Meta-path Based Neighbors for Behavioral Target Generalization in Sequential Recommendation

Junyang Chen, *Member, IEEE*, Zhiguo Gong, *Senior Member, IEEE*, Yuanman Li, Huanjian Zhang, Hongyong Yu, Junzhang Zhu, Ge Fan, Xiaoming Wu, Kaishun Wu, *Senior Member, IEEE*

Abstract—Click-through rate (CTR) prediction is a crucial task in recommender systems, which aims to model users' dynamic preferences from their historical behaviors. To achieve this goal, most of the previous models adopt sequential neural networks (e.g., GRU) to encode the historical interactions into item representations for recommendations. Though these methods can perform well on recommending highly relevant items to users, we argue that such models are sub-optimal for the long-term user experience due to highly skewed recommendations: Monotonous items with similar subjects get more exposure because of inadequate interest explorations. Thus, some items which are not quite relevant to the users' historical preferences should be considered. To address these limitations, we propose a Heterogeneous Graph Enhanced Sequential Neural Network, HGESNN, to explore the interests of users beyond their historical interactions by explicitly modeling item relations with meta-path constructions. We incorporate a transformer-based network to embed personalized user intents into sequential learning. In the experiments on both public and industrial datasets, HGESNN significantly outperforms the state-of-the-art solutions. Specifically, HGESNN has been deployed in the main traffic of our Image-Text feed recommender system, which obtains 6.28%, 6.82%, and 4.77% CTR gains on news, novels, and entertainment contents, respectively.

Index Terms—Sequential Recommendation, Behavioral Target Generalization, Recommender Systems, CTR Prediction

1 Introduction

In recommender systems, the performance of the click-through rate (CTR) makes great effects on user experience and thus influences the final revenue of products. The main challenge of modeling CTR prediction is elaborately modeling users' preferences from their historical behaviors.

Recently, various CTR models are proposed for sequential recommendations with users' historical behaviors. GRU-BPR [1] and its variant [2] employ a Gated Recurrent Unit (GRU) [3] to model behavioral sequences for the session-based recommendation. Then, Bert4Rec [4] deems that unidirectional architectures (the above GRU-based methods) restrict the power of hidden representations, and thus models sequences with a bidirectional self-attention network trained through Cloze tasks. A similar idea also has been proposed in BST [5] which incorporates the Transformer [6] to encode behavior sequences in the learning.

Though these models are prevalent and effective for recommending highly relevant items to users, we argue that their sequential learning methods are not sufficient to learn optimal representations of user behaviors and inevitably damage the long-term user experience. The main limitations include: (a) Highly skewed recommendations. Items with highly similar attributes have more chances of exposure, which may make users feel monotonous to the recommended contents. (b) Inadequate interest explorations. Systems should be designed to help users to explore their interests, e.g., by recommending items that are not quite relevant to their historical interactions but of potential interest to them. An effective interest exploration strategy will be beneficial for improving the efficiency of Page View (PV), Unique Visitor (UV), and CTR. Hence, it is crucial to explore users' latent interests beyond their historical behaviors for content recommendations.

However, it is not straightforward to find out the latent interested items of users, particularly in the ***session-based recommendation*** where the session click information of each user is limited. One intuitive way is randomly generating a set of new items for each user outside his/her behavioral sequence, while this way could damage user experience since these items may be completely unrelated with the user. Therefore, for user interest exploration, it is necessary to take both user historical behaviors and exploratory item relations into consideration simultaneously.

To address this problem, we propose a heterogeneous graph enhanced sequential neural network (HGESNN) for personalized interest exploration in CTR prediction. Specifically, inspired by the success of Heterogeneous Graph Neural Networks (HGNN) [7], [8] in automated information propagation, we firstly model items and their associated relations as the *item-item graph* based on the user historical preferences. Then, we capture the relations between items and tags (each item contains one or more tags) as the *item-tag graph*. Last, we can construct the meta-path [9], a widely used structure to capture the semantics of two objects connecting by a composite relation, to search out users' potential interests. As illustrated in Figure 1 (a), the heterogeneous

J. Chen, Y. Li, and K. Wu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518057, China. Email: {junyangchen, yuanmanli, Wu}@szu.edu.cn
Z. Gong is with State Key Laboratory of Internet of Things for Smart City, Department of Computer Information Science, University of Macau, Macau. Email: fstzgg@umac.mo
H. Zhang, H. Yu, J. Zhu, and G. Fan are with the Platform and Content Group, Tencent Inc., Shenzhen 518057, China. Email: {alexhjzhang, willsyu, johannzhu, gefan}@tencent.com;
X. Wu is with the Department of Comuting, The Hong Kong Polytechnic University, xiao-ming.wu@polyu.edu.hk;

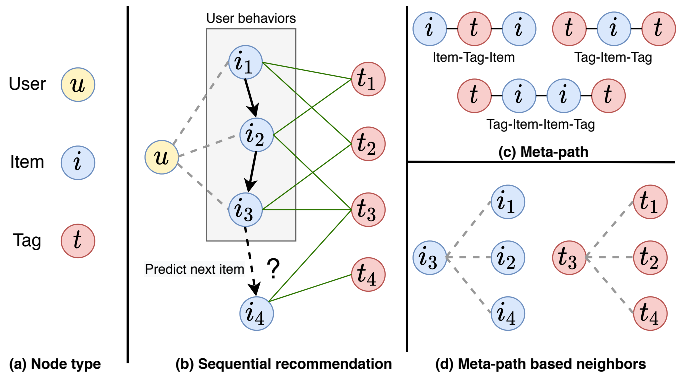


Fig. 1: An example of a heterogeneous graph constructed by user behavior sequences and item-tag relations. (a) Types of nodes. (b) Item recommendation based on the sequential behavior. (c) Three meta-paths involving item-tag relations. (d) Meta-path based neighbors of the item and tag nodes.

