

Micro-Behavior Encoding for Session-based Recommendation

Jiahao Yuan¹, ✉Wendi Ji¹, Dell Zhang^{§2,3}, Jinwei Pan¹, and Xiaoling Wang¹

¹Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

²Birkbeck, University of London, UK

³ByteDance AI Lab, London, UK

{jhyuan, jwpan}@stu.ecnu.edu.cn, {wdji, xlwang}@cs.ecnu.edu.cn, dell.z@ieee.org

Abstract—Session-based Recommendation (SR) aims to predict the next item for recommendation based on previously recorded sessions of user interaction. The majority of existing approaches to SR focus on modeling the transition patterns of *items*. In such models, the so-called *micro-behaviors* describing how the user locates an item and carries out various activities on it (e.g., *click*, *add-to-cart*, and *read-comments*), are simply ignored. A few recent studies have tried to incorporate the *sequential* patterns of micro-behaviors into SR models. However, those sequential models still cannot effectively capture all the inherent interdependencies between micro-behavior operations. In this work, we aim to investigate the effects of the micro-behavior information in SR systematically. Specifically, we identify two different patterns of micro-behaviors: “sequential patterns” and “dyadic relational patterns”. To build a unified model of user micro-behaviors, we first devise a multigraph to aggregate the sequential patterns from different items via a graph neural network, and then utilize an extended self-attention network to exploit the pair-wise *relational* patterns of micro-behaviors. Extensive experiments on three public real-world datasets show the superiority of the proposed approach over the state-of-the-art baselines and confirm the usefulness of these two different micro-behavior patterns for SR.

Index Terms—session-based recommendation, micro-behavior modeling, graph neural networks, self-attention mechanism.

I. INTRODUCTION

Recommender systems are a subclass of *information retrieval* applications where the information need is typically represented by not an explicit query but a user’s profile and context [1]. The core functionality of such online systems is to predict the rating or preference the users would give to different items by analyzing their past behaviors. In many commercial websites that employ recommender systems, the sequence of a user’s interactions can be naturally segmented into a number of sessions each of which occurs within a certain period of time and reflects the user’s interest at that moment. Due to user privacy concerns, real-life recommender systems are often unable to identify each user and track their long-term interests. However, users usually have a very specific and clear short-term intention when they visit an e-commerce or other online service websites [2]. Therefore, *Session-based Recommendation* (SR) which aims to predict the next item

for a user in the current session, has recently attracted a lot of attention from Internet companies [3].

Most existing methods for SR focus on modeling the transition patterns of *items* within a session through techniques like Markov chain [4], Recurrent Neural Network (RNN) [5], attention mechanism [6], and Graph Neural Network (GNN) [7, 8, 9]. Although the field of SR has witnessed significant progress in the last few years, it is still facing some challenging problems. Firstly, those methods only analyze the sequence of items in a session but discard the detailed operations carried out by the user on each item. However, the sequence of items, also known as *macro-behaviors* [10], could not paint the complete picture. Compared with the coarse-grained sequence of items, the fine-grained sequence of different operations performed on each item should be able to reflect the user’s intentions and preferences more precisely. In this paper, we refer to such operations with respect to a particular item within a session as *micro-behaviors*, and try to make effective use of them for SR. Furthermore, although GNN-based methods have achieved exciting results and offered a promising direction for modeling the transition patterns of macro items, they cannot incorporate multiple micro-operations of the same item with their graph construction and information aggregation methods. Secondly, to the best of our knowledge, a few recently emerged studies attempting to incorporate micro-behaviors into SR all just model the sequential pattern of micro-behaviors with RNN [10, 11, 12]. However, micro-behaviors are often correlated with each other, and it would be very difficult for RNN-based sequential models to capture such interdependencies beyond the immediate predecessor/successor relations. In our opinion, it would be beneficial to analyze the *relational* patterns of micro-behaviors in addition to the *sequential* patterns.

Fig. 1 shows an example with two imaginary users. At the level of macro-behaviors, user 1 and user 2 are not distinguishable as they have exactly the same session: the sequence of items are iPhone, MacBook, AirPods, Apple-Watch and Magic-Keyboard. By contrast, at the level of micro-behaviors, user 1 and user 2 look quite different. Firstly, the former was probably buying computer hardware for her work since she bought MacBook and Magic-Keyboard, while the latter might be only interested in the products made by Apple Inc. Thus, the micro-operations in different items can help to distinguish

[§]Dell Zhang is currently on leave from Birkbeck, University of London and working full-time for ByteDance AI Lab, London, UK.

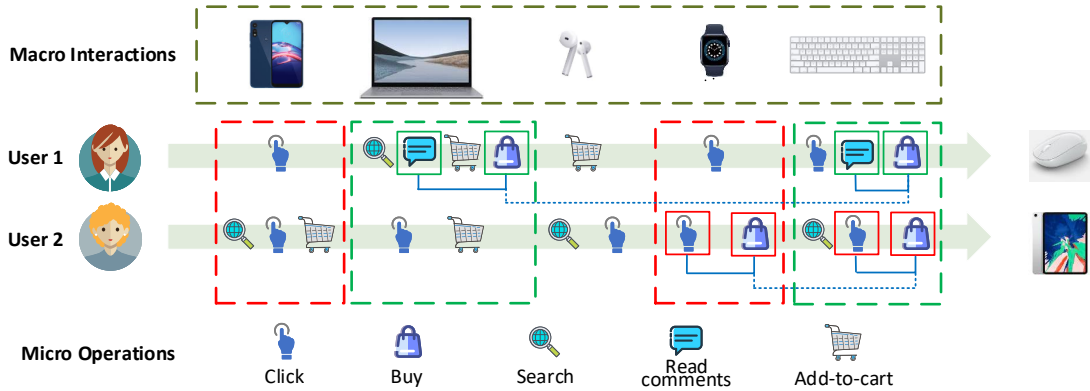


Fig. 1. Given the same item sequence with different operations, our model can make different predictions for the next item.

the intentions of the user within a session. Secondly, user 1 tends to read the customer comments before making a purchase, while user 2 goes straightaway to place order. Such micro-behavioral differences would not be easily captured by a sequential model, but they could be explicitly modeled using dyadic relational patterns $\langle \text{read-comments}, \text{purchase} \rangle$ and $\langle \text{click}, \text{purchase} \rangle$. In other words, dyadic relational patterns may have more discriminative power than sequential patterns for the purpose of SR.

According to the above observations, the sequential patterns are reflected in the successive micro-operations of the single item, which relate to users' preference of the item. The dyadic relational patterns, on the other hand, convey the pair-wise semantics of micro-behaviors which contain different meanings about the items. In this paper, we propose a novel framework to Encode Micro-Behaviors in Session-based Recommendation (named **EMBSR**¹) to jointly learn those two different patterns. Specifically, EMBSR captures the user's intention and preference in an interaction session by integrating the sequential patterns with the (dyadic) relational patterns of micro-behaviors, and thus achieves significant performance gains in recommendation accuracy. Firstly, we propose a new encoding scheme for converting a session into a multigraph in order to incorporate micro-behavior in GNNs. The graph construction is the premise of the proposed framework to aggregate the information of micro-behaviors since a macro item may correspond to multiple successive micro-operations. Secondly, we employ Gated Recurrent Unit (GRU) [13] to describe the sequential pattern for each single item, which has aggregated micro-behavior information in iterations of GNNs. Next, inspired by self-attention with relative position representations [14], we develop an operation-aware self-attention mechanism which incorporates the dyadic operation representation into the self-attention network for exploiting the pair-wise semantics of micro-behaviors. Finally, we combine such different representations of the user's interest with the fusion gating mechanism to predict the next item.

¹The implementation has been released at <https://github.com/SamHaoYuan/EMBSR>

The main contributions of this paper are as follows.

- We consider a user's interaction session as a fine-grained sequence of micro-behaviors and discover two different types of patterns (sequential and relational) for the modeling of micro-behaviors in SR.
- We design a novel encoding scheme that transforms a session into a directed multigraph and employ a novel aggregation stage to incorporate the sequential patterns in the graph using GRUs.
- We propose a novel operation-aware self-attention mechanism to encode the dyadic relations of micro-behaviors, and combine the valuable information embedded in a session's sequential and relational patterns to enhance the prediction of next item.
- We conduct extensive experiments on three public real-world datasets to demonstrate that our proposed model can outperform all the state-of-the-art baselines for SR significantly.

II. PRELIMINARIES

In this section, we introduce the preliminary knowledge about GNNs and give its message passing paradigm, and then we formulate the problem to address in this paper.

A. Graph Neural Network

GNN generalizes traditional neural networks to graph for capturing structural information. Take advantage of the powerful modeling capabilities for nodes and edges, the task of SR is formulated as the graph classification problem.

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a graph with nodes \mathcal{V} and edges \mathcal{E} , (u, e, v) denote an edge $e \in \mathcal{E}$ where $u \in \mathcal{V}$ is the source node and $v \in \mathcal{V}$ is the target node. Here we use a symbol e to distinguish different edge from $u \rightarrow v$. Let $x_v \in \mathbb{R}^{d_1}$, $x_u \in \mathbb{R}^{d_1}$ denote the representation of two nodes respectively, $w_e \in \mathbb{R}^{d_2}$ denote the edge feature. Generally, the process of GNNs to learn node embeddings in the $l+1$ -th layer can be formulated as follows:

$$\begin{aligned}
 m_e^{l+1} &= f_m(x_v^l, x_u^l, w_e^l), (u, e, v) \in \mathcal{E} \\
 a_e^{l+1} &= f_a(\{m_e^{l+1} : (u, e, v) \in \mathcal{E}\}) \\
 x_v^{l+1} &= f_u(x_v^l, a_e^{l+1})
 \end{aligned}$$

In the above equation, f_m is the message function defined on each edge to generate a message by using the edge feature and its endpoint feature, and m_e^{l+1} denotes the message information of this edge. f_a is an aggregation function that aggregates neighbors' information of the node along edges, a_e^{l+1} denotes the aggregated information of all edges of this node. Different information can be considered here based on the incoming or the outgoing edges of the node. Lastly, an update function f_u is applied to generate a new representation of the node, and x_v^{l+1} denotes the novel representation of the node v . After information propagates for multiple layers, we can get a new representation of each node (or edge sometimes) and use it in the downstream task.

