inputs also shown in Figure 2. A gate vector $\boldsymbol{G}_t^u \in \mathbb{R}^{d\times 1}$ is used to decide contribution percentages of short- and long-term at time t

$$\boldsymbol{G}_t^u = \text{sigmoid}(\boldsymbol{W}^1 \boldsymbol{e}_u + \boldsymbol{W}^2 \boldsymbol{s}_t^u + \boldsymbol{W}^3 \boldsymbol{p}^u + b) \quad (13)$$

The final output, i.e., user behavior vector $\boldsymbol{o}_t^u \in \mathbb{R}^{d\times 1}$, is computed by

$$\boldsymbol{o}_t^u = (\mathbf{1} - \boldsymbol{G}_t^u) \odot \boldsymbol{p}^u + \boldsymbol{G}_t^u \odot \boldsymbol{s}_t^u \quad (14)$$

where $\odot$ is element-wise multiplication.

4 EXPERIMENT SETUP

4.1 Datasets

We construct an offline-online train/validation/test framework to develop our model. Models are evaluated on two offline real-world e-commerce datasets. One is a large dataset sampled from daily logs of online system on **Mobile Taobao App**. The other is from **JD**[3]. Our code and offline datasets are available at https://github.com/alicogintel/SDM.

Offline Dataset of Taobao. We randomly select active users who interacted more than 40 items within 8 consecutive days in December 2018. In addition, we filter users whose interactions are more than 1000 items, which we believe are spam users. Then we collect their historical interaction data, in which the first 7 days for training and the 8th day for testing. We filter out items that appear less than 5 times in the dataset. Session segmentation follows the rules in section 3.1 and we limit the maximum size of each $\mathcal{L}_f^u$ to 20. During training process, we remove sessions whose length are less than 2. In the test stage, we select approximately 10 thousands active users in the 8th day for quick evaluation. Their first 25%

[3]https://www.jd.com/

Table 1: Statistics of offline and online datasets.

Dataset	Data Type	Data Split	#[a]Users	#Items	#Records	#Sessions	S.len[b]	L.size[c]	Time Interval
JD	offline	train	802,479	154,568	9,653,777	2,666,189	3.3	20	15/Mar/2018 - 8/Apr/2018
		test	10,366	74,564	498,492	15,069	8.6	20	9/Apr/2018 - 15/Apr/2018
Taobao	offline	train	498,633	2,053,798	45,157,298	7,011,385	6.1	20	15/Dec/2018 - 21/Dec/2018
		test	13,237	588,306	1,170,401	13,237	9.2	20	22/Dec/2018
	online	train	3.3×10^8	1×10^8	2.1×10^{10}	2.7×10^9	7.1	50	Dec/2018
		test	3.3×10^8	1×10^8	/	/	8.1	50	Dec/2018

[a]# means the number of. [b]S.len is the average length of short-term behaviors. [c]L.size is the maximum size of each subset in long-term behaviors.

short-term sessions on the 8th day are fed into models and the remaining interactions are ground truth. Besides this, customers may browse some items more than once in a day and repeating recommendation should not be encouraged, so we remain these items only once in the test data for a user.

Offline Dataset of JD. Because this dataset is relatively sparser and smaller, we select user-item interaction logs of three weeks for training and one week for testing. Other data construction and cleaning process are the same as Taobao. Statistic details of the two offline datasets are listed in Table 1.

Online Dataset. We select the most effective offline models to deploy on Taobao's production environment. The training data is from the user-item interaction logs of Mobile Taobao App from past 7 days without sampling. The same data cleaning process is applied as offline training datasets. Scales of online users and items expand to hundred million, which can cover the most active products at Taobao, and more long-term behaviors are used. Details can also be found in Table 1. The online model is updated daily as well as the corresponding item and user features.

4.2 Evaluation Metrics

4.2.1 Offline Evaluation. To evaluate the offline effectiveness of different methods, we use **HitRate@K**, **Precision@K**, **Recall@K** and **F1@K** metrics, which are also widely used in other related works in section 2.

HitRate@K represents the proportion of test cases (n_{hit}) which has the correctly recommended items in a top K position in a ranking list, defined as

$$\text{HitRate@K} = \frac{n_{hit}}{N}$$

where N denotes the number of test data. In our experiment, $K = 100$ and $K = 20$ are used for the tests.