graph contains three types of nodes including user, item, and tag. Figure 1 (b) reveals the task of the sequential recommendation: to predict whether i_4 should be recommended to the user given his historical sequence $i_1 \rightarrow i_2 \rightarrow i_3$, where t_1, t_2, t_3, and t_4 are the tags associated with the items. Figure 1 (c) shows three meta-paths that capture different composite relations between items or tags. One can find meta-path based neighbors of the item and tag nodes as shown in Figure 1 (d), which helps to predict user latent interests (e.g., i_4) under the assumption that users would be interested in items with the same tag. In summary, we believe that using a more sophisticated graph model by introducing item nodes, tag nodes, and different types of paths can overcome the limitations of previous CTR prediction algorithms, since it enables us to encode different types of item relations inside and outside the historical preferences to explore user latent intentions. The main contributions of this paper are as follows:

- In this paper, we focus on personalized interest exploration in session-based CTR prediction tasks, and propose a heterogeneous graph enhanced sequential neural network (HGESNN) to explore the latent interests of users.
- Apart from taking sequential behaviors as interests, in the training process, we consider different types of item relations for behavioral target generalization. HGESNN can exploit the meta-path based neighbors of items to improve the recommendation results.
- Extensive experiments on both public and industrial datasets demonstrate that HGESNN outperforms the state-of-the-art models, including RNN-based models and Transformer-based models, validating the effectiveness of our method. Notably, online experiments on our Image-Text feed recommendation show that HGESNN can obtain 6.28%, 6.82%, and 4.77% improvements of CTR predictions in news, novels, and entertainment contents, respectively. The proposed method has been tested and officially deployed in our main traffic.

The rest of this paper is organized as follows. In Section 2, we introduce our proposed HGESNN model. We discuss experimental results in Section 3 and present the related works in Section 4. Section 5 concludes our work.

2 Proposed Model

2.1 Problem Statement

In sequential recommendations, given the historical behavior of user u, without loss of generality, we aim to predict the item that u will click at the next time step. Moreover, we perform personalized user interest exploration on the fly.

2.2 Model Overview

Figure 2 gives an overview of our proposed HGESNN model for personalized interest exploration. HGESNN mainly consists of three parts: heterogeneous graph representation (Section 2.4), behavioral target generalization (Section 2.6), and node sequential training (Section 2.7). In the followings, we first elaborate on how we construct the item graph from user historical behaviors (Section 2.3), and then present the representation learning method and behavioral target generalization with meta-path based neighbors. Afterwards, we introduce Transformer layer [6] for sequential training. Compared with the left-to-right unidirectional architectures in RNN based models [1], the Transformer employs a bidirectional attention mechanism on modeling sequences [10] and show its success in learning more powerful representations in recommendation tasks [4].

2.3 Construction of Behavioral Item Graph

In this part, we introduce the construction of the behavioral item graph from user historical sequences. In practice, it is difficult to utilize all historical information of users in the training. One of the common ways is to truncate the user behaviors within a certain time window (e.g., one hour) as session-based sequences as shown in Figure 2 (a). Then, two items are connected by an undirected edge if they occur consecutively in the sequences. For example, as shown in Figure 2 (b), item D and item A are connected since user u_1 clicks them consecutively. As such, we can capture the global transitive dependencies among items across all user sequences which carry rich semantic information for behavioral target generations. Besides, since each item is associated with multiple tags as shown in Figure 2 (c), we can search out its meta-path based neighbors for behavioral target generalization. Here item C is the meta-path neighbor of item B generating by *Item-Tag-Item*. After that, we conduct user interest exploration by regarding item C as the latent target prediction as shown in Figure 2 (d). More details of the heterogeneous graph embedding learning will be introduced in the following sections.

2.4 Unified Heterogeneous Graph Representations

Based on user historical preferences, we can model items and their co-occurrences as the *item-item* graph. Besides, since each item contains one or more tags, we can extract the relations between items and tags as the *item-tag* graph. However, it is not straightforward to fuse two different types of relations. To address this challenge, we propose ***message propagation step*** and ***message aggregation step*** to fuse two types of nodes and relations in the representation learning.

2.4.1 *Message Propagation Step.*

There are two types of nodes in the graphs, i.e., items and tags. For each item, the message propagation is performed between a centered node (e.g., i_A in Figure 2 (c)) and its neighbors including direct neighbors (i_D and i_B) and meta-path based neighbors