Note that all notations in this subsection are only used here to explain the preliminary knowledge of GNN and may be inconsistent with the following expression.

B. Problem Statement

Let V and O denote the set of distinct items involved in all sessions and the set of distinct operations a user can perform, respectively. Given the sequence of micro-behaviors in a session, $S_t = \{s_1, s_2, \dots, s_t\}$, where s_i is the i -th micro-behavior of the user and t is the maximum length of the session. Specifically, $s_i = (v_i, o_i)$ is a tuple that combines an item and its corresponding operations.

To capture the fine-grained preference for the item, we first merge the successive micro-behaviors that characterize the same item to get the chronological sequence of macro-items $S_t^v = \{v^1, v^2, \dots, v^n\}$ and its corresponding micro-operation sequence $S_t^o = \{o^1, o^2, \dots, o^n\}$, where $n \leq t$ denote the length of the macro-item sequence. For each item $v^i \in S_t^v$, its operation sequence is $o^i = \{o_1^i, o_2^i, \dots, o_k^i\}$, because one item usually has multiple corresponding operations. That is to say, we treat the successive micro-behaviors of the same item as a whole. The goal of the proposed model is to predict the next macro-item v^{n+1} . Noting that we do not directly predict the next micro-behavior s_{t+1} or its item v_{t+1} , since the last macro-item may also have multiple micro-behaviors. In other words, there is a high probability of $v_t = v_{t+1}$, which leads to information leakage.

To achieve this goal, the proposed model learns to generate a session representation m based on the given session, and uses it to calculate a score \hat{y}_i for each item $v \in V$. Finally, the items of the top- K scores will be returned to the user as recommendations. For clarity, Tab. I summarizes the main notations and their meanings used in this paper.

III. RELATED WORK

A. Session-based Recommendation

Since the long-term user profile is unknown, the task of SR is limited to the context within the sessions. Hence, for conventional methods, simple matrix factorization [15, 16] and Item-KNN [17] without considering the order of items are not suitable for SR. Thus, FPMC [4] used tensor factorization that combines Markov chain to simulate the sequential behavior

TABLE I
SUMMARY OF NOTATIONS

Notations	Descriptions
V, O	The item set and operation set
S_t, S_t^v, S_t^o	the given session, the corresponding macro-item sequence, the corresponding micro-operation sequence
s_i, v_i, o_i	the i -th micro-behavior, item, and operation of the given session, where $s_i = (v_i, o_i)$
v^i, o^j, o_j^i	the i -th macro item, the micro-operation sequence of v^i , the j -th micro-operation in o^i
$\mathcal{G}_t, \mathcal{V}_t, \mathcal{E}_t$	the directed multigraph, the node set, the edge set
M^V, M^O	embedding metrics of item and operation
M^P, M^R	embedding metrics of position and dyadic relation
S_t^u, u_i	the distinct item set of S_t , the distinct item in S_t (it is also the node in the multigraph)
e_{u_i}, e_{o_i}	the embeddings of u_i and o_i
v_s, e_{u_s}	the star node and its embedding
$\tilde{h}^i, \tilde{h}_j^i, \tilde{h}_o$	the hidden state of the operation o_j^i , the sequential encoding of o^i , the sequential pattern encoding of the micro-operation in the given session
$m_{i+}^l, m_{i-}^l, m_i^l$	the message for incoming edges, outgoing edges and all edges of the node u_i at layer l
$E_{in}(i), E_{out}(i)$	the set of incoming edges and outgoing edges of the node u_i
h^i, h^f	the hidden layer of GNN, the final representation of satellite nodes
$M_{st}, r_{ij}, e_{r_{ij}}$	the relation matrix of the session S_t , the dyadic relation index between o_i and o_j , the embedding of the dyadic relation r_{ij}
X_t, x_i, x_s	the input embeddings for the operation-aware self-attention mechanism, the embedding of the tuple s_i , the embedding of the star node
e_{p_i}, e_{ij}	the embedding of the i -th position, the intermediate value to measure the correlation between x_i and x_j
z_s, m	the output of the operation-aware self-attention mechanism, the final output of the proposed approach

between two adjacent clicks. Kamehkhosh et al. [18] combined Markov chain with association rules. Recently, methods based on nearest-neighbors obtain competitive performance. SKNN [19] is a session-based k-nearest-neighbors approach, which considers the sessions that contain any item of the current session as neighbors. STAN [20] is an extension version that additionally considers other factors. However, this kind of approach is generally limited to represent complex dependencies.

Neural network methods are proposed to capture the dynamic preferences from user's sequential behaviors, which have gained momentum in SR. GRU4Rec [21] first introduced GRU to model user sessions. GRU4Rec+ [22] proposed a data

augmentation method and took the changes in user behavior over time into account. Moreover, NARM [5] proposed an encoder-decoder architecture based on RNN and attention mechanism. STAMP [6] used the attention mechanism to capture both the users’ long-term and short-term interests. Bert4Rec [23] employed the deep bidirectional self-attention to identify the correlations of items. These attention-based models effectively capture both the user’s general and current interest. CSR [24] and CoSAN [25] incorporates collaborative neighborhood information into neural SR models.

Recently, GNNs have been applied to SR to capture the complex transition patterns. SR-GNN [7] first introduced a gated GNN into this task, which converts a session to graph-structured data. Furthermore, GC-SAN [8] proposed a graph contextualized self-attention model, which utilizes both GNNs and the self-attention mechanism. FGNN [26] replaced the gated GNN with multi-layered weighted graph attention networks [27]. To capture the long-term dependencies, LESSR [28] used an GAT layer to learn the global dependency by propagating information along long-range edges, and SGNN-HN [9] introduced a star GNN for the problem. For global information, GCE-GNN [29] build a global graph to learn the global-level item embeddings by modeling pair-wise item-transitions over session, and DHCN [30] adopted a dual channel hypergraph convolutional network.

To summarize, a majority of approaches for SR have focused on exploring the complex transition of macro-items and neural methods, especially GNN-based methods, which show promising potential to model dependencies of items. However, they all neglect user micro-behaviors in sessions, which reveals the fine-grained preferences or attitudes of the user. Compared with these macro-behavior models, the proposed framework investigates the effects of the micro-behavior information in SR systematically and explores two different patterns of micro-behaviors.

B. Micro-Behaviors in Recommendation

Micro-behaviors are also integrated in the task of conventional recommendation (called multi-behavior-based recommendation [31]), which aims to leverage the micro-behaviors to improve the recommendation performance on the target behavior (e.g. purchase).

Quadrana et al. [3] proposed behavioral factorization, which extends the collective matrix factorization [32] to handle different behaviors in online social network (comment, re-share, and create-post). Liu et al. [33] employed behavior-specific transition matrices in recurrent and time-aware Log-BiLinear [34] model to capture the properties of different types of behaviors. Wan and McAuley [35] utilized tensor decomposition framework to model monotonic behavior chains where user behaviors are supposed to follow the same chain. Gao et al. [31] proposed to correlate the model prediction of each behavior type in a cascaded way and trained the whole model in a multi-task manner. Moreover, Lo et al. [36] analyzed the purchasing behavior of users to determine short-term and long-term signals in user behavior that indicate

higher purchase intent. Multi-behaviors are also considered as features and extracted from user clickstreams to help predict purchase [37, 38]. Furthermore, from the perspective of learning, there are many works exploiting multi-behaviors as auxiliary action for sampling [39, 40, 41]. However, these works verify the effectiveness of multi-behaviors to help model the target behavior in conventional recommendation scenarios but have not explored the complex patterns in SR.

Recently, a few works have begun to take micro-behaviors into consideration in SR. Zhou et al. [10] first adopted RNN to model micro-behaviors, and Gu et al. [11] extended it to a hierarchical architecture to distinguish the differences between micro-operations and macro-items. These RNN-based models ignore the different dependencies between items and operations. To tackle this problem, MKM-SR [12] fed the operation sequence and the item sequence of a session into RNN and GNN, respectively. However, MKM-SR only considers the transition of macro items in GNNs and cannot incorporate the micro-behavior information in the iterative process of GNNs. It treated items and operations separately and only concatenated them for final session representations. In addition, all those methods only consider the sequential pattern of the micro-operation and are very difficult to capture the interdependencies of dyadic relational patterns. In contrast, our work devises a multigraph to aggregate the sequential patterns and then utilizes an extended self-attention network to exploit the pair-wise relational patterns of micro-behaviors.

IV. APPROACH

In this section, we present the proposed model in detail. We first summarize the pipeline of the proposed model. Then we describe the two major components of the proposed method to encode sequential patterns and dyadic relational patterns of micro-behaviors. Lastly, we introduce a prediction layer to generate the session representation with a fusion gating mechanism.

A. Model Overview

The framework of the proposed model is illustrated in Fig 2. For an input session, we transform each micro-behavior into item embedding, operation embedding, and relation embedding. Then for encoding sequential patterns, we convert the macro-item sequence of the input session into a multigraph with a star node and use a GRU for the micro-operation sequence to get the feature on edges. Here, we learn a new item embedding for each macro-item by the star graph. For encoding dyadic relational patterns, the new item embeddings incorporating the relation embeddings are fed into an operation-aware self-attention mechanism to exploit the pair-wise semantics of micro-behaviors. Lastly, the session is represented by combining a general preference and a recent interest in the session with a fusion gating network for generating the scores on all candidate items.

