Collaborative Sequential Recommendations via Multi-view GNN-transformers

TIANZE LUO, School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
YONG LIU, Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore, Singapore
SINNO JIALIN PAN, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China

Sequential recommendation systems aim to exploit users' sequential behavior patterns to capture their interaction intentions and improve recommendation accuracy. Existing sequential recommendation methods mainly focus on modeling the items' chronological relationships in each individual user behavior sequence, which may not be effective in making accurate and robust recommendations. On the one hand, the performance of existing sequential recommendation methods is usually sensitive to the length of a user's behavior sequence (i.e., the list of a user's historically interacted items). On the other hand, besides the context information in each individual user behavior sequence, the collaborative information among different users' behavior sequences is also crucial to make accurate recommendations. However, this kind of information is usually ignored by existing sequential recommendation methods. In this work, we propose a new sequential recommendation framework, which encodes the context information in each individual user behavior sequence as well as the collaborative information among the behavior sequences of different users, through building a local dependency graph for each item. We conduct extensive experiments to compare the proposed model with state-of-the-art sequential recommendation methods on five benchmark datasets. The experimental results demonstrate that the proposed model is able to achieve better recommendation performance than existing methods, by incorporating collaborative information.

CCS Concepts: • **Information systems** → **Information systems applications**; *Information retrieval*; • **Computing methodologies** → *Artificial intelligence;*

Additional Key Words and Phrases: Recommender systems, sequential recommendation, graph neural networks, transformers

ACM Reference Format:
Tianze Luo, Yong Liu, and Sinno Jialin Pan. 2024. Collaborative Sequential Recommendations via Multi-view GNN-transformers. *ACM Trans. Inf. Syst.* 42, 6, Article 141 (June 2024), 27 pages. https://doi.org/10.1145/3649436

Sinno J. Pan thanks the Hong Kong Jockey Club Charities Trust to the JC STEM Lab of Integration of Machine Learning and Symbolic Reasoning for support.
Authors' addresses: T. Luo, School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore; e-mail: tianze001@ntu.edu.sg; Y. Liu, Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore, Singapore; e-mail: stephenliu@ieee.org; S. J. Pan, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China; e-mail: sinnopan@cuhk.edu.hk.

ACM 1046-8188/2024/06-ART141
https://doi.org/10.1145/3649436

1 INTRODUCTION

Effectively understanding users' sequential behaviors has been proven to be crucial for building sequential recommendation systems, which generate item recommendations to users based on their historical behavior sequences. Various deep learning techniques, e.g., **Recurrent Neural Networks (RNNs)** [16, 41], **Convolutional Neural Networks (CNNs)** [46], and attention mechanisms [23, 42, 45, 61, 74], have been applied for sequential recommendations. In general, these existing methods mainly focus on capturing the chronological relationships between items in each user's behavior sequence to predict her next interactions with items.

Although existing deep learning-based methods usually achieve state-of-the-art sequential recommendation performance, they may still suffer from the following deficiencies. *First*, conventional sequential recommendation methods, including RNN-based models, transformer-based models, and **Graph Neural Network (GNN)**-based models, usually utilize a unitary message passing strategy: they only encode information along each individual user behavior sequence to make predictions for this user. For example, RNN-based models iteratively encode the behavior history of a user; transformer-based models directly extract the information from the entire behavior sequence of a user through self-attention. They ignore the collaborative information among the behavior sequences of other similar users. If the user's behavior suddenly fluctuates, for example, the user interacts with un-relevant items out of curiosity, then the recommendation model may be disturbed and fail to make accurate predictions. *Second*, the performance of traditional sequential recommendation models, such as those presented in References [23, 32, 45], is sensitive to the length of user behavior sequences. An over-reliance on individual user histories can limit model performance, particularly when a user's behavior history is short. Thus, solely modeling the item chronological relationships in each individual user's behavior sequence is not enough for sequential recommendations.