Derive the recalled set of items for a user u as P_u ($|P_u| = K$) and the user's ground truth set as G_u. Precision@K reflects how many interested items for a user in the candidates. It is calculated as

$$\text{Precision@K}(u) = \frac{|P_u \cap G_u|}{K}$$

Recall@K represents the ability of coverage in the user's ground truth. It's calculation is

$$\text{Recall@K}(u) = \frac{|P_u \cap G_u|}{|G_u|}$$

To combine precision and recall, F1@K is derived as

$$\text{F1@K}(u) = \frac{2 \times \text{Precision@K}(u) \times \text{Recall@K}(u)}{\text{Precision@K}(u) + \text{Recall@K}(u)}$$

4.2.2 Online Evaluation. We consider the most important online metrics: **pCTR**, **pGMV** and **discovery**.

pCTR is the Click-Through-Rate per page view where each page can recommend 20 items for a user

$$\text{pCTR} = \frac{\#\text{clicks}}{\#\text{pages}}$$

pGMV is the Gross Merchandise Volume per 1,000 page views. Its calculation is in same way

$$\text{pGMV} = 1000 \times \frac{\#\text{pay amount}}{\#\text{pages}}$$

Besides the amount of online traffic and incomes, we also consider user shopping experience. Define *discovery* to measure how many novel items the recommender system can provide for a user

$$\text{discovery} = \frac{\#\text{new categories}}{\#\text{all categories}}$$

where the denominator is the number of all categories that a user clicks per day and the numerator is the number of new ones in past 15 days. We take the average of all users.

4.3 Comparison Methods

We use the following methods to compare with our model on two offline datasets. We also include five variants of our model for ablation study.

- **Item-based CF** [16]. It's one of the major candidate generation approaches in industry. Collaborative Filtering method generates item-item similarity matrix for recommending.
- **DNN** [4]. A deep neural network based recommendation approach proposed by YouTube. Vectors of videos and users are concatenated and fed into a multi-layer feed forward neural network.
- **GRU4REC** [7]. Hidasi et al. firstly applies recurrent neural network to solve session-based recommendation, which outperforms traditional methods significantly.
- **NARM** [14]. It is an improved version of GRU4REC with global and local attention-based structure. A hybrid encoder with an attention mechanism to model the user's sequential behavior is explored.
- **SHAN** [29]. It incorporates both users' historical stable preferences and recent shopping demands with a hierarchical attention network.
- **BINN** [15]. BINN applies RNN-based methods to encode present consumption motivations and historical purchase behaviors. It generates the unified representation by concatenation operation.
- **SDMMA**. **S**equential **D**eep **M**atching with **M**ulti-head **A**ttention is our multi-head self-attention enhanced model.
- **PSDMMA**. **P**ersonalized **SDMMA** adds user attention module to mine fine-grained personalized information.

Table 2: Comparisons of different models on offline datasets of Taobao and JD.

Models	Taobao				JD			
	HitRate@100	Recall@100	Precision@100	F1@100	HitRate@20	Recall@20	Precision@20	F1@20
Item-based CF	60.27%	3.24%	2.00%	2.43%	67.50%	9.08%	9.41%	8.99%
DNN	60.88%	2.85%	1.83%	2.18%	68.43%	8.93%	9.65%	8.98%
GRU4REC	65.60%	3.66%	2.30%	2.77%	69.44%	9.33%	9.83%	9.29%
NARM	66.97%	3.57%	2.25%	2.70%	70.33%	9.07%	9.58%	9.04%
SHAN	67.30%	3.71%	2.33%	2.80%	70.54%	9.42%	10.02%	9.41%
BINN	67.55%	3.49%	2.20%	2.64%	72.19%	9.38%	9.93%	9.36%
SDMMA	68.24%	3.68%	2.32%	2.79%	70.41%	9.21%	9.72%	9.18%
PSDMMA	69.43%	3.75%	2.37%	2.84%	71.21%	9.21%	9.78%	9.20%
PSDMMAL	70.72%	**3.86%**	2.44%	**2.93%**	73.25%	9.47%	10.13%	9.48%
PSDMMAL-N	**73.13%**	3.83%	**2.45%**	2.92%	**74.33%**	**9.68%**	**10.42%**	**9.72%**
PSDMMAL-NoS	65.41%	3.38%	2.14%	2.56%	70.07%	9.05%	9.60%	9.03%

- **PSDMMAL**. **PSDMMA** combines representations of short-term sessions and **L**ong-term behaviors.
- **PSDMMAL-N**. Based on **PSDMMAL**, during training, we take the following ***N*** items as target classes as Tang and Wang [24] does at the current time step. $N = 5$ in this experiment.
- **PSDMMAL-NoS**. **PSDMMAL** does **No**t contain the embeddings of **S**ide information in short-term sessions and long-term behaviors except for the ID feature of item and user.