B. Encoding Sequential Patterns

Based on the basic principle of SR [3], the premise of obtaining a session representation is to learn each object’s

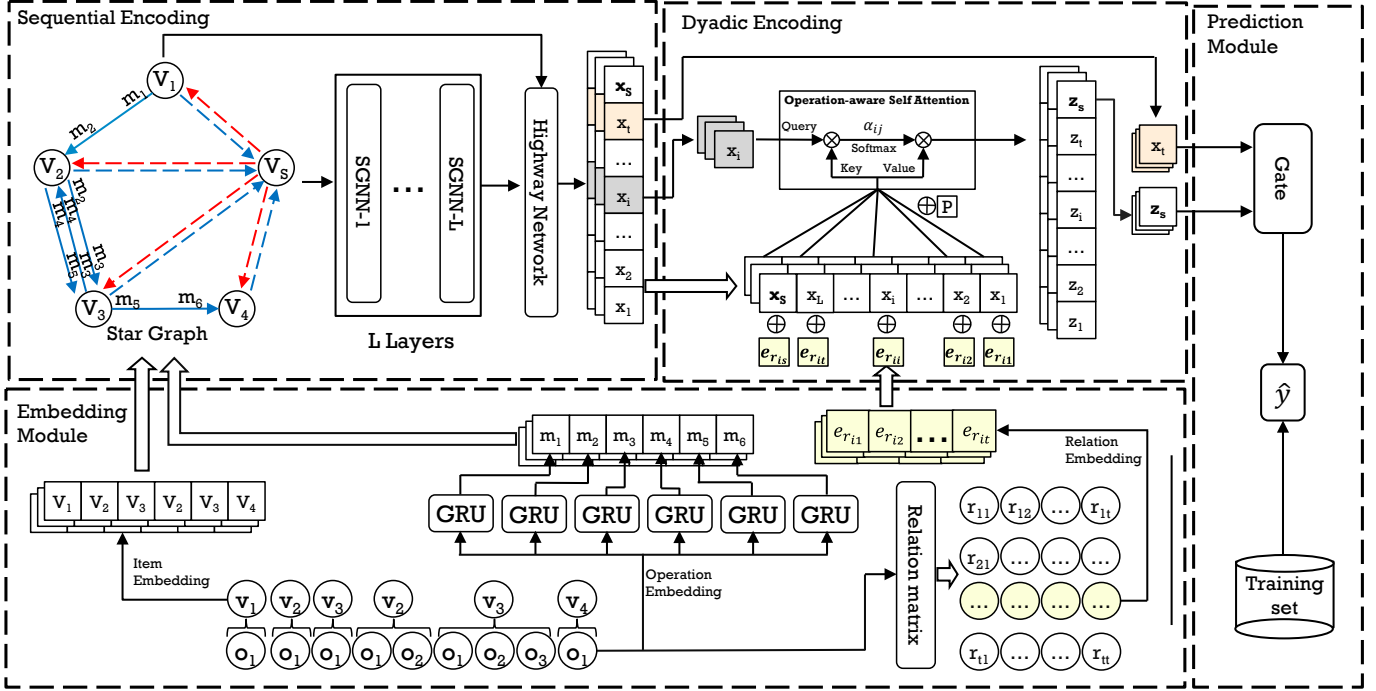


Fig. 2. Overview of the proposed EMBSR model.

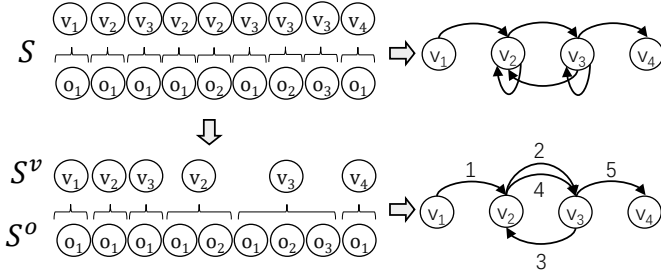


Fig. 3. Different way of graph construction. We use the second way that transforming the session into a multigraph

embedding in the session. In the setting of this paper, the object is the micro-behaviors, including items and operations. Compared with the operation sequence of a session, the transition pattern of the item sequence is more complex and does not simply exhibit the sequential pattern [12]. Thus, we adopt GNNs to model the macro-item sequence and aggregate micro-operation information by GRU in it.

Next, we introduce how we convert the input session into a multigraph and present how to propagate information between the macro-items and micro-operations via the proposed model.

1) *Graph Construction*: We convert the macro-item sequence to a directed multigraph that preserves the edge order. Specifically, For each macro-item sequence $S_t^v = \{v^1, v^2, \dots, v^n\}$, we model it as a directed multigraph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$. In \mathcal{G}_t , each node is the distinct item in S_t^v and each directed edge $(v^i, v^{i+1}) \in \mathcal{E}_t$ is the transition $v^i \rightarrow v^{i+1}$ in S_t^v . It is worth noting that the graph is a multigraph since there

may be multiple transitions between the same items. The edges are ordered by the time of their occurrences in the session in order to distinguish the micro-operation information that their endpoints have at different times.

In Fig. 3, we use an example to illustrate the way of converting an input session to a multigraph. For the input session S , it has four different items $\{v_1, v_2, v_3, v_4\} \in V$ and three different micro-operations $\{o_1, o_2, o_3\} \in O$. First of all, we merge the successive micro-behaviors with the same item to get the macro-item sequence $S^v = \{v_1, v_2, v_3, v_2, v_3, v_4\}$ and its corresponding sequence list $S^o = \{(o_1), (o_1), (o_1), (o_1, o_2), (o_1, o_2, o_3), (o_1)\}$. Then we convert S^v to a multigraph and record the order by giving each edge an integer attribute to help propagate the micro-operation information of endpoints. Since the same item has a different micro-operation sequence on different positions, the multigraph ensures that the neighbor nodes pass the different message along edges based on the different micro-operation sequences.

Inspired by [9], we add a *star node* to capture the long-range information by propagating information from non-adjacent items, which is v_s in Fig. 2. For convenience, we call other nodes from the macro-item sequence *satellite node*. The star node has a bidirectional edge with each satellite node, but we update it in different way from satellite nodes.

2) *Initialization of Node Embedding*: Before passing the nodes into the proposed model, we construct an embedding matrices $M^V \in \mathbb{R}^{|V| \times d}$ for items, where d is the latent dimension. We use $S_t^u = \{u_1, u_2, \dots, u_c\}$ to denote the distinct items set in the input session S_t , where c is the number of

distinct items in S_t . Next, for each distinct item $u_i \in S_t^u$, we get its embeddings $e_{u_i} \in \mathbb{R}^d$ by the lookup of the item embedding matrix and use it as initial embedding of satellite nodes in \mathcal{G}_t . So the initial hidden layer h^0 is:

$$h^0 = \{e_{u_1}, e_{u_2}, \dots, e_{u_c}\} \quad (1)$$

As for the star node, we apply an average pooling on the satellite nodes to get its initialization:

$$e_{u_s} = \frac{1}{c} \sum_{i=1}^c e_{u_i} \quad (2)$$

3) *Sequential Information of Micro-Operation*: For each micro-operation $o_j^i \in S_t^o$, we also construct an embedding matrix $M^O \in \mathbb{R}^{|\mathcal{O}| \times d}$ and get the micro-operation embedding $e_{o_j^i} \in \mathbb{R}^d$ by the lookup of M^O . So for each item $v^i \in S_t^v$, we have the representation of its micro-operation sequence o^i as $\{e_{o_1^i}, e_{o_2^i}, \dots, e_{o_k^i}\}$

To capture the sequential pattern of the micro-behavior, we apply RNN to the embedding of the micro-operation sequence. To avoid the problem of gradient vanishing, we use GRU, which is an improved variant of RNN. Then the j -th step of the GRU is:

$$\tilde{h}_j^i = GRU(e_{o_j^i}, \tilde{h}_{j-1}^i; \Phi_{GRU}) \quad (3)$$

where Φ_{GRU} denotes all GRU parameters and \tilde{h}_j^i is the hidden state of the operation o_j^i . For each micro-operation sequence $o^i \in S_t^o$, we use $\tilde{h}^i = \tilde{h}_k^i$ to represent the sequential information of it, where \tilde{h}_k^i is the last hidden state of GRU. Thus, we obtain the learned embeddings as

$$\tilde{h}_o = \{\tilde{h}^1, \tilde{h}^2, \dots, \tilde{h}^n\} \quad (4)$$

where $\tilde{h}_o \in \mathbb{R}^{n \times d}$ denote the whole sequential pattern of the micro-operation in the input session. Since the number of \tilde{h}_o is the same as the number of macro-item sequence, we can match different sequential information for each node based on the order of the edges.

4) *Aggregation Stage*: The aggregation state includes two processes. The first process is to generate a message by the message function for each edge. The other process is to aggregate neighbors' information of each node along its edges by aggregation function. Our main idea for the information propagation is to consider the micro-operation influence on user's preference of items. Thus, the same node will pass the different messages along the different edges based on its micro-operation sequence in that position. So the whole process for each satellite node can be formulated as follow:

$$\begin{aligned} m_{i+}^{l+1} &= f_m^+(\{e_{u_j}^l, \tilde{h}_j : (u_j, u_i) \in E_{in}(i)\}) \\ m_{i-}^{l+1} &= f_m^-(\{e_{u_j}^l, \tilde{h}_j : (u_i, u_j) \in E_{out}(i)\}) \end{aligned} \quad (5)$$

where $e_{u_j}^l$ is the representation of the node u_j at layer l , $E_{in}(i)$ and $E_{out}(i)$ denote the set of incoming edges and outgoing edges of the node u_i respectively. The message functions f_m^+ and f_m^- are used to compute the message to be propagated

from the neighbor node along the incoming and outgoing edges, which are defined as:

$$\begin{aligned} f_m^+(e_{u_j}, \tilde{h}_j) &= W_m^+([e_{u_j}; \tilde{h}_j]) + b_m^+ \\ f_m^-(e_{u_j}, \tilde{h}_j) &= W_m^-([e_{u_j}; \tilde{h}_j]) + b_m^- \end{aligned} \quad (6)$$

where $W_m^+, W_m^- \in \mathbb{R}^{2d \times d}$ and $b_m^+, b_m^- \in \mathbb{R}^d$ are learnable parameters, $[\cdot]$ denote the concatenation operation. Hence, we can obtain $m_i^{l+1} \in \mathbb{R}^{|\mathcal{E}(i)| \times d}$ to denote the message for all edges of the node u_i . Noting that we do not consider the edge with the star node here to avoid destroying the structural information of the original session. After that, we aggregate messages of all edges for node u_i by

$$a_i^{l+1} = \left[\sum_{k=1}^{|E_{in}(i)|} m_{i+,k}^{l+1}; \sum_{k=1}^{|E_{out}(i)|} m_{i-,k}^{l+1} \right] \quad (7)$$

where $m_{i,k}^{l+1} \in \mathbb{R}^d$ denotes the k -th element in m_i^{l+1} and $a_i^{l+1} \in \mathbb{R}^{2d}$ denotes the aggregated information of u_i .