In this work, we propose a novel sequential recommendation architecture, which can pass messages not only through each individual user behavior sequence but also through the global item dependency graph built from all users' behavior sequences. The global item dependency graph integrates user collaboration information, which aligns with the preferences of the target user, thereby enhancing the accuracy of predictions pertaining to the target user. This is motivated by the recent success of applying GNNs to exploit different item graphs, e.g., user-item interaction graph [14, 56], item knowledge graph [31], and item multi-modal graph [30], to model users' preference on items. The proposed architecture utilizes higher-order item dependency information, which can surpass the constraint that the length of a user behavior sequence is limited, and also construct more robust representations for each user's behaviours, beyond the user's chronological behavior patterns.

Figure 1 shows the message passing strategies of different sequential recommendation models. As shown in Figure 1(a), conventional RNN- and Transformer-based sequential recommendation models pass messages following the user behavior sequence [23, 45]. The graph-based sequential recommendation methods [35, 40, 55, 58, 62, 68] first build the item relation graphs and then extract information along the edges of the graphs with GNNs, as shown in Figure 1(b). Our proposed recommendation architecture, shown in Figure 1(c), passes messages between relevant items in the item dependency graph and also along the user behavior sequences. The proposed method can not only utilize the sequential behavior information but also the higher-order context information (i.e., item's dependency subgraph), which can provide additional information to the model and mitigate the issues caused by user behavior fluctuations. Specifically, the issue may happen when a user interacts with un-relevant items out of curiosity, for example, the user wants to buy a phone but clicks a computer advertisement instead. In this case, conventional sequential recommendation models such as SASRec [23] may be misled by this action, and have a higher chance

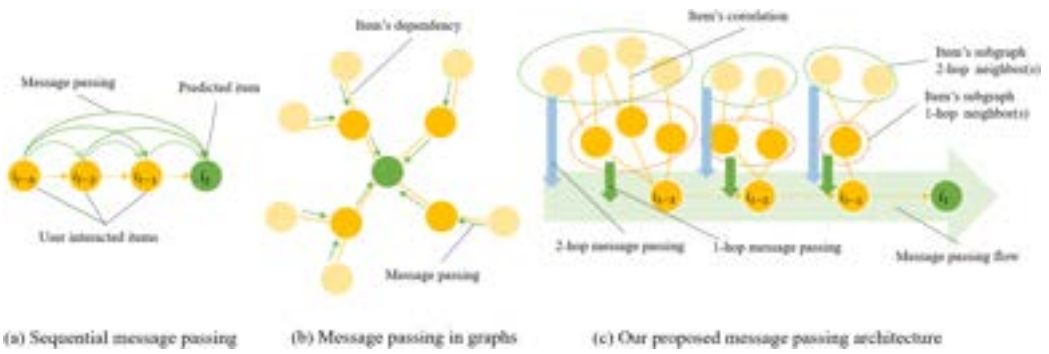

Fig. 1. Message passing in sequential recommendations. (a) Sequential message passing: Flow of messages is passed along the user behavior sequences. (b) Message passing on graphs: Messages are aggregated through the message passing on the graph topology. (c) Our proposed message passing architecture: User behavior messages are passed along both item-item graphs and user behavior sequences.

to recommend computers to the user. However, our model will be more robust to this fluctuation. This is because the impact on the message passing in the higher-order context will be much less than the impact of the message passing on the item sequence, as phones and computers are both in the electronics category and share many common higher-order context information. Thus, the higher-order information can enhance the tolerance of the model to user behavior fluctuations and improves the model's prediction accuracy.

However, in practice, it is challenging to jointly pass messages in the item dependency graph using GNNs and pass messages along the item sequences (i.e., a user's behavior sequence) using RNN or Transformer structures.