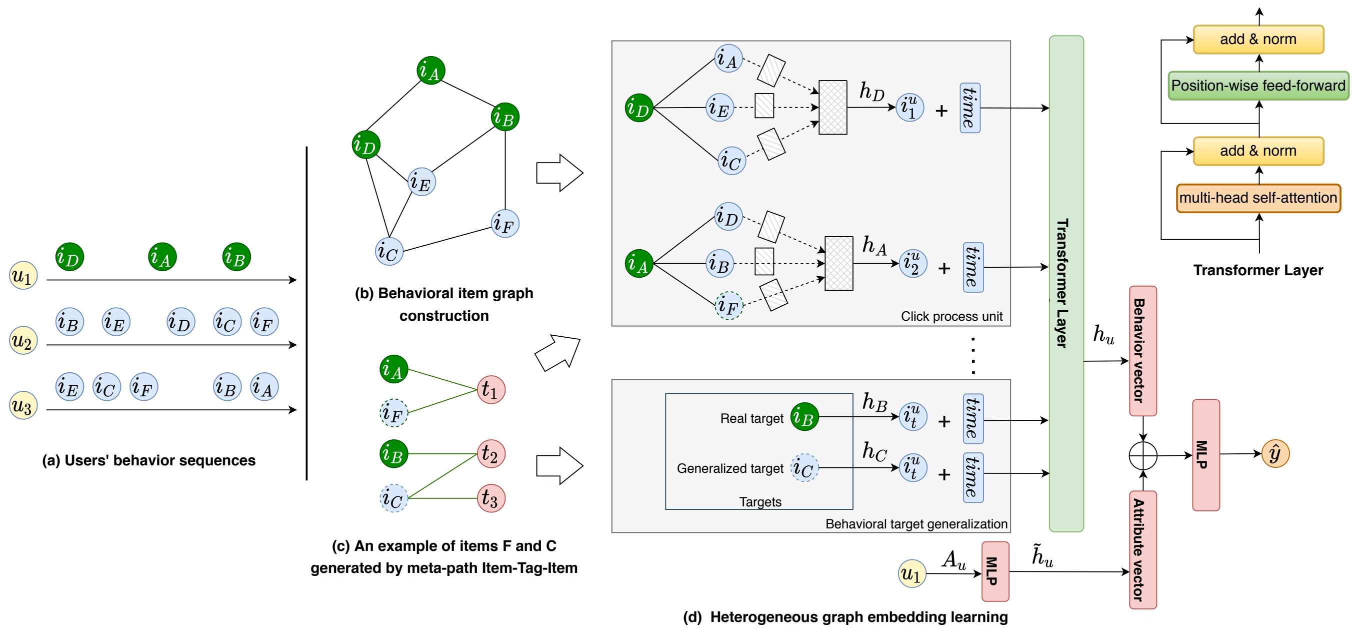

(a) Users' behavior sequences
(b) Behavioral item graph construction
(c) An example of items F and C generated by meta-path Item-Tag-Item
Click process unit
Real target
Generalized target
Targets
Behavioral target generalization
Transformer Layer
Behavior vector
Attribute vector
MLP
add & norm
Position-wise feed-forward
multi-head self-attention
time
(d) Heterogeneous graph embedding learning

Fig. 2: Overview of our proposed HGESNN model: (a) Users' behavior sequences; (b) Behavioral item graph construction; (c) An example of items F and C generated by the meta-path Item-Tag-Item; (d) Heterogeneous graph embedding learning with the Transformer layer for behavioral target generalization.

(i_F). As linear transform has been proven to be effective at encoding features from local structured neighbors [11], we define the message from node v_j to v_i with a transformation as follows:

$$\boldsymbol{m}_{v_i \leftarrow v_j} = \boldsymbol{M}_v \boldsymbol{h}_{v_j} = f_m(e_{v_j v_i}, \boldsymbol{h}_{v_j}) \cdot \boldsymbol{h}_{v_j}, \tag{1}$$

where $\boldsymbol{m}_{v_i \leftarrow v_j}$ is the message from node v_j to v_i with a dimension d, $\boldsymbol{M}_v \in \mathbb{R}^{d\times d}$ is the transformation matrix, $e_{v_j v_i}$ denotes the relation types with one-hot encoded (e.g., direct neighbors $\{0,1\}$ or meta-path generated neighbors $\{1,0\}$), $f_m(\cdot)$ takes as inputs both the relation type $e_{v_j v_i}$ and the representation of neighboring node $\boldsymbol{h}_{v_j} \in \mathbb{R}^d$ and outputs the transformation matrix $\boldsymbol{M}_v$. We concatenate $e_{v_j v_i}$ and $\boldsymbol{h}_{v_j}$, then use a Multi-Layer Perception (MLP) to learn the mapping. The detail of $f_m(\cdot)$ is defined as follows:

$$f_m(e_{v_j v_i}, \boldsymbol{h}_{v_j}) = \text{MLP}(e_{v_j v_i} \oplus \boldsymbol{h}_{v_j}), \tag{2}$$

where $\oplus$ represents the concatenation operation.

2.4.2 Message Aggregation Step.

After receiving messages propagated from the centered node's neighbors, we can aggregate them with different aggregators. In the following, we introduce *Mean* and *Attention* aggregators, respectively.

Mean Aggregator. We average the node's neighbor information as follows:

$$\boldsymbol{h}_{v_i} = \sigma\bigg(\boldsymbol{W} \cdot \textbf{CONCAT}(\boldsymbol{h}_{v_i}, \frac{1}{|\mathcal{N}_{v_i}|} \sum_{v_j \in \mathcal{N}_{v_i}} \boldsymbol{m}_{v_i \leftarrow v_j})\bigg), \tag{3}$$

where $\mathcal{N}_{v_i}$ is the set of neighbors of node v_i, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is a shared weight matrix, and σ is an activation function, e.g., ReLU [12]. The mean aggregator is nearly equivalent to the convolutional propagation rule used in GCN framework [13] by averaging the neighbor influences of centered nodes.