4.4 Implementation Details

We implement these deep learning models by distributed Tensorflow[4]. To be fair, all of these models share the same training and testing datasets, as well as input features of items and users and other training hyper-parameters. For training, we use 5 parameter severs (PSs) and 6 GPU (Tesla P100-PCIE-16GB) workers with average 30 global steps/s on offline experiment and we use 20 PSs and 100 GPU workers with average 450 global steps/s on online experiment. Adam optimizer with learning rate 0.001 is used to update parameters and gradient clipping is adopted by scaling gradients when the norm exceeded a threshold of 5. Besides this, we use a mini-batch size of 256 and sequences with similar length are organized to be a batch. Any input feature embedding and model parameters are learned from scratch without pre-training. The learnable parameters are initialized by orthogonal initializer. For the recurrent neural network, we use LSTM of multiple layers with dropout (probability 0.2) and residual network [19] between vertical LSTM stacks. The hidden unit size of LSTM is set to 64 and 128 on offline and online experiments. Different parameter settings are set respectively, because we need to evaluate model performance efficiently on offline experiment. As for the multi-head attention structure, we set the number of heads to 4 and 8 on offline and online experiments respectively. We also use layer normalization and residual adding operation as Vaswani et al. [25] does. The unit size of item embedding vectors, short/long-term behavior vector and user behavior vector keep the same with the one in LSTM. During online serving, our model only costs about **15 milliseconds** for each matching process with **top 300 items** retrieved.

[4]https://www.tensorflow.org/

5 EMPIRICAL ANALYSIS

5.1 Offline Results

Results on offline datasets of different models are shown in Table 2. We select the best results from all the training epochs of these models. In general, deep learning based methods outperform traditional item-based CF dramatically except for YouTube-DNN. Average pooling over item sequences neglects the inherit correlation among items causing hurts on recommending quality (Recall, Precision) severely. GRU4REC and NARM consider the evolution of short-term behaviors. They perform better than original DNN models. The reason why SHAN and BINN can beat GRU4Rec in almost all metrics is that they incorporate more personalized information including long-term behaviors and user profile representation.

Our proposed SDMMA makes use of the multi-head attention structure and has overwhelming superiority over NARM. We conduct a detailed case study in section 5.3 to further explain how multi-head attention could well capture multiple interests. By introducing user profile representation, PSDMMA strengthens the model because users of different types focus on different aspects of items. It make sense that the more accurate the short-term representation is, the more customers can find their interested items in the candidate list. But it's difficult to recommend potentially novel items for a user. More precise preferences should be inferred by considering long-term behaviors.

PSDMMAL can beat all of the models above remarkably by taking long-term preferences into account. Different from SHAN and BINN, it applies a fusion gate to combine short- and long-term behavior representations. The gate has more powerful representation ability for decision than a hierarchical attention structure. SHAN simply applies user profile representation as query vector to decide attention weights of long-term and short-term preference representations. Our proposed PSDMMAL models the specific correlation between them. An interesting case that properly explains the design of fusion gate is shown in section 5.4. PSDMMAL-N is our best variant that takes next 5 items as target classes during training process. It can recall more diversified items that meet requirements of broader users and be more suitable for matching task of recommender system. The results of PSDMMAL-NoS show our model's performances would dramatically decrease without side information.

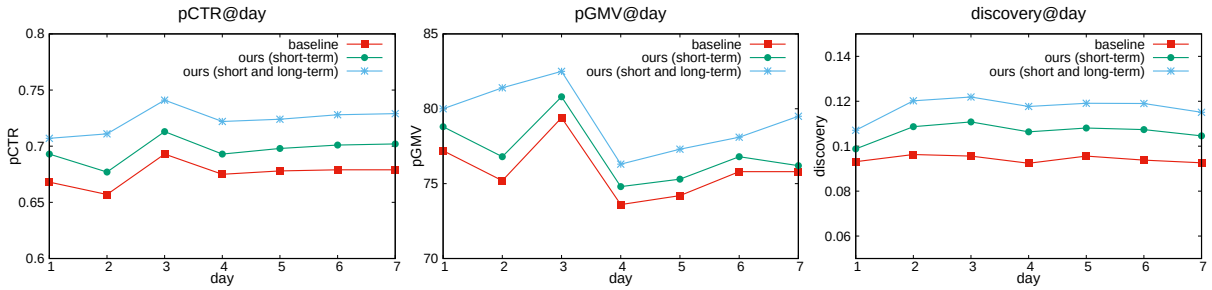

Figure 3: Online performances of our models compared with baseline in 7 days during December 2018.