Typical GNN models [14, 24, 51, 56] need to process the entire item graph adjacency or Laplacian matrix, to obtain the localized representation of each node, with time complexity proportional to the total number of edges. However, Transformer structures only require traces of user behaviors in an individual user behavior sequence to obtain the sequence representation. Thus, directly stacking GNNs and Transformer structures is not efficient. In each batch, GNNs compute the localized representation for each node, but only a small amount of node features (along the user behavior trace) will be passed to the Transformer modules. Such a process may cause great computational cost, especially when processing large-scale item dependency graphs.

To solve the above issues, we propose a hierarchical graph aggregation mechanism to extract the representations of the 1 to K-hop sub-graphs of each node's neighborhood. Then, we pass the sub-graph representations to a transformer-based model. Meanwhile, we also form 1 to K-hop views of each item's neighborhood. The larger sub-graph, the more item dependency information is preserved. The smaller sub-graph, the more the chronological relationship of the current user behaviors is enhanced. Then, we aggregate the multiple views and obtain the final representation of each user behavior sequence. Note that the graph model incorporates global item relationships instead of the user-specific chronological relationships extracted by the sequential model. Compared to a single view, the multi-view architecture can form more comprehensive representations of the user behaviors.

Our main contributions are summarized as follows.

— We propose a hierarchical graph aggregation model to support efficient graph aggregation operations on graphs, and analyze its running complexity, comparing with commonly used graph aggregation methods.

— We apply the proposed hierarchical graph aggregation model to build a novel sequential recommendation framework, which has better capability of user behavior modeling than existing sequential recommendation models.
— We also design a multi-view architecture and propose the Dirichlet sampling method to improve the performance and robustness of the sequential recommendation model. We show the findings in the ablation studies.
— We conduct extensive experiments on five real datasets with comparisons to state-of-the-art sequential recommendation methods. The experimental results show that the proposed model obtains superior performance than baseline methods.

2 RELATED WORK

Sequential recommendation (SR) is one of the main streams of recommender systems, which focuses on exploiting users' sequential behaviors to make recommendations [11, 60, 66]. The sequential recommendation is based on the assumption that historical behaviors can reflect the trend of the user's preference and therefore, we can recommend proper items not only based on the user's preference but also based on the user's historical behaviors. Prior works in sequential recommendations use the Markov chain to model transitions among the user-item interactions in a user behavior sequence. For example, Feng et al. [12] proposed embedding Markov chains into Euclidean space and calculating transition probabilities between interactions based on their Euclidean distance.

Latent representation models, however, use learned embeddings of users and items to capture more implicit and complex dependencies between interactions. FPMC [43] combines matrix factorization and first-order Markov chain to model users' global preferences and short-term interests. Hidasi et al. [18] proposed a factorization framework for context-aware SR. References [53, 54] learn latent representations of users and items as input to a network and calculate interaction scores between users and items or successive user actions.

In recent years, DNN-based SR models, especially those using RNNs such as **Long Short-term Memory (LSTM)** [59], **Gated Recurrent Unit (GRU)** [7], as well as Transformers [50], have gained popularity due to their ability to model sequential dependencies effectively. These models incorporate multiple hidden layers to capture complex relationships among user-item interactions, resulting in more accurate recommendations. In addition to RNNs, Tang et al. [47] utilized CNNs to achieve top-K SR. Furthermore, inspired by the success of the Transformer model in NLP tasks, Kang et al. proposed SASRec [23], a deep SR model incorporating the self-attention mechanism from Transformer to capture long-term semantics and emphasize the most relevant interactions when making predictions. To improve the smoothness of SR models, Zhou et al. [77] designed a **multi-view smoothness (MVS)** optimization framework to smooth and enrich the one-hot representations of contexts and labels to better depict the underlying user preference. Recently, there has been a growing interest in applying GNNs to model high-order features and capture complex transitions over user-item interactions in a sequence [24, 51]. In general, from the architectural perspective, modern sequential recommendation models can be mainly categorized into RNNs architecture, transformers architecture and GNNs architecture.