Attention Aggregator. For each node, we leverage the attention [6] mechanism to learn the importance weights of its neighbors. Given a node pair (v_i, v_j), the weight coefficient α_{v_i,v_j} can be defined as follows:

$$\alpha_{v_i,v_j} = \frac{\exp\Big(\sigma(\mathbf{a}^T \cdot [\boldsymbol{W}\boldsymbol{h}_{v_i} || \boldsymbol{W}\boldsymbol{m}_{v_i \leftarrow v_j}])\Big)}{\sum_{v_k \in \mathcal{N}_{v_i}} \exp\Big(\sigma(\mathbf{a}^T \cdot [\boldsymbol{W}\boldsymbol{h}_{v_i} || \boldsymbol{W}\boldsymbol{m}_{v_i \leftarrow v_k}])\Big)}, \tag{4}$$

where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is a shared weight matrix, and $\mathbf{a} \in \mathbb{R}^{2d}$ denotes a weight vector. Then, the representation of node v_i is obtained by aggregating the messages passing from its neighbors with the weight coefficients as follows:

$$\boldsymbol{h}_{v_i} = \sigma\bigg(\sum_{v_j \in \mathcal{N}_{v_i}} \alpha_{v_i,v_j} \boldsymbol{W}\boldsymbol{m}_{v_i \leftarrow v_j}\bigg). \tag{5}$$

Similar to [6], we can adopt multi-head attention to stabilize the learning process of self-attention. Specifically, we repeat the aggregation of Eq. (5) K times and concatenate the embeddings as follows:

$$\boldsymbol{h}_{v_i} = \Big\Vert_{k=1}^{K} \sigma\bigg(\sum_{v_j \in \mathcal{N}_{v_i}} \alpha^k_{v_i,v_j} \boldsymbol{W}^k \boldsymbol{m}_{v_i \leftarrow v_j}\bigg). \tag{6}$$

Up to now, we have introduced how we can aggregate the representations of nodes as shown in Figure 2 (d). Before feeding them into the Transformer layer, we need to consider the order of the input sequences.

2.5 Time-Diff Embedding Layer

To exploit the sequential information of the inputs, we need to inject positional embeddings into the graph representations. However, some drawbacks exists in the previous methods. The original positional embedding in [6] utilizes fixed sinusoid embeddings which restricts the expression of representations. After that, Bert4rec [4] introduces learnable embeddings for achieving better performance. Nevertheless, this method imposes a restriction on the maximum length of sequences and need to truncate the overlength ones. Hence, in this work, we make a compromise by scaling the time differences between items into $(0, N)$, where N is set to 150 in the experiments. Then, we propose to use a time-diff embedding matrix $\boldsymbol{P} \in \mathbb{R}^{N \times d}$ to encode the input representations. Specifically, given $\boldsymbol{h}_{v_i}$, we make the following summation:

$$\boldsymbol{h}_{v_i} = \boldsymbol{h}_{v_i} + \boldsymbol{p}_i, \tag{7}$$

where $\boldsymbol{p}_i \in \boldsymbol{P}$ is the d-dimensional time-diff embedding with index i calculated by the time difference between node v_{i-1} and v_i. Especially, for the first node in the sequences, its $\boldsymbol{p}_i$ is set to $\boldsymbol{p}_0$.

2.6 Behavioral Target Generalization

As mentioned in the introduction section, different from the existing CTR models, one of our main tasks is to explore the interests of users beyond their historical interactions to make recommendations. To achieve this task, we firstly propose to construct a heterogeneous graph to encode the item and tag relations inside and outside the user historical preferences. Then, we adopt a common assumption that users might be interested in items under the same tag. Base on this assumption, for each target prediction in the training process, we additionally generalize a potential target which is not present in the original sequences for auxiliary training. For example, in Figure 2 (d), item C is not clicked by u_1, but it has the same tag with item B, i.e., B and C are latent neighbors based on the meta-path *Item-Tag-Item*. Thus, when using the sequence $i_D \rightarrow i_A$ to predict i_B, we also use it to predict i_C. We call this process behavioral target generalization.

Note that there are two main differences compared with the graph session-based [14] and heterogeneous graph-based [15] methods: The first one is we perform message propagation and aggregation steps to fuse the heterogeneous information of a target item and its neighbors for obtaining node representations. Then, we exploit the meta-path based neighbors of items to perform personalized user interest exploration (the Q&A 1 in Section 3.5 has demonstrated the effectiveness of behavioral target generalization). The second one is we adopt a bidirectional attention mechanism to train the obtained node representations. Compared with the unidirectional training model (e.g, SR-GNN [14], our method can learn more powerful representations in recommendation tasks (we report the comparison in the Q&A 3 of Section 3.5).

2.7 Transformer Layer

As illustrated in Figure 2 (d), we feed the hidden representations of items into the Transformer Layer [6]. In general, the Transformer layer contains two important components, *Multi-Head Self-Attention* and *Position-Wise Feed Forward*, which are described as follows.