5) *Update Stage*: The update stage is to update the node feature by using the aggregation information for each node. We feed a_i^{l+1} and the node u_i 's previous embedding into the gated GNN [42] as follows:

$$\begin{aligned} \tilde{z}_i^{l+1} &= \sigma(W_z a_i^{l+1} + U_z e_{u_i}^l) \\ r_i^{l+1} &= \sigma(W_r a_i^{l+1} + U_r e_{u_i}^l) \\ \tilde{e}_{u_i}^{l+1} &= \tanh(W_u a_i^{l+1} + U_u (r_i^{l+1} \odot e_{u_i}^l)) \\ \hat{e}_{u_i}^{l+1} &= (1 - \tilde{z}_i^{l+1}) \odot e_{u_i}^l + \tilde{z}_i^{l+1} \odot \tilde{e}_{u_i}^{l+1} \end{aligned} \quad (8)$$

where $W_z, W_r, W_u \in \mathbb{R}^{2d \times d}$ and $U_z, U_r, U_u \in \mathbb{R}^d$ are trainable parameters of the network. $\sigma(\cdot)$ denotes the *sigmoid* function and \odot is the element-wise multiplication. In addition, \tilde{z}_i^{l+1} and r_i^{l+1} are update gate and reset gate respectively, which controls how much information should be preserved or updated between two layers. In this way, information from satellite nodes can be propagated by the GNN.

Next, we consider the connection from the star node to explicitly capture the overall information of the session. For each satellite node, we adopt a gating network to decide how much information should be propagated from the star node and the adjacent nodes, which is as:

$$\begin{aligned} \alpha_i^{l+1} &= \frac{(W_{q1} \hat{e}_{u_i}^{l+1})^T W_{k1} e_{u_s}^l}{\sqrt{d}} \\ e_{u_i}^{l+1} &= (1 - \alpha_i^{l+1}) \hat{e}_{u_i}^{l+1} + \alpha_i^{l+1} e_{u_s}^l \end{aligned} \quad (9)$$

where $W_{q1}, W_{k1} \in \mathbb{R}^{d \times d}$ are both parameter matrices. This gating network determines how to selectively integrate the information from $\hat{e}_{u_i}^{l+1}$ and the former star node $e_{u_s}^l$ to generate the new representation of satellite node u_i .

For updating the star node, we use attention mechanism to assign different weight to the satellite nodes by regarding the star node as *query*:

$$\begin{aligned} \beta_i &= \text{softmax}\left(\frac{(W_{k2} e_{u_i}^{l+1})^T W_{q2} e_{u_s}^l}{\sqrt{d}}\right) \\ e_{u_s}^{l+1} &= \sum_{i=1}^{l_s} \beta_i e_{u_i}^{l+1} \end{aligned} \quad (10)$$

where $W_{q2}, W_{k2} \in \mathbb{R}^{d \times d}$ are the learnable parameters, β_i is the weight of the node u_i . Moreover, we apply a highway network [43] to selectively obtain information from the item embeddings before and after the stacked GNN layers, which is denoted as:

$$\begin{aligned} g &= \sigma(W_g[h^0; h^{last}]) \\ h^f &= g \odot h^0 + (1 - g) \odot h^{last} \end{aligned} \quad (11)$$

where $W_g \in \mathbb{R}^{2d \times d}$ is the trainable parameter. By this highway network, we can obtain the final representation of satellite nodes as h^f and the corresponding star node $e_{u_s}^{last}$ (denoted as e_{u_s} for brevity).

C. Encoding Dyadic Relational Patterns

To encode dyadic relation patterns of micro-behaviors and generate the session representation, we should aggregate the embedding of all micro-behaviors in the input session. In the above GNNs, we have already integrated the sequential pattern of the micro-operation into the representation of the items, but the relational pattern of the micro-operation has still been neglected. In this subsection, we introduce a method to encode dyadic micro-operation and present a novel operation-aware self-attention mechanism.

1) *Dyadic Micro-Operation Encoding*: In an attempt to model the pairwise relations between all the operations in the input session, we create a relation matrix of operation $M^R \in \mathbb{R}^{|O|^2 \times d}$. Each vector with dimension d in M^R represents a dyadic encoding of the operation pair. For example, suppose that we have 10 different micro-operations, combining them in pairs, there will be 100 different couples, e.g., $\langle \text{click}, \text{purchase} \rangle$, $\langle \text{click}, \text{read-comments} \rangle$. Therefore, for the operation sequence $O_t = \{o_1, o_2, \dots, o_t\}$ of the session S_t , the relation matrix of it is:

$$M_{S_t} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1t} \\ r_{21} & r_{22} & \dots & r_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ r_{t1} & r_{t2} & \dots & r_{tt} \end{bmatrix}$$

After retrieving the relation matrix M^R , we get the embedding $e_{r_{ij}} \in \mathbb{R}^d$ for $r_{ij} \in M_{S_t}$.

2) *Operation-Aware Self-Attention Mechanism*: To capture the dyadic information in micro-behaviors, we propose an extension to self-attention, which incorporates the pairwise operation embedding in the sequence. Given the micro-behavior session S_t , we use the embedding sequence $X_t = \{x_1, x_2, \dots, x_t\}$ as the input vectors, where $x_i \in \mathbb{R}^d$ is the representation for the tuple $s_i = (v_i, o_i) \in S_t$. It is calculated by:

$$x_i = e_{v_i} + e_{o_i}. \quad (12)$$

where $e_{v_i} \in \mathbb{R}^d$ is the item embedding from the corresponding satellite nodes $h^f \in \mathbb{R}^{l \times d}$, e_{o_i} is the operation embedding by lookup the matrix M^O . Since the star node has fused the entire session's information in the GNNs, inspired by [44], we use

it as the representation of the target item and suppose that it has the same micro-operation with the next item:

$$x_s = e_{u_s} + e_{o_{t+1}} \quad (13)$$

where $x_s \in \mathbb{R}^d$ can be used to denote the user's global preference and is concatenated on the end of the X_t .

Then we create a learn-able embedding matrix $M^P \in \mathbb{R}^{L \times d}$ for positional information in the self-attention mechanism. In this layer, each output element, $z_i \in \mathbb{R}^d$, is computed by

$$z_i = \sum_{j=1}^t \alpha_{ij} (x_j + e_{r_{ij}} + e_{p_j}) \quad (14)$$

where $e_{p_j} \in \mathbb{R}^d$ is the positional embedding for the position j in the session. α_{ij} is the attention weight, which is calculated as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^t \exp(e_{ik})} \quad (15)$$

Furthermore, e_{ij} is the intermediate value, which is used to measure the correlation between the pair of micro-behaviors. It is computed as:

$$e_{ij} = \frac{x_i W^Q (x_j + e_{r_{ij}} + e_{p_j})}{\sqrt{d}} \quad (16)$$

where $W^Q \in \mathbb{R}^{d \times d}$ is the input projection for the query, which is used to make the representation more flexible. Compared with the standard self-attention mechanism, we have different query, key, and value vectors here. Essentially, the output has incorporated the information of dyadic operations, which can better reflect the intents and preferences of the user based on micro-behavior patterns.

Then we apply *Position-wise Feed-Forward Network* to endow the model with more non-linearity,

$$FFN(z_i) = \max(0, z_i W_1 + b_1) W_2 + b_2 \quad (17)$$

where $W_1, W_2 \in \mathbb{R}^{d \times d}$ are the weight matrices and $b_1, b_2 \in \mathbb{R}^d$ are the bias vectors. After that, we add residual connections, layer normalization, and dropout mechanism as in the basic self-attention model. Finally, we get the learned vector z_s as the representation of the user's final global preference, which corresponds to x_s in the input.

D. Session Representation and Prediction

To obtain a session representation, we take into account a user's global preference and recent interest. For the recent interest, we directly take the representation of the last micro-behavior to denote it, i.e. x_t in the equation 12. Then, we concatenate these representation and apply a fusion gating network for the final representation of the proposed model,

$$\begin{aligned} \beta &= \sigma(W_m[z_s; x_t] + b_m) \\ m &= \beta \odot z_s + (1 - \beta) \odot x_t \end{aligned} \quad (18)$$

where $m \in \mathbb{R}^d$ denotes the final output of the proposed model, $W_m \in \mathbb{R}^{2d \times d}$ is the weighting matrix, $b_m \in \mathbb{R}^d$ is the bias

vector. Lastly, for each item $v_i \in V$, we produce the prediction score as follows:

$$\hat{m} = w_k L_2 Norm(m), \hat{v}_i = L_2 Norm(v_i) \quad (19)$$

$$\hat{y}_i = softmax(\hat{m}^T \hat{v}_i)$$

where v_i is the initial embedding of the item i and \hat{y}_i denotes the probability of the item in the candidate item set I . $L_2 Norm$ is the L_2 Normalization function and w_k is the normalized weight. This weighted normalization [45] and the Regularizing Softmax loss [46] make the training process more stable and insensitive to hyper-parameters.