2.1 RNN-based Methods

The RNN-based models are pioneer deep learning models to extract information from user behavior sequences [66]. Hidasi et al. [17] applied RNN to model the user behavior patterns by taking the user's clicked items into the RNN structure and outputting the prediction of the user's preference for the next item. Hidasi and Karatzoglou [16] further enhanced the GRU model by adopting a new sampling method and loss function. Xu et al. [65] designed recurrent convolutional neural

networks to hierarchically combine recurrent networks and convolutional networks to analyze users' sequential behaviors and provide further recommendations. Yu et al. [69] applied LSTM model with a time-aware controller and a content-aware to capture user's long-term and short-term preferences. Zheng et al. [73] used RNNs model with sentiment attention networks to extract the influence of temporal sentiments on user preferences, to enhance SR. Cui et al. [8] designed a meta-learned sequential-knowledge-aware recommender model incorporating RNNs and GATs to extract sequential information from sequential knowledge graphs.

2.2 Transformer-based Methods

The transformer architecture can not only capture long-term semantics (like RNN) but it can also focus on relatively few high-impact actions through the attention mechanism, which makes it an outstanding architecture for sequential recommendations. Kang and McAuley [23] used Transformer instead of RNN to improve sequential recommendation performance. Sun et al. [45] adapted the famous BERT [9] architecture in natural language processing, replacing the single-directional attention in SASRec [23] to bi-directional attention architecture. Fan et al. [10] introduced distribution encoding into the representation of user's sequential preferences. Xie et al. [64] applied the contrastive learning method to the transformer encoders to enhance sequential recommendations accuracy. Ma et al. [33] improved the transformer-based sequential recommendation models with a preference editing-based self-supervised learning mechanism.

To improve the sequential recommendation performance, some methods exploit additional features, such as user profiles and item properties [19, 26]. DIN [76] and DIEN [74] adopt user profiles and context features along with the user behavior histories for sequential recommendations. Ren et al. [42] applied generative adversarial networks with transformers to process the item context information. Based on transformers, Li et al. [27] developed a time-interval aware self-attention model to encode time information into the sequential recommendation model. Liu et al. [29] encoded inter-sequence relations for the sequential recommendation.

2.3 Graph-based Methods

Graph-based models are another mainstream of sequential recommendations, especially in session recommendation scenarios. Instead of learning from the user-item bipartite graph, session graphs are constructed based on the user behaviors in certain sessions and apply graph-based models, e.g., GCN, GraphSage, and GAT, to extract graph features and generate recommendations [35, 40, 55, 58, 62, 68]. Recommendation methods based on session graphs usually involve two types of graphs: the graph built from the current session to learn item embeddings and the global user session graphs.

Extracting information from each user's session to predict the next user's behaviors in this session is a common solution in various session-based recommendation methods. For example, Wu et al. [62] designed the SRGNN model, which is a GNN model integrated with recurrent gated units, to predict the next preferred item for each user based on the current user's session graph. Qiu et al. [39] proposed FGNN to learn each item representation by aggregating its neighbors' embeddings through multi-head attention and obtain the final session representation by repeatedly combining each learnt embeddings with the relevance of each time to the session. DUVRec [67] utilizes the item view and factor view to encode both user preferences from the item-level as well as higher level through coarse-grained graphs. Moreover, SURGE [3] employs a metric learning method to automatically build graph structures for each sequence, and then uses interest-fusion graph convolutional layers and interest-extraction graph pooling layers to extract long-/short-term interest for sequential prediction. Huang et al. [20] applied position information as well as temporal information incorporating graph neural networks to make sequential recommendations

for user sessions. Zhang et al. [71] designed the kernel-enhanced transformer networks to fuse a substitutable product-product graph and a complementary product-product graph, to enhance the expressive ability of the GNN recommendation model.