2.7.1 Multi-Head Self-Attention.

Given an input sequence with length n_u, and the hidden representation $h^l_{v_i} \in \mathbb{R}^d$ of item i at layer l, we can obtain a matrix $\boldsymbol{H}^l \in \mathbb{R}^{n_u \times d}$ to denote this sequence, where $h^l_{v_i} \in \boldsymbol{H}^l$. Then we employ multi-head self-attention by projecting $\boldsymbol{H}^l$ into n subspaces, which allows the model to jointly attend to information from different representation subspaces at different positions in the sequence. The formulation is shown as follows:

$$\begin{aligned}&\text{Multi-Head}(\boldsymbol{H}^l) = \text{Concat}(\text{head}_1, ..., \text{head}_n)\boldsymbol{W}^O,\\&\text{where head}_i = \text{Attention}(\boldsymbol{H}^l\boldsymbol{W}_i^Q, \boldsymbol{H}^l\boldsymbol{W}_i^K, \boldsymbol{H}^l\boldsymbol{W}_i^V),\end{aligned} \tag{8}$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{d \times d/n}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d \times d/n}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d \times d/n}$, and $\boldsymbol{W}_i^O \in \mathbb{R}^{d \times d}$ are learnable projection matrices. Moreover, the Attention function in Eq. (8) is defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d/n}})V, \tag{9}$$

where Q, K, V represent $\boldsymbol{H}^l\boldsymbol{W}_i^Q$, $\boldsymbol{H}^l\boldsymbol{W}_i^K$, and $\boldsymbol{H}^l\boldsymbol{W}_i^V$, respectively. Besides, $\sqrt{d/n}$ is a temperature used for scaling to avoid extremely small gradients [6].

2.7.2 Position-Wise Feed Forward.

To endow the model with nonlinearity expression and interactions among dimensions, a position-wise feed-forward network (FFN) layer [6] is adopted to the outputs of the above multi-head self-attention. The FFN layer consists of two linear transformations with a ReLU activation in between as follows:

$$\text{FNN}(\boldsymbol{x}) = \max(0, \text{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1))\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{10}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times 4d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{4d \times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^{4d}$, and $\boldsymbol{b}_2 \in \mathbb{R}^d$ are learnable parameters.

2.8 Optimization Objective

Our goal is to predict the likelihood of a user u clicking a next item i given his/her previous clicks. To this end, we feed a sequence of the previous clicks before item i into the Transformer, and take the last output $\boldsymbol{h}_u$ as the representation of the sequence, as shown in Figure. 2(d). In addition, we want to utilize statistical attributes from user long-term historical sequences, e.g., the tags of the top-k items that a user clicked within 30 days. We feed the statistical features $\boldsymbol{A}_u$ of user u into a MLP to obtain the representation $\tilde{\boldsymbol{h}}_u$. Then, we concatenate the embeddings of $\boldsymbol{h}_u$ and $\tilde{\boldsymbol{h}}_u$ and use another MLP to get the prediction score:

$$\hat{y}_{u,i} = \sigma(\text{MLP}(\boldsymbol{h}_u \oplus \tilde{\boldsymbol{h}}_u)), \tag{11}$$

where σ is an activation function, $\oplus$ denotes concatenation, and $\hat{y}_{u,i}$ is the prediction score of user u clicking item i next given previous clicks. Finally, we optimize our model with a widely used Bayesian personalized ranking (BPR) loss [16]. BPR is a pairwise loss that pushes the model to rank positive samples higher than negative samples. Given a historical interaction sequence of user u, at any point of the sequence, the BPR loss is defined as:

$$\mathcal{L} = \sum_{(u,i,j) \in \mathcal{T}} \log\sigma(\hat{y}_{u,i} - \hat{y}_{u,j}), \tag{12}$$

where $\mathcal{T}$ denotes a training batch, $\hat{y}_{u,i}$ is the prediction score of a positive sample (item i clicked by user u given previous clicks), and $\hat{y}_{u,j}$ is the score of a negative sample (e.g., an item from other sequences in the training batch).

TABLE 1: The statistics of datasets.

Dataset	1st-day	3st-day	5st-day
Total sequences	235,086	241,368	240,140
Training clicks	1,913,269	1,978,580	1,979,876
Testing clicks	2,391,586	2,473,225	2,474,845
Training sequences	105,789	108,616	108,063
Validation sequences	11,754	12,068	12,007
Testing sequences	117,543	120,684	120,070
Item size	236,976	246,252	252,169
Tag size	37,575	38,634	38,970
New items in test set	56,363	58,482	59,685
New tags in test set	5,433	5,480	5,452
Density	8.59e-5	8.32e-5	8.17e-5

3 Experiments

3.1 Datasets

We collect real-world large-scale datasets of web feed from QQ browser homepage[1] including news, novels, and entertainment contents. Specifically, we extract 3-day data with an interval of one day so as to estimate the service of the trained model on the next day. For each day, we collect 8-hour user behaviors that cover the peak of user access. After removing those user behavioral interactions with less than 5 items, we can obtain about 0.71 million historical sequences in total, consisting of around 13 million clicks with 0.73 million unique items and 0.11 million unique tags. More details of the statistics are shown in Table 1. We separate the total sequences into training sequences, validation sequences, and testing sequences with ratio 4:1:5. In general, our datasets have the following unique characteristics. First, the datasets are large enough, containing millions of historical clicks in both training and testing sets. Second, our datasets contain up to one-quarter of new items in the test set. Third, the density (average number of items in each sequence) / (the size of unique item set) of the datasets is extremely sparse. These characteristics of data bring great challenges to our model.

3.2 Comparison Methods

To verify the effectiveness of our proposed method, we employ several state-of-the-art approaches as the baselines:

- **GRU4Rec** [2]: It adopts one Gated Recurrent Unit (GRU) layer with ranking based loss for session-based recommendations.
- **Two-layer GRU Network**: Similar to [2], we use a two-layer GRU network to model user sequential behaviors.
- **Bert4Rec** [4]: It uses a deep bidirectional Transformer to model user behaviors, and achieves state-of-the-art performance on sequential recommendations.
- **HGESNN-meanAgg**: It is our proposed method with a mean aggregator which averages the node information passing from centered nodes' neighbors.
- **HGESNN-attAgg**: It is another method with an attention aggregator that leverages an attention mechanism to learn the importance weights of neighbors.