For training the model, we use cross-entropy as the optimization objective to learn the parameters:

$$L = - \sum_i y_i \log(\hat{y}_i) \quad (20)$$

V. EXPERIMENTS

A. Settings

1) *Datasets*: We conduct experiments on three publicly available real-world datasets. The first two datasets ² are from a large Chinese e-commerce site JD.com [11], and the other is from the RecSys Challenge 2019 ³:

- **JD Datasets**: The datasets contain the user interaction sessions of online shopping in two product categories, “Appliances” and “Computers”, respectively. There are **10** different types of micro-behavior operations in each of them, such as “SearchList2Product”, “Detail_comments” and “Order”.
- **Trivago Dataset**: The dataset is provided by trivago ⁴, which is a global hotel search platform focused on re-shaping the way travelers search for and compare hotels. It contains user actions about the hotel and specifies the type of action that has been performed. We only use the train set of this challenge and take **6** types of micro-operations such as “interaction item image” and remove the operation whose reference value is not the item, such as “filter selection” and “search for destination”.

For a fair comparison, we follow the previous work [11] to filter out the items with fewer than 50 occurrences in *JD datasets*; use 70%, 10%, and 20% of sessions as the training, validation, and testing set, respectively; and use the last item in each session as the ground truth for our SR predictions. In addition, we exclude the sessions consisting of only a single item from training and testing [7]. For *trivago dataset*, the only difference is that we filter out the items with fewer than 5 occurrences. The statistics of these three datasets after preprocessing are shown in Tab. II.

2) *Baselines*: To evaluate the effectiveness of our proposed model, we compare it with the following state-of-the-art methods for SR. Those baselines are classified into two categories: macro-behavior models that only utilize the sequences of items

TABLE II
STATISTICS OF THE DATASETS USED.

Datasets	JD-Appliances	JD-Computers	Trivago
# train	583,255	577,301	26,0877
# validation	83,279	82,391	37,027
# test	166,670	164,782	74,770
# items	75,159	93,140	183,561
# micro-behavior	32,736,184	24,245,132	5,726,369

for SR, and micro-behavior models that also take the detailed operations on each item into consideration.

Macro-Behavior Models:

- **S-POP** [21, 47] simply recommends the most popular items in the current session. It is an improved version of the popularity-based method for SR.
- **SKNN** [19] is the session-based k -nearest-neighbors approach, which scores an item based on the similarity between the target session and historical sessions.
- **NARM** [5] applies RNN and the attention mechanism to capture the user’s main purpose.
- **STAMP** [6] is a short-term memory priority model, which combines the user’s general interest and current interest reflected by the last-click to generate recommendation results
- **SRGNN** [7] is a session-based recommendation model that utilizes a graph neural network to learn the item and session representation
- **GCSAN** [8] models sessions as directed graphs and makes recommendations with a self-attention network.
- **BERT4Rec** [23] uses deep bidirectional self-attention to represent user macro-behaviors.
- **SGNN-HN** [9] considers long-distance relations between items in a session by star graph neural network (SGNN) and applies highway networks to deal with the overfitting problem for SR.

Micro-Behavior Models:

- **RIB** [10] is the first paper that incorporates micro-behaviors into SR by simply using a GRU.
- **HUP** [11] proposes a hierarchical RNN framework to harvest the sequential information of users’ micro-behaviors.
- **MKM-SR** [12] is the latest SR model that employs GNN for item embedding and GRU for operation embedding; the variant which does not include the auxiliary task of knowledge embedding is used in our experiments since we don’t have the knowledge graph of items.

3) *Metrics*: Following previous works [9, 11, 12], we adopt two commonly used performance measures for SR — *Hit Rate* (H) and *Mean Reciprocal Rank* (M) at top K — to evaluate our proposed model EMBSR and the competing methods.

- **H@K**: It is the proportion of cases that the ground truth is ranked amongst the top-K items.

$$H@k = \frac{n_{hit}}{N} \quad (21)$$

²The full datasets are available at <https://tinyurl.com/ybo8z4yz>.

³<http://www.recsyschallenge.com/2019/>

⁴<https://www.trivago.com/>

TABLE III
PERFORMANCES (%) OF ALL THE SR METHODS. THE HIGHEST SCORES ARE BOLDFACED; THE 2ND HIGHEST SCORES ARE UNDERLINED.

Datasets	Metrics	S-POP	SKNN	NARM	STAMP	SR-GNN	GC-SAN	BERT4Rec	SGNN-HN	RIB	HUP	MKM-SR	EMBSR	Imp.
Appliances	H@5	31.66	25.06	30.94	30.74	32.65	30.36	31.02	<u>34.80</u>	30.12	31.91	33.82	37.34	7.30%
	H@10	42.45	36.96	42.69	42.10	43.80	42.02	42.67	<u>47.07</u>	40.84	43.39	45.02	49.57	5.31%
	H@20	49.56	49.30	54.74	53.98	55.32	54.02	54.30	<u>59.36</u>	51.61	54.73	56.57	61.64	3.84%
	M@5	17.29	13.15	17.90	18.21	19.63	17.83	16.96	<u>21.00</u>	16.97	17.83	20.73	23.58	12.29%
	M@10	18.74	14.73	19.46	19.72	21.11	19.38	18.52	<u>22.64</u>	18.40	19.37	22.22	25.21	11.35%
	M@20	19.25	15.59	20.30	20.55	21.91	20.22	19.33	<u>23.49</u>	19.15	20.16	23.03	26.06	10.94%
Computers	H@5	17.18	15.11	18.31	18.18	20.08	18.79	17.90	<u>21.53</u>	16.93	18.87	21.00	24.17	12.26%
	H@10	24.82	23.56	28.14	26.97	29.11	28.75	26.79	<u>32.01</u>	24.56	27.62	30.21	34.75	8.56%
	H@20	30.22	33.55	39.34	37.44	39.72	39.98	36.98	<u>43.67</u>	33.58	37.49	40.86	46.29	6.00%
	M@5	9.45	7.89	9.40	10.09	11.38	9.26	9.42	11.61	9.26	10.15	<u>12.01</u>	13.98	16.40%
	M@10	10.47	9.01	10.70	11.26	12.57	10.58	10.59	13.00	10.27	1.31	<u>13.23</u>	15.38	16.25%
	M@20	10.86	9.70	11.48	11.98	13.31	11.35	11.30	13.81	10.89	11.99	<u>13.97</u>	16.18	15.82%
Trivago	H@5	0	7.89	12.89	13.11	11.97	14.15	11.01	<u>14.58</u>	9.00	10.06	12.34	15.80	8.37%
	H@10	0	14.03	18.02	17.13	15.91	20.10	15.00	<u>20.13</u>	11.65	14.05	16.63	22.95	14.01%
	H@20	0	20.69	23.84	21.49	20.13	26.21	19.43	<u>26.39</u>	14.39	18.65	21.45	31.18	18.15%
	M@5	0	2.65	7.57	8.09	7.42	7.76	6.60	<u>8.79</u>	5.69	6.00	7.58	9.05	2.96%
	M@10	0	3.47	8.25	8.62	7.94	8.55	7.13	<u>9.53</u>	6.04	6.53	8.14	10.00	4.93%
	M@20	0	3.93	8.65	8.92	8.24	8.97	7.43	<u>9.96</u>	6.24	6.85	8.48	10.56	6.02%

where N is the number of test sessions and n_{hit} is the number of cases that the ground truth is included in the top K list.

- **M@K**: It is the average of reciprocal ranks of the desired items, which is the evaluation of ranked results. When the rank is larger than k , the reciprocal rank is set to zero.

$$M@K = \frac{1}{N} \sum_{v' \in S_{test}} \frac{1}{Rank(v')} \quad (22)$$

where v' is the ground truth and S_{test} is the recommended list of the test dataset.

4) *Hyperparameters*: The hyperparameters for all methods in comparison are tuned on the validation set via grid search. For all methods, we use Adam as the model optimizer, tuned the learning rate in $[0.001, 0.003, 0.005, 0.008, 0.01]$ and the dropout rate in $[0, 0.1, 0.2, 0.3, 0.4, 0.5]$. Furthermore, in our PyTorch implementation of the neural network models, the parameters are initialized the same with [12], the embedding size is $d = 100$, the mini-batch size is 512, and the largest number of epochs is 50 for fair comparison. Following [9], the normalized weight w_k is set to 12 on three datasets.

B. Overall Performances

We first compare the top- K recommendation performance with other state-of-the-art methods and set $K = [5, 10, 20]$ to evaluate the performance. Tab. III shows our experimental results which lead to the following findings.

First, our proposed EMBSR method consistently outperforms all the baselines on three datasets, which clearly demonstrates the effectiveness of utilizing not only the sequential patterns but also the explicit (dyadic) relational patterns of user micro-behaviors. Compared with those macro-behavior models, EMBSR goes further to take into account the user’s micro-behavior operations which have finer granularity and provide a deeper understanding of the user. Compared with the other micro-behavior models, EMBSR works as it exploits the pairwise semantics of micro-behaviors by encoding and examining their dyadic relations directly rather than just relying on

the sequential patterns. According to the *Wilcoxon signed-rank test*, the performance improvements brought by our proposed EMBSR over the best performing baselines are statistically significant with the p -values $\ll 0.01$ on all datasets.

Second, Deep-learning approaches in general achieve much better performances than traditional methods, such as SKNN. Despite the recent improvements made to such non-neural models, they are still limited in detecting and exploiting useful patterns in sessions. Moreover, we can observe that S-POP method completely fails on Trivago, which means that the ground truth item rarely appears in the session. We believe that it is why EMBSR has different improvements on two metrics of the three datasets. For Trivago, since the ground truth is not included in the session, the proposed model needs more ability to find it, leading to a huge improvement in $H@K$. For Appliances and Computers, the information of the ground truth is easier to find; our model is able to give it a higher score to improve the ranking performance.

Third, GNN-based models (SRGNN, GCSAN, SGNN-HN, and MKM-SR) in general outperform the RNN-based models (NARM, RIB, HUP) and the attention-based models (STAMP, BERT4Rec), testifying the superior capacity of GNN in modeling session information. SGNN-HN has the second best performance in most cases, which verifies the effectiveness of SGNN on modeling the complex transition relationship of macro-items. In addition, MKM-SR can outperform SRGNN in most cases, showing that incorporating the micro-behavior by RNN can also improve the performance as MKM-SR considers micro-operation based on the same GNN structure. Nevertheless, our proposed EMBSR outperforms all those GNN baselines with a large margin, suggesting that two patterns of micro-behavior encoded by EMBSR probably have captured the most useful information.