Table 3: Results of various number of heads. ($K = 100$)

#heads	HitRate@K	Recall@K	Precision@K	F1@k
1	70.00%	3.82%	2.40%	2.88%
2	70.64%	3.83%	2.41%	2.89%
4	**70.72%**	**3.86%**	**2.44%**	**2.93%**
8	70.21%	3.77%	2.37%	2.85%

Table 4: Comparisons of different fusion methods. ($K = 100$)

fusion	HitRate@K	Recall@K	Precision@K	F1@K
multiply	67.09%	3.42%	2.16%	2.59%
concat	69.74%	3.70%	2.34%	2.80%
add	70.24%	3.75%	2.37%	2.84%
gated	**70.72%**	**3.86%**	**2.44%**	**2.93%**

5.2 Online A/B Test

Currently, online matching algorithm in Taobao is a two-staged method. We define *trigger items* as the latest interacted items of a user. Item-based CF firstly generates item-item similarity matrix. The trigger items recall similar items by the similarity matrix. Then these recalled items are re-ranked as matching candidates by a gradient boosting tree according to users' click and purchase logs. Such method is our online baseline and we only replace it with our model as a standard A/B test.

We deploy PSDMMAL-N, our best sequential deep matching model, as well as the version without long-term behaviors on production environment on Mobile Taobao App. Compared with the baseline model, the quality of recommended items inferred from customers' sequence behaviors is much better than the similar items generated by item-based CF. Especially for customers who often casually browse online, our model would recommend novel items to them and attract more eyeballs to encourage potential ordering rates. Figure 3 shows the online results in 7 successive days during December 2018. Two sequential deep models outperformed the baseline with a large margin, where PSDMMAL-N has an overall average improvements of **7.04%**, **4.50%** and **24.37%** with respect to pCTR, pGMV and discovery. Incorporating with long-term behaviors brings much more improvements. Long-term behaviors always indicate personal preferences that can affect customers' current shopping decisions. Note that our sequential matching model has been working well online since December 2018.

5.3 The Effect of Multi-head Attention

We explore the influence of various number of heads in our matching model. Intuitively, representation of the short-term session will get more accurate with the number of heads increasing. Table 3 reports the results on offline Taobao dataset. In this experiment, only the number of heads is different in PSDMMAL and the dimension of model hidden units d is set to 64.

We can observe that changes caused by the head number keep consistent in terms of the four metrics. When the number of heads is less than 4, the effects present positive relationship with the amount of heads. While the number of heads is greater than 4, the results become worse dramatically. We can conclude more heads are not necessarily positive because $d_{\text{head}_i} = \frac{64}{\#\text{heads}}$ would get smaller causing worse representation. In our settings, four heads can get the best results and we visualize the attention weights of the different heads over the short-term session of a user sampled from offline Taobao test dataset in Figure 4. We choose the last hidden output $\boldsymbol{h}_t^u$ of LSTM as the query vector in multi-head attention to get the weights attending to $[\boldsymbol{h}_1^u, ..., \boldsymbol{h}_t^u]$. The weight vector is also the tth row vector of $\boldsymbol{A}_i^u$ in Equation 8. head_1 and head_2 mainly concentrate on the first several items in the session, which are white down jackets. head_3 captures the dress interest and head_4 gives more attention to the jeans.

5.4 The Fusion Gate

Element-wise multiplication, concatenation and addition operation all directly work on unfiltered long-term preference representations and ignore that a small number of preferences in long-term have strong correlation with the current short-term session. These simple combination methods take all the information from long-term preferences and naturally hurt the fusion performance shown in Figure 4 tested on the Taobao offline dataset. In contrast, our proposed gated fusion network accurately captures multi-level user preferences and achieves the best results. Information highly related to current session in the long-term preference can be fused with current short-term vector.