Many recent methods utilize the global session graph to take advantage of the external context of user sessions. For example, GCE-GNN [58] utilizes the global-level and session-level item correlations to learn bi-level item representations, and aggregates the learnt item representations by a soft attention mechanism. DHCN [63] models all users' session data as a hypergraph, and designs a dual channel hypergraph convolutional network to process both the hypergraph and line graph data. GES [79] fuses both the global session graph and the semantic graph that records semantic item relations built upon item attributes, to predict the user behaviors in each session. FGNN [38] utilizes cross-session graph information to learn the representation of each individual session. These graph-based methods share some similar spirits with our proposed method. In our method, we also utilize the context information from the global graph to enhance the user behavior modeling. However, the major difference is that we construct multi-view graph representations for user behaviors, while processing each view with the multi-view transformers. This design enables higher-order message-passing along the user behavior sequence. In contrast, in existing GNN models, the higher-order information does not dynamically flow along the user behavior sequence, but mainly serves as a static global context to assist information extraction from local graphs.

2.4 Sequential Modeling with User Collaborative Information

In addition to modeling a user's past actions, a slew of methods incorporate supplementary user collaborative information to enhance the precision of sequential recommendations, aiming for improved accuracy in predicting the subsequent item. For instance, Reference [2] capitalizes on the category information of each item within the user's behavioral sequence, culminating in the proposal of CoCoRec (CategOry-aware COllaborative sequential Recommender).

Drawing from a congruent line of thought, Reference [48] centers its approach around discerning a user's latent intent for specific categories. This is achieved via a temporal convolutional network layer, which subsequently steers the sequential recommendation model to adeptly forecast the user's next item of preference. However, Reference [36] brings to the fore the **Intent-guided Collaborative Machine for Session-based Recommendation (ICM-SR)**. This mechanism encodes an active session by synergizing both the prior sequential items and the most recent one. The result is a coherent session representation, mirroring a user's intent. Such intent becomes instrumental in pinpointing the relevant neighboring sessions. The overall prediction then emerges as a fusion of insights from the current session and its neighboring counterparts. [78] introduced the **key-array memory network (KA-MemNN)**. This network is architecturally designed to hierarchically merge both a user's intention and preference, offering a more nuanced prediction of the ensuing item. GRU4REC [21] employs recurrent neural networks to encapsulate user behaviors within a session and predict the next item of that session. In contrast, our method efficiently adopts other user's behavior histories by our proposed multi-view GNN-transformer model to enhance the prediction of the current user.

2.5 Reinforcement-learning-based Methods

Sequential recommendation can be aptly framed as a sequential decision-making problem, capturing the nuances of user-system interaction more comprehensively than when approached as a mere classification or prediction challenge. This is why it is quite fitting to model the sequential recommendation using a **Markov decision process (MDP)** and consequently address it using **reinforcement learning (RL)** algorithms [1].

Echoing this perspective, there has been a notable surge in research works focusing on sequential recommendations harnessed by reinforcement learning. For instance, Reference [37] delves into an innovative reinforcement learning approach for sentiment-augmented knowledge graphs, pioneering the **Sentiment-Aware Policy Learning (SAPL)**, which guides recommendations based on RL techniques. Similarly, Reference [57] unveiled a distinctive RL framework that leverages both ontology-view and instance-view **Knowledge Graphs (KGs)** to capture multi-faceted user interests. Moreover, the **Knowledge-guidEd Reinforcement Learning model (KERL)**, as proposed by Reference [52], adeptly integrates KG information within an RL framework, marking a significant stride in sequential recommendation strategies.

3 PROPOSED RECOMMENDATION MODEL

3.1 Preliminaries

In this work, we denote the set of users by $\mathcal{U}$ and the set of items by $\mathcal{I}$. For each user u, we denote the sequence of the user's interacted items by $\mathcal{I}_u = [i_1^u, i_2^u, \ldots, i_{n_u}^u]$, where the items are sorted in chronological order based on the interaction timestamps, and n_u denotes the