C. Ablation Studies

In order to measure the contribution from the different components of the proposed model, we contrast it with three

TABLE IV
PERFORMANCES (%) OF ABLATION STUDIES.

Method	JD-Appliances				JD-Computers				Trivago			
	H@10	H@10	M@10	M@20	H@10	H@20	M@10	M@20	H@10	H@20	M@10	M@20
EMBSR-NS	46.73	59.32	21.12	22.00	32.19	43.85	12.51	13.32	23.04	31.74	9.76	10.36
EMBSR-NG	46.78	59.43	19.04	19.92	32.04	44.02	11.91	12.74	22.19	30.52	9.96	10.53
EMBSR-NF	48.99	60.79	25.60	26.42	15.17	33.35	44.66	15.96	9.90	20.56	26.80	10.33
EMBSR	49.57	61.64	25.21	26.07	34.75	46.29	15.38	16.18	22.95	31.18	10.00	10.56

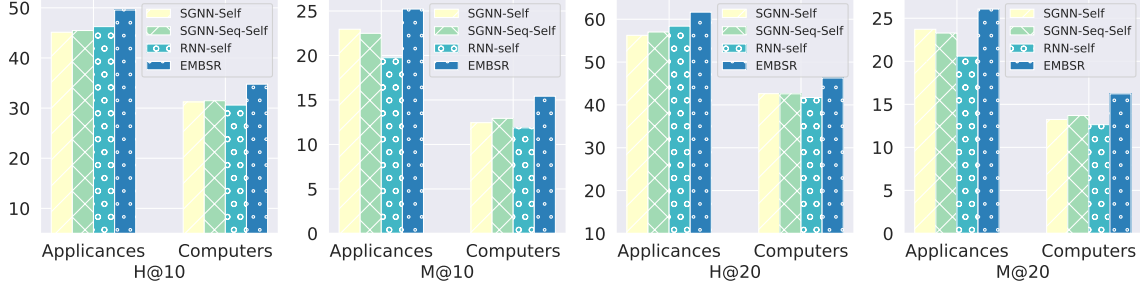


Fig. 4. Performance (%) comparison to assess the utility of the sequential pattern of micro-behaviors.

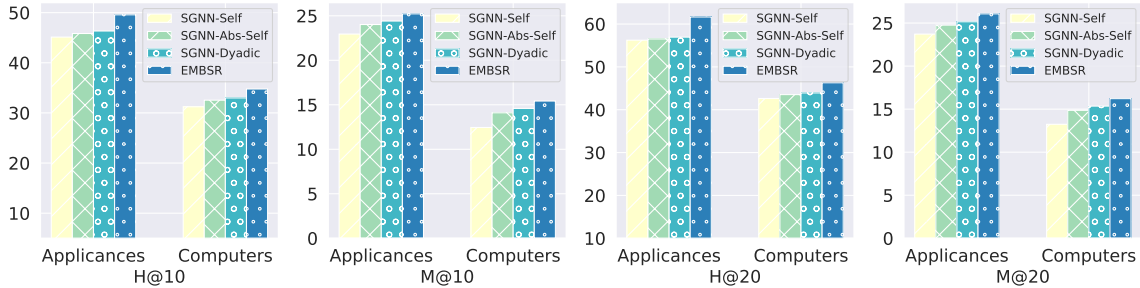


Fig. 5. Performance (%) comparison to assess the utility of the dyadic relation pattern of micro-behaviors.

ablated versions and set $K = [10, 20]$ to evaluate the performance:

- **EMBSR-NS** removes the operation-aware self-attention layer and only encodes the sequential pattern of micro-behaviors for the task.
- **EMBSR-NG** removes the entire GNN Layer, including the GRU layer for micro-operation sequence, and only encoding the dyadic relational pattern of micro-behaviors for the task.
- **EMBSR-NF** removes the fusion gate network for the final representation, while we directly concatenate these two embeddings and feed it into an MLP for the representation of the session

Tab. IV presents the results of the ablation studies for three datasets. In Appliances and Computers, EMBSR-NS and EMBSR-NG yield the worst performance, confirming that only modeling a single pattern of the session cannot fully capture the user’s real intent and preference; In comparison, EMBSR-NS is in general slightly better than EMBSR-NG, especially on $M@K$, proving that the GNN-based model incorporating

the sequential pattern of micro-behaviors has indeed helped to make better predictions; EMBSR-NF in general has the second best performance, which has further demonstrated the effectiveness of explicitly modeling two different patterns of micro-behaviors since it only uses another way to generate the representation of the session by these two patterns.

In Trivago, the results are slightly more complicated. EMBSR-NF generally yields the worst performance, while EMBSR-NS and EMBSR-NR both deliver the competitive performance, indicating that the correct fusion mechanism is also essential in the dataset. Therefore, the fact that the full EMBSR model in general performs best confirms the advantages of fusing different representations by the fusion gating network.

D. Utility of Sequential Patterns

To understand the impact of the sequential pattern, we choose the Appliances and Computers datasets for further investigation since they have more types of micro-operations. Since the motivation of this paper is to utilize the micro-behaviors to help improve the recommendation quality on

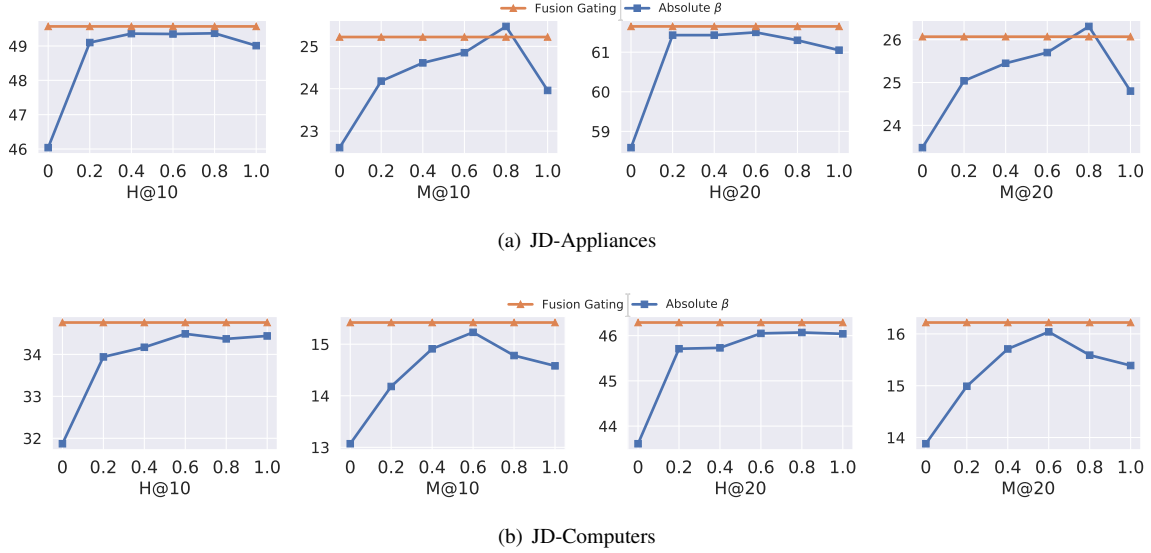


Fig. 6. Performance (%) comparison to assess the utility of the fusion gating mechanism.

items, we also design three variants to investigate the way of incorporating the different patterns of micro-behaviors:

- **SGNN-Self** uses the star graph without GRU and a standard self-attention mechanism. It has no information of micro-behaviors and can only learn the representation of the session by macro-items.
- **SGNN-Seq-Self** encodes the information of sequential pattern on the SGNN with GRU based on SGNN-Self.
- **RNN-Self** replaces the whole GNN layer with an RNN layer compared with SGNN-Self. It directly concatenates the item embedding and operation embedding of the initial session and feeds them to the RNN for learning the sequential pattern of micro-behaviors.

Fig. 4 illustrates the result of the comparison. We see that EMBSR achieves the best performance in all cases. As for the variants, SGNN-Seq-Self in general performs better than SGNN-Self. We attribute this to the sequential pattern of micro-behaviors, which are related to users’ preference of the macro-items. However, the RNN-based method, RNN-Self, performs worst in most cases, especially on $M@10$ and $M@20$. Although it also includes the micro-operations information, only a simple encoding method based on RNN cannot capture the complex transition pattern of macro-items and make full use of micro-behavior information. Thus, EMBSR presents an approach to encode the sequential pattern of micro-behaviors in GNN, including a novel method to convert a session into a multigraph, which is able to model the user’s real intention and preference.

E. Utility of Dyadic Relational Patterns

To demonstrate the impact of dyadic encoding, we compare the experimental results with two other variants:

- **SGNN-Abs-Self** replaces operation-aware self-attention and dyadic encoding with standard self-attention and absolute operation embedding.

- **SGNN-Dyadic** encodes dyadic relational patterns but only uses SGNN for macro-items without RNN.

We also add SGNN-Self and EMBSR for a clear comparison. As illustrated in Fig. 5, we observe that RNN-Self has achieved the worst performance again since it does not have any micro-behavior information. Moreover, SGNN-Dyadic outperforms SGNN-Abs-Self in all cases. Here, SGNN-Abs-Self encodes the micro-behaviors by a simple way that adds the absolute operation embedding in a standard self-attention network. It cannot capture pair-wise semantic of the micro-operations well, which is important to understand the difference between two behaviors. Furthermore, there are some surprising improvements about the performance of SGNN-Dyadic, which is lack of sequential patterns of micro-behaviors, but still achieves competitive results, especially on $M@K$ of Computers. This indicates that the dyadic relational pattern, which has never been exploited or discussed in SR, is essential to model the user’s real preference on the item.

F. Utility of Fusion Gating Mechanism

In order to investigate the utility of the fusion gating mechanism in equation 18, we tune β in $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and use it as an absolute weight to control the fusion of different information. Specifically, x_t represents the recent interest of the user, but z_s has both sequential and dyadic encoding information. β controls what information is used to generate the final representation.