1. https://www.qq.com/

Note that there are many other fancy session-based recommendation models but we do not consider them here, because either their performance is inferior to these baselines as shown in corresponding papers or they are incapable of handling large-scale datasets that consume too much time on training and inferring during online serving.

3.3 Detailed Implementation

We implement our method with Tensorflow [17]. For all models, we uniformity set the embedding size as 32. Before training, we randomly initialize the model parameters with a Gaussian distribution, and optimize the models with mini-batch Adam [18]. The learning rate is set to 1e-3, the batch size is 1024, and the maximum sequence length is set to 50. For Bert4Rec and our HGESNN, we both set the number of Transformer layer and the number of attention head as 4. Besides, in HGESNN, we set the number of sampled neighbors as 5 and 15 for Mean Aggregator and Attention Aggregator, respectively.

3.4 Evaluation Metrics

As online resources are limited, we cannot perform online experiments for all mentioned methods. Thus, we conduct offline experiments to pick up the best baseline for further online comparison. The experimental metrics are as follows.

Offline Metrics. In offline experiments, we use *Hit Ratio* (HR) and *Mean Reciprocal Rank* (MRR) to evaluate the performance of different models. To apply these metrics, we adopt a leave-one-out evaluation which has been widely used before [19], [20], [21]. Specifically, for each sequence in the test set, we hold out the last item as the ground truth and treat the item just before the last as the inputs. Since we only have one ground-truth item for prediction in each sequence, HR@k is equivalent to Recall@k and proportional to Precision@k. Besides, MRR is equivalent to Mean Average Precision (MAP) [4]. Here we report HR with $k \in \{5, 10\}$ and MRR metrics.

Online Metrics. In online experiments, there are four widely used indices for evaluation, including *Click Page View* (CPV), *Exposure Page View* (EPV), *Click Unique Visitor* (CUV), and *Exposure Unique Visitor* (EUV). Based on these indices, we can obtain popular online metrics, *Click-Through Rate* (CTR), *Per Click Capita Consumption* (PCC), and *Per Exposure Capita Consumption* (PEC), as follows:

$$\text{CTR} = \frac{CPV}{EPV}, \ \text{PCC} = \frac{CPV}{CUV}, \ \text{PEC} = \frac{CPV}{EUV}. \tag{13}$$

Exploration Rate (ER). One of our purposes is to explore users' interests by behavioral target generalization. Therefore, we define a metric named *Exploration Rate* which is represented by the number of content subjects that a user has not clicked in the past 30 days. Overall, for all these metrics including offline and online, the higher values represent the better performance that models can achieve.

3.5 Experimental Results and Analysis

Offline Evaluation. The performances of the proposed HGESNN and the baselines are reported in Table 2. Bold numbers denote the best results and the underlined ones represent the second best. In general, we can observe that:

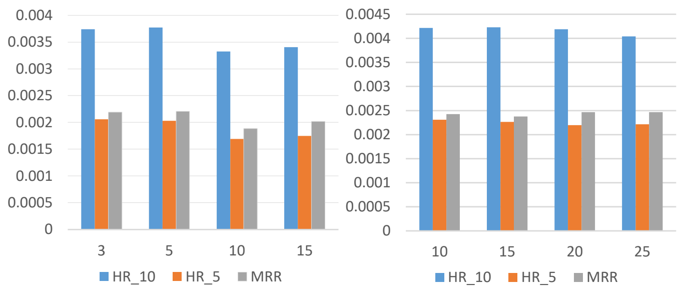
(1) Among the results of the baselines, *BERT4Rec* achieves the second-best performance in most cases on all the datasets.

TABLE 2: Results of the next-item prediction on industrial datasets. The highest scores are highlighted in boldface, while the underlined ones are the second best. Improv. denotes the improvements achieved by our best method over the strongest baselines, where * represents the results are statistically significant with p-value < 0.05.

Method	1st-day			3rd-day			5th-day		
	HR@10(%)	HR@5(%)	MRR(%)	HR@10(%)	HR@5(%)	MRR(%)	HR@10(%)	HR@5(%)	MRR(%)
GRU4Rec	0.323	0.154	0.187	0.426	0.218	<u>0.249</u>	0.306	0.159	0.194
Two-layer GRU Network	0.352	0.205	0.211	0.328	0.186	0.213	0.298	0.155	0.189
BERT4Rec	<u>0.436</u>	<u>0.218</u>	<u>0.254</u>	<u>0.436</u>	<u>0.229</u>	0.247	<u>0.352</u>	<u>0.167</u>	<u>0.205</u>
HGESNN-meanAgg	**0.467***	0.239	0.260	0.490	0.269	0.282	0.343	0.178	0.206
HGESNN-attAgg	0.464	**0.274***	**0.282***	**0.512***	**0.271***	**0.288***	**0.423***	**0.227***	**0.237***
Improv.	+7.11%	+25.69%	+11.02%	+17.43%	+18.34%	+15.66%	+20.17%	+35.93%	+15.61%

TABLE 3: Ablation Test of Behavioral Target Generalization.

Method	3rd-day			5th-day		
	HR@10(%)	HR@5(%)	MRR(%)	HR@10(%)	HR@5(%)	MRR(%)
HGESNN-meanAgg	0.449	0.238	0.247	0.335	0.175	0.198
w/o BTG	-8.37%	-11.52%	-12.41%	-2.33%	-1.69%	-3.41%
HGESNN-attAgg	0.484	0.267	0.282	0.407	0.219	0.226
w/o BTG	-5.47%	-1.48%	-2.08%	-3.78%	-3.52%	-4.64%



(a) HGESNN-meanAgg (b) HGESNN-attAgg

Fig. 3: Impact of the number of sampled neighbors.