For better explanation of the gated fusion, we use a real-world case of a sampled user at Taobao to interpret the function of gate. As shown in Figure 5, $\mathcal{R}^u$ contains items recommended by our model, which are clicked by the user simultaneously. We can see the user is browsing kinds of glasses including red wine and champion glasses. Our model can directly recommend champion glasses because they are related to the last clicks in his short-term session $\mathcal{S}^u$. It means he is more probably interested in champion glasses at present and the gate allows this information remain. Meanwhile, our gated fusion could capture the most relevant items **red wine** among his massive long-term behaviors $\mathcal{L}^u$, which also includes many irrelevant clicks such as beer, paring knife and small plate, and combine with the short-term session items **red wine glasses** to

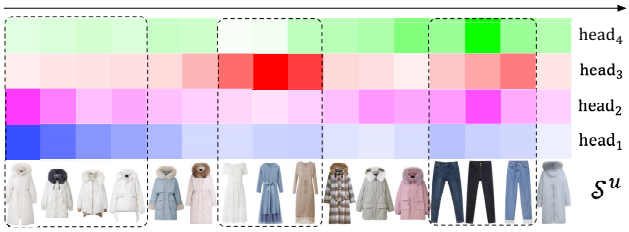

Figure 4: Visualization of attention weights (the last hidden output of LSTM as query vector to get the weights) from head_1 to head_4 over a short-term session $\mathcal{S}^u$ sampled from offline test dataset of Taobao.

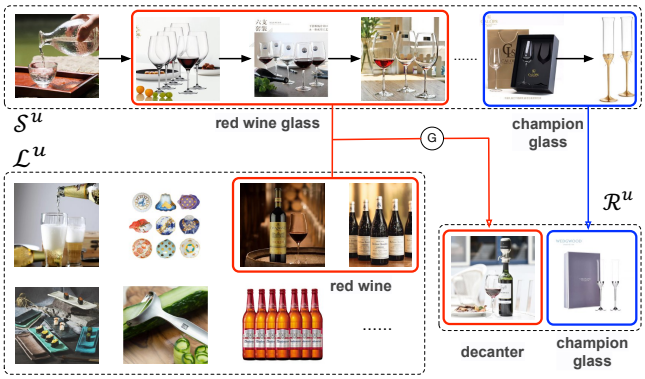

Figure 5: A short-term session $\mathcal{S}^u$ and long-term behaviors $\mathcal{L}^u$ from a sampled user on our online system. $\mathcal{R}^u$ is the set of items recommended by our model, which are also clicked by the user.

generate the recommended item **red wine decanter**. The case shows our gate module has effective and accurate fusion ability.

6 CONCLUSIONS

In this paper, we propose a sequential deep matching model to capture users' dynamic preferences by combining short-term sessions and long-term behaviors. We employ multi-head self-attention to capture multiple interests in short-term sessions and long-short term gated fusion network to incorporate long-term preferences. Extensive offline experiments show the effectiveness of our model. The matching model is successfully deployed on Taobao's recommender system with improvements in term of important commercial metrics.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Ting Bai, Pan Du, Wayne Xin Zhao, Ji-Rong Wen, and Jian-Yun Nie. 2019. A Long-Short Demands-Aware Model for Next-Item Recommendation. *arXiv preprint arXiv:1903.00066* (2019).

[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. ACM, 108–116.

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

[5] Disheng Dong, Xiaolin Zheng, Ruixun Zhang, and Yan Wang. 2018. Recurrent Collaborative Filtering for Unifying General and Sequential Recommender. In *IJCAI*. 3350–3356.

[6] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *IJCAI*. 2301–2307.

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[8] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *WSDM*. ACM, 573–581.

[9] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. ACM, 505–514.

[10] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDE*. IEEE, 197–206.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[14] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. ACM, 1419–1428.

[15] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors. In *KDD*. ACM, 1734–1743.

[16] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.

[17] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*. ACM, 1831–1839.

[18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[19] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182* (2017).

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

[21] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. ACM, 285–295.

[23] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H Chi. 2019. Towards Neural Mixture Recommender for Long Range Dependent User Sequences. *arXiv preprint arXiv:1902.08588* (2019).

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

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[26] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. *arXiv preprint arXiv:1803.02349* (2018).

[27] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *SIGIR*. ACM, 403–412.

[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[29] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In *IJCAI*.

[30] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *arXiv preprint arXiv:1806.01973* (2018).

[31] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*. ACM, 582–590.

[32] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* (2018).

[33] Wei Zhao, Benyou Wang, Jianbo Ye, Yongqiang Gao, Min Yang, and Xiaojun Chen. 2018. PLASTIC: Prioritize Long and Short-term Information in Top-n Recommendation using Adversarial Training. In *IJCAI*. 3676–3682.

[34] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *KDD*. ACM, 1079–1088.