As illustrated in Fig. 6, it is reasonable to infer that how to select the information is crucial to the final performance. When $\beta = 0$, we only use the recent interest x_t , and it achieves the worst performance, evidencing that only the recent interest cannot fully express the user’s preference. When $\beta = 1$, the result is competitive since z_s has considered all information of micro-behaviors within the session. However, due to too much information, it may not be able to capture the user’s

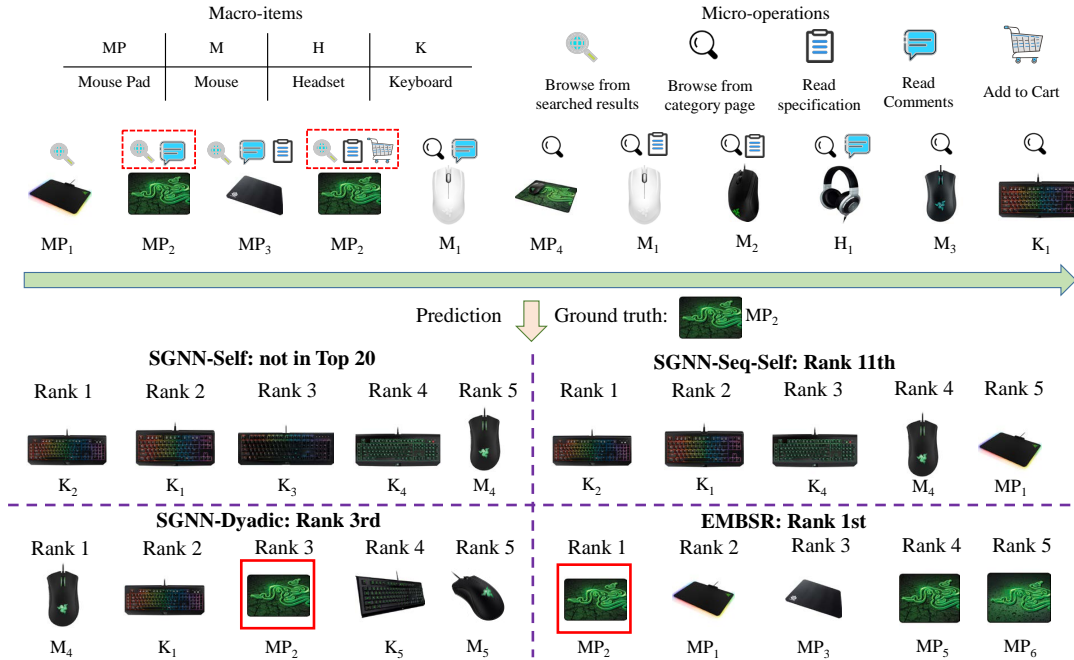


Fig. 7. A case for the session-based recommendation with the micro-behavior.

recent interest. Thus, a simple way for the information fusion with an absolute β can achieve better performance. Moreover, the final result is not always sensitive to β , since the learning process can automatically adapt to achieve a better result. The proposed fusion gating network avoids the choice of β , and has the best results in general.

G. Case Study

Fig. 7 illustrates a real case from “Computers” dataset to demonstrate the rationality of our proposed framework. This session has a total of 20 micro-behaviors on 11 macro items. The bottom part is the prediction results of three variants (SGNN-Self, SGNN-Seq-Self, SGNN-Dyadic) and EMBSR for this session. We also list the top five items that each method recall and can observe that:

(1) The macro-item sequence only reflect the coarse-grained preference of the user. From the item sequence of the case, we can infer that the user tends to pick out one computer accessory. In general, the recent items have more impacts on the next click [6]. Therefore, the top five items recommended by the SGNN-Self are all keyboards since the last item is a keyboard. However, the keyboard is far from the user’s real intention, resulting in the failed recall of SGNN-Self.

(2) Micro-behaviors play a significant role in understanding users’ fine-grained preferences for items. In this case, the user has obvious signals for the mouse pad. For MP₂ and MP₃, the user reads the detail specification and comments after click, and finally adds MP₂ to the shopping cart. Therefore, SGNN-Seq-Self, SGNN-Dyadic, and EMBSR successfully recall the MP₂ in the top 20 since they all have considered the micro-operation signals. Moreover, EMBSR integrates sequential patterns with dyadic relational patterns, which enables the

model to capture the real intention of the user. As we can see, the top five items recalled by EMBSR are all mouse pads. Noting that MP₂, MP₅, and MP₆ are the same items with different sizes (medium, small and large, respectively), which proves that EMBSR understands the user’s preferences and accurately captures the similarities among items.

VI. CONCLUSIONS

In this paper, we propose a novel approach to SR — EMBSR — which considers not only the sequential patterns but also the dyadic relational patterns of micro-behaviors within each session. Specifically, we have designed a graph neural network to aggregate the sequence of micro-behaviors, and developed an operation-aware self-attention mechanism to extract the pair-wise semantics of micro-behaviors. Our experiments have demonstrated that the proposed EMBSR model significantly outperforms all state-of-the-art SR methods. For future work, it would be interesting to investigate how to exploit other operations that are for all items such as filtering and sorting to further improve the performance of SR systems, and whether it would be beneficial to weight, or filter, micro-behavior operations according to their importance.

ACKNOWLEDGMENT

This work was supported by NSFC grants (No. 62136002 and 61972155), National Key R&D Program of China (No. 2021YFC3340702), the Science and Technology Commission of Shanghai Municipality (20DZ1100300) and the Open Project Fund from Shenzhen Institute of Artificial Intelligence and Robotics for Society, under Grant No. AC01202005020, Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

REFERENCES

- [1] H. Garcia-Molina, G. Koutrika, and A. Parameswaran, "Information seeking: Convergence of search, recommendations, and advertising," *Communications of the ACM*, vol. 54, no. 11, pp. 121–130, Nov 2011.
- [2] D. Jannach, L. Lerche, and M. Jugovac, "Adaptation and evaluation of recommendations for short-term shopping goals," in *RecSys*, 2015, pp. 211–218.
- [3] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *CSUR*, vol. 51, no. 4, pp. 1–36, 2018.
- [4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.
- [5] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *CIKM*. ACM, 2017, pp. 1419–1428.
- [6] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: Short-term attention/memory priority model for session-based recommendation," in *KDD*. ACM, 2018, pp. 1831–1839.
- [7] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *AAAI*, vol. 33, 2019, pp. 346–353.
- [8] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *IJCAI*, 2019, pp. 3940–3946.
- [9] Z. Pan, F. Cai, W. Chen, H. Chen, and M. de Rijke, "Star graph neural networks for session-based recommendation," in *CIKM*, 2020, pp. 1195–1204.
- [10] M. Zhou, Z. Ding, J. Tang, and D. Yin, "Micro behaviors: A new perspective in e-commerce recommender systems," in *WSDM*, 2018, pp. 727–735.
- [11] Y. Gu, Z. Ding, S. Wang, and D. Yin, "Hierarchical user profiling for e-commerce recommender systems," in *WSDM*, 2020, pp. 223–231.
- [12] W. Meng, D. Yang, and Y. Xiao, "Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation," in *SIGIR*, 2020, pp. 1091–1100.
- [13] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1724–1734.
- [14] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *NAACL-HLT, Volume 2 (Short Papers)*. New Orleans, Louisiana: ACL, Jun. 2018, pp. 464–468.
- [15] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NeurIPS*, 2008, pp. 1257–1264.
- [16] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*. Springer, 2015, pp. 77–118.
- [17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.
- [18] I. Kamehkhosh, D. Jannach, and M. Ludewig, "A comparison of frequent pattern techniques and a deep learning method for session-based recommendation," in *RecTemp@ RecSys*, 2017, pp. 50–56.
- [19] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *RecSys*, 2017, pp. 306–310.
- [20] D. Garg, P. Gupta, P. Malhotra, L. Vig, and G. Shroff, "Sequence and time aware neighborhood for session-based recommendations: Stan," in *SIGIR*, 2019, pp. 1069–1072.
- [21] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.
- [22] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *DLRS*. ACM, 2016, pp. 17–22.
- [23] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *CIKM*, 2019, p. 1441–1450.
- [24] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, and M. de Rijke, "A collaborative session-based recommendation approach with parallel memory modules," in *SIGIR*, ser. SIGIR'19, 2019, p. 345–354.
- [25] A. Luo, P. Zhao, Y. Liu, F. Zhuang, D. Wang, J. Xu, J. Fang, and V. S. Sheng, "Collaborative self-attention network for session-based recommendation," in *IJCAI*, 2020, pp. 2591–2597.
- [26] R. Qiu, J. Li, Z. Huang, and H. Yin, "Rethinking the item order in session-based recommendation with graph neural networks," in *CIKM*, 2019, pp. 579–588.
- [27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
- [28] T. Chen and R. C.-W. Wong, "Handling information loss of graph neural networks for session-based recommendation," in *KDD*, 2020, pp. 1172–1180.
- [29] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, and M. Qiu, "Global context enhanced graph neural networks for session-based recommendation," in *SIGIR*, 2020, pp. 169–178.
- [30] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *AAAI*, vol. 35, no. 5, 2021, pp. 4503–4511.
- [31] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T.-S. Chua, L. Yao, Y. Song, and D. Jin, "Learning to recommend with multiple cascading behaviors," *IEEE TKDE*, 2019.
- [32] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *KDD*, 2008, pp. 650–