Specifically, *BERT4Rec* improves *GRU4Rec* by 34.98%, 2.35%, and 15.03% in terms of HR@10 on the three datasets; and outperforms *Two-layer GRU network* by 23.86%, 32.93%, and 18.12%, respectively. This is because both *GRU4Rec* and *Two-layer GRU network* are left-to-right unidirectional models which predict the next items sequentially, while *BERT4Rec* is a bidirectional model with the Transformer layer, suggesting that the self-attention architecture is more powerful to learn influences among items for sequential recommendation. These improvement results demonstrate the effectiveness of adopting the Transformer layer in our model design. Moreover, we notice that *GRU4Rec* generally can obtain better performance than *Two-layer GRU network*, e.g., on the datasets of 3rd-day and 5th-day. One possible reason is that the GRU network may get over-fitting with more layers in training.

(2) The proposed methods, including *HGESNN-meanAgg* and *HGESNN-attAgg*, consistently outperforms the other baselines on all datasets under various metrics. For example, *HGESNN* gains 14.90% HR@10, 26.65% HR@5, and 14.10% MRR improvements on average against the strongest baseline. In Table 2, two-sided significant tests show the improvements are statistically significant. Moreover, *HGESNN-attAgg* generally achieves better performances than *HGESNN-meanAgg*, which also demonstrates the validity of the attention mechanism.

Though the above results show the effectiveness of the designed models, some research questions may be raised:

Question 1: *Do the gains come from the proposed Behavioral Target Generalization (BTG)?*

Answer 1: We perform an ablation study to verify the effect of BTG in *HGESNN* as shown in Table 3. We report the results on two datasets due to space limitations. The results show that the performances drop by as much as 12.41% in terms of MRR, proving that BTG plays a critical role in the sequential recommendation.

Question 2: *What is the impact of the number of sampled neighbors in the proposed method?*

Answer 2: We investigate the impact of the number of sampled neighbors on our methods with the 5th-day dataset. As shown in Figure 3, we can see that: (1) The performance of *HGESNN* benefits from a suitable size of sampled neighbors; (2) A large size may involve noisy neighbors and degrade the performance of *HGESNN-meanAgg*; (3) In contrast, *HGESNN-attAgg* is more robust to the number of sampled neighbors. One possible reason is that the attention mechanism can automatically learn the importance weights of neighbors to restrict the noisy neighbors.

Question 3: *Would HGESNN perform well comparing with the baselines on public datasets?*

Answer 3: As shown in Table 4, we conduct experiments on Movielens [22], a popular benchmark dataset for evaluating recommendation algorithms. Note that here we include SR-GNN [14] modeling session sequences as graph-structured data into the comparison of the public dataset. We exclude it in our industry datasets because its construction of session graphs takes huge memory consumption. The highest scores are highlighted in boldface. Our methods can achieve around 10% to 28% improvement on the second-best scores (the underlined ones).

Question 4: *How does HEGSNN handle the online serving and perform for online A/B testing?*

Answer 4: Online A/B testing was conducted in our web browser from August 13 to 26. During the online serving, we update our model from scratch every one hour using the past 8-hour user behavior data. We truncate the number of nodes' neighbors to 100 with an importance sampling strategy, and save the adjacent matrices of graphs into tensors. Besides, when encountering unknown items, we uniformly assign them a randomly generated embedding for simplicity. As shown in Table 5, we select the *BERT4Rec* model as the base model since it is already used in our real system, and we adopt *HGESNN-attAgg* for comparison because it achieves the best performance in our offline experiments. We observe that our method achieves performance improvements in all metrics. For example, we gain improvements of 6.28%, 6.82%, and 4.77% in CTR on the Image-Text Feeds of news, novels, and entertainment contents, respectively. Moreover, we achieve 2.46% ER improvements, which indicates that more new contents that users are interested in can be exposed to exploring user interests.

Question 5: *What are the main differences between HGESNN and BERT4Rec?*

Answer 5: In general, there are two main differences for our method compared with BERT4Rec. On the one hand, we perform message propagation and aggregation steps to fuse the heterogeneous information of a target item and its neighbors for obtaining node representations before feeding into sequential training. On the other hand, we exploit the meta-path based neighbors of items to perform personalized user interest exploration (the Q&A 1 has demonstrated the effectiveness of behavioral target generalization).

Question 6: *What is the effect of other meta-path selection for behavioral target generalization?*

Answer 6: In our method, we use the *Item-Tag-Item* meta-path to generate the neighbors of target items as user latent interests for training. We have tried using more meta-paths for this purpose, e.g., *i-t-t-i* and *i-t-i-t-i* (i denotes item and t is tag), but the model performance degrades after adding these meta-paths one by one. We conjecture that the generated targets may become too generalized as the length of the meta-path increases. Thus, there is a trade-off between target generalization and target prediction accuracy.

Question 7: *Give an example to illustrate why HGESNN performs better on highly skewed data than sequential recommendation models.*

Answer 7: In our Short-Video feed recommendation, we may recommend a short video about "Naruto" to users. This video contains multiple tags, including Ninjutsu, ninja, and Ramen. Without our method, the system may continually recommend animation about action scenes. The proposed method is capable of mining the relation between "Ramen" and "Food Program", and thus may benefit the long-term user experience.