- [33] Q. Liu, S. Wu, and L. Wang, "Multi-behavioral sequential prediction with recurrent log-bilinear model," *IEEE TKDE*, vol. 29, no. 6, pp. 1254–1267, 2017.
- [34] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *ICML*, 2007, pp. 641–648.
- [35] M. Wan and J. McAuley, "Item recommendation on monotonic behavior chains," in *RecSys*, 2018, pp. 86–94.
- [36] C. Lo, D. Frankowski, and J. Leskovec, "Understanding behaviors that lead to purchasing: A case study of pinterest," in *KDD*, 2016, pp. 531–540.
- [37] R. Olbrich and C. Holsing, "Modeling consumer purchasing behavior in social shopping communities with clickstream data," *International Journal of Electronic Commerce*, vol. 16, no. 2, pp. 15–40, 2011.
- [38] S. Yehezki and A. Dhini, "Classifying purchase decision based on user clickstream in e-commerce using web usage mining," in *Proc. Int. Conf. Bus. Inf. Manage.*, 2017, pp. 57–61.
- [39] B. Loni, R. Pagano, M. Larson, and A. Hanjalic, "Bayesian personalized ranking with multi-channel user feedback," in *RecSys*, 2016, pp. 361–364.
- [40] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, and X. Wang, "Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems," *Knowledge-Based Systems*, vol. 138, pp. 202–207, 2017.
- [41] H. Qiu, Y. Liu, G. Guo, Z. Sun, J. Zhang, and H. T. Nguyen, "Bprh: Bayesian personalized ranking for heterogeneous implicit feedback," *Information Sciences*, vol. 453, pp. 80–98, 2018.
- [42] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in *ICLR*, 2016.
- [43] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
- [44] J. Yuan, Z. Song, M. Sun, X. Wang, and W. X. Zhao, "Dual sparse attention network for session-based recommendation," in *AAAI*, vol. 35, no. 5, 2021, pp. 4635–4643.
- [45] P. Gupta, D. Garg, P. Malhotra, L. Vig, and G. Shroff, "Niser: Normalized item and session representations to handle popularity bias," *arXiv*, pp. arXiv–1909, 2019.
- [46] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," in *CVPR*, 2018, pp. 5089–5097.
- [47] Z. Wang, C. Chen, K. Zhang, Y. Lei, and W. Li, "Variational recurrent model for session-based recommendation," in *CIKM*, 2018, pp. 1839–1842.

Supplemental Materials

I. ADDITIONAL EXPERIMENTS

A. Optimal performance for macro-behavior baselines

Since most of existing methods of session-based recommendation are designed for the sequence of macro item, the performance of macro-behavior baselines utilizing micro-behavior may not be able to reach their optimal performance if there are a large number of operations which are treated equally. Therefore, we have identified one type of operation and used that to redefine the sequence of items for the additional experiment. Since most methods for SR use the clickstream data, we only keep the click-related data in two JD datasets and "click-outs" data in Trivago for macro-behavior baselines. Meanwhile, we add other operations to the sequence while ensuring that the ground truth of each sequence is consistent for a fair comparison with our proposed approach.

The result is reported in Tab. I, we compare EMBSR with the best baseline SGNN-HN and the typical baseline BERT4Rec. This table shows similar patterns with using all operations for macro-behavior models. In addition, we have achieved a considerable improvement on Trivago. Intuitively, EMBSR that are able to exploit the information embedded in all the operations does work better than those that are restricted to utilize only some of the operations. Macro-behavior models do have many limitations.

Furthermore, assigning different importance weights to different operations is probably more a promising way than simply discarding some insignificant operations. It is indeed an interesting question whether it would be beneficial to weight, or filter, micro-behavior operations according to their importance. Besides, the importance of micro-behavior operations may be not static but vary in different sequences or at different positions. This line of thoughts, though interesting, would make the model much more complicated than its current form. We have to leave the bulk of experiments for the future work.

B. Dyadic relational encoding with SGNN-HN

In Section V-E (Figure 5), we do witness the high utility of dyadic relational patterns for recommendation effectiveness. In fact, **SGNN-Dyadic** is the model that we isolate the idea of dyadic encoding to the best macro-behavior baseline SGNN-HN, which has achieved the competitive results. We have reported their performances on two JD datasets in Tab. II here. From this table, we observe that EMBSR-Dyadic outperforms SGNN-HN, except for H@10 and H@20 on Appliances, further confirming the importance of the dyadic relationship and the generalizability of the proposed idea. In addition, EMBSR still achieves a large improvement, showing that the proposed encoding scheme that transforms a session into a directed multigraph and the novel aggregation stage are more suitable for micro-behavior encoding.

C. Top Ranked Results

In order to more comprehensively evaluate the performance of our proposed EMBSR, we have reported the result of

TABLE I
PERFORMANCES (%) OF DEFINING THE SEQUENCE OF ITEMS WITH THE SINGLE TYPE OF OPERATION.

Datasets	Metrics	BERT4Rec	SGNN-HN	EBMSR	Imp
Appliances	H@5	30.16	<u>34.95</u>	37.41	7.04%
	H@10	41.50	<u>47.17</u>	49.76	5.49%
	H@20	52.97	<u>59.56</u>	62.00	4.97%
	M@5	16.54	<u>21.34</u>	23.74	11.25%
	M@10	18.05	<u>22.96</u>	25.40	10.63%
	M@20	18.85	<u>23.82</u>	26.25	10.20%
Computers	H@5	17.54	<u>21.34</u>	24.23	13.54%
	H@10	26.67	<u>31.93</u>	34.91	9.33%
	H@20	37.11	<u>43.72</u>	46.50	6.36%
	M@5	8.81	<u>11.49</u>	13.83	20.37%
	M@10	10.02	<u>12.89</u>	15.24	18.23%
	M@20	10.74	<u>13.71</u>	16.04	16.99%
Trivago	H@5	13.98	<u>17.21</u>	25.03	45.44%
	H@10	17.51	<u>22.69</u>	29.89	31.73%
	H@20	20.48	<u>27.94</u>	34.92	24.98%
	M@5	8.57	<u>10.72</u>	18.97	76.96%
	M@10	9.04	<u>11.44</u>	19.61	71.42%
	M@20	9.25	<u>11.81</u>	19.96	69.01%

TABLE II
PERFORMANCES (%) OF APPLYING THE DYADIC ENCODING TO SGNN-HN.

Datasets	Metrics	SGNN-HN	EMBSR-Dyadic	EBMSR
Appliances	H@5	34.80	<u>35.64</u>	37.34
	H@10	47.04	46.36	49.57
	H@20	<u>59.36</u>	56.94	61.64
	M@5	21.00	<u>22.98</u>	23.58
	M@10	22.64	<u>24.41</u>	25.21
	M@20	23.49	<u>25.15</u>	26.06
Computers	H@5	21.53	<u>23.09</u>	24.17
	H@10	32.01	<u>32.99</u>	34.75
	H@20	43.67	<u>43.92</u>	46.29
	M@5	11.61	<u>13.28</u>	13.98
	M@10	13.00	<u>14.59</u>	15.38
	M@20	13.81	<u>15.35</u>	16.18

top 1,3,5 in Tab. III. From this table, we can observe that the performances of top 1, 3, and 5 results exhibit similar patterns with top 10, and 20. It is worth noting that there is no difference between $H@I$ and $M@I$, so they have the same values. In addition, as we analyzed in the paper, since the ground truth is not included in the input session, our proposed approach has not achieved the best performance on Trivago when $K = 1$.

TABLE III
PERFORMANCES (%) OF $K = [1, 3, 5]$. THE HIGHEST SCORES ARE BOLDFACED; THE 2ND HIGHEST SCORES ARE UNDERLINED.

Datasets	Metrics	S-POP	SKNN	NARM	STAMP	SR-GNN	GC-SAN	BERT4Rec	SGNN-HN	RIB	HUP	MKM-SR	EMBSR	Imp.
Appliances	H@1	9.38	6.97	10.84	11.39	12.46	11.05	9.15	13.48	9.65	10.02	<u>13.49</u>	16.06	19.05%
	H@3	23.45	17.49	23.20	23.41	25.13	22.91	23.07	<u>26.67</u>	22.73	23.93	26.32	29.24	9.64%
	H@5	31.66	25.06	30.94	30.74	32.65	30.36	31.02	<u>34.80</u>	30.12	31.91	33.82	37.34	7.30%
	M@1	9.38	6.97	10.84	11.39	12.46	11.05	9.15	13.48	9.65	10.02	<u>13.49</u>	16.06	19.05%
	M@3	15.42	11.42	16.14	16.54	17.92	16.14	15.15	<u>19.15</u>	15.29	16.02	19.03	21.74	13.52%
	M@5	17.29	13.15	17.90	18.21	19.63	17.83	16.96	<u>21.00</u>	16.97	17.83	20.73	23.58	12.29%
Computers	H@1	5.27	4.21	4.79	5.86	6.80	4.32	4.95	6.46	5.17	5.52	<u>7.24</u>	8.64	19.34%
	H@3	12.60	10.35	12.62	13.07	14.66	12.72	12.58	15.22	12.24	13.54	<u>15.43</u>	17.78	15.23%
	H@5	17.18	15.11	18.31	18.18	20.08	18.79	17.90	<u>21.53</u>	16.93	18.87	21.00	24.17	12.26%
	M@1	5.27	4.21	4.79	5.86	6.80	4.32	4.95	6.46	5.17	5.52	<u>7.24</u>	8.64	19.34%
	M@3	8.41	6.82	8.11	8.94	10.15	7.88	8.21	10.18	8.19	8.94	<u>10.75</u>	12.54	16.65%
	M@5	9.45	7.89	9.40	10.09	11.38	9.26	9.42	11.61	9.26	10.15	<u>12.01</u>	13.98	16.40%
Trivago	H@1	0	0.05	4.68	5.31	4.87	4.39	4.20	5.64	3.80	3.81	4.95	<u>5.49</u>	-2.66%
	H@3	0	4.34	9.73	10.19	9.43	10.19	8.37	<u>11.14</u>	7.20	7.61	9.57	11.62	4.31%
	H@5	0	7.89	12.89	13.11	11.97	14.15	11.01	<u>14.58</u>	9.00	10.06	12.34	15.80	8.37%
	M@1	0	0.05	4.68	5.31	4.87	4.39	4.20	5.64	3.80	3.81	4.95	<u>5.49</u>	-2.66%
	M@3	0	1.85	6.85	7.42	6.84	6.85	6.00	<u>8.01</u>	5.28	5.44	6.94	8.10	1.12%
	M@5	0	2.65	7.57	8.09	7.42	7.76	6.60	<u>8.79</u>	5.69	6.00	7.58	9.05	2.96%