Question 8: *Can the proposed method be extended to other sequential neural models?*

Answer 8: We extend our method to DIEN [23], a GRU-based sequential neural network, used in our real system for Short-Video feed recommendation. We conduct online A/B testing from Jul. 19 to 27. Experiments show that our method obtains 12.92%, 5.88%, and 35.09% gains on CTR, PCC, and PEC, respectively. And the performance of ER increases by 21.87%. These improved results verify the effectiveness and expansibility of our method.

TABLE 4: Results on public datasets – MovieLens (ML-20m).

Method	ML-20m		
	HR@10(%)	HR@5(%)	MRR(%)
GRU4Rec	<u>4.613</u>	2.131	2.025
Two layer GRU Attention	4.197	2.165	<u>2.174</u>
SG-GNN	4.302	2.158	2.121
HGESNN-meanAgg	4.821	2.334	2.257
BERT4Rec	4.517	<u>2.171</u>	2.135
HGESNN-meanAgg	4.821	2.334	2.257
HGESNN-attAgg	**5.107**	**2.793**	**2.484**
Improv.	+10.71%	+28.65%	+14.26%

TABLE 5: Results from Online A/B testing.

Image-Text Feeds	CTR Gain	PCC Gain	PEC Gain	ER Gain
News	+6.28%	+4.16%	+17.66%	
Novels	+6.82%	+9.55%	+5.62%	+2.46%
Entertainment Contents	+4.77%	+1.62%	+12.65%	

4 Related Work

This work draws on two research areas: (1) sequential recommendation and (2) graph neural networks.

Sequential Recommendation. Early work on recommendation systems usually employs Collaborative Filtering to model users' behavioral sequences [24], [25]. While these methods usually ignore the order information of the sequences, they are not suitable for sequential recommendations. Recently, Recurrent Neural Network (RNN) [26] and its variants, GRU [27] and LSTM [28], are becoming prevalent for modeling user behaviors [29]. For example, NARM [30] incorporates an attention mechanism into RNN for capturing both users' preferences and their main purposes in sessions. GRU4Rec [2] incorporates GRU with designed loss functions that are tailored to sequential recommendations. The main idea of these methods is to encode users' behaviors into embeddings which reflect their preferences for making predictions. Nevertheless, these recurrent networks constrain the expression ability to model mutual influences between items in a session. Thus, inspired by the Transformer architecture [6], SASRec [20] and BERT4Rec [4] propose to use it to learn the mutual influence scores of items in a sequence. Among them, BERT4Rec is most related to our work. However, our model can explore the interests of users beyond their historical interactions by target behavioral generalization with meta-path constructions.

Graph Neural Networks. Many research efforts have demonstrated the power of Graph Neural Networks (GNNs) [31] to model graph-structured data. For example, the variants of GNNs including GCN [32], GAT [33] and GraphSAGE [13] show ground-breaking performance on node representation learning. Recently, GNNs are widely applied for recommendations. For example, PinSage [34] incorporates GraphSAGE and achieves the largest application of deep graph embeddings, which paves the way for a new generation of web-scale recommender systems. MEIRec [8] exploits a metapath-guided GNN [7] to model complex objects and rich interactions of heterogeneous graph for search intent recommendation. However, all these methods are not designed for sequential recommendations since they ignore the orders in users' behaviors. Then, SR-GNN [14] proposes to model session sequences as graph-structured data and obtain more accurate item embedding by taking complex transitions of items into consideration. Fi-GNN [35] presents the multi-field features of advertising systems in a graph structure and incorporates a gated GNN [36] to learn sequential feature interactions. HyperRec [37] employs a hypergraph structure to represent the short-term correlations of items and adopts convolutional layers to learn the embedding. MA-GNN [38] applies a GNN method to model the short-term item correlations and uses a memory network to model the long-term item dependency. Nevertheless, all of these methods are left-to-right unidirectional models that cannot learn the mutual influences of items in a sequential list. Moreover, these approaches fail to consider behavioral target generalization in CTR predictions.

5 Conclusion

In general, we propose a GNN-based model, HGESNN, to explore the latent interests of users beyond their historical behaviors for sequential recommendations. HGESNN explicitly models item relations of all behavioral sequences with meta-path constructions, and utilizes meta-path guided neighbors to perform behavioral target generalization. Before sequential training, we introduce the message propagation and aggregation steps to fuse the heterogeneous information of a target item and its neighbors for obtaining node representations. Then, we employ the Transformer layer, a deep bidirectional sequential model, to predict next items. In this way, our method can exploit the meta-path based neighbors of items to perform personalized user interest exploration. Extensive results on offline and online experiments demonstrate the effectiveness and expansibility of our method.

References

[1] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[2] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 843–852.

[3] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[4] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1441–1450.

[5] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1–4.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[7] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[8] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2478–2486.

[9] M. Zhang, H. Hu, Z. He, and W. Wang, "Top-k similarity search in heterogeneous information networks with x-star network schema," *Expert Systems with Applications*, vol. 42, no. 2, pp. 699–712, 2015.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[13] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[14] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 346–353.

[15] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[20] W. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 197–206.

[21] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 565–573.

[22] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[23] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 5941–5948.

[24] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*. Springer, 2015, pp. 77–118.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[26] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 2017, pp. 152–160.

[30] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.

[31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[34] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[35] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 539–548.

[36] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[37] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 1101–1110.

[38] C. Ma, L. Ma, Y. Zhang, J. Sun, X. Liu, and M. Coates, "Memory augmented graph neural networks for sequential recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5045–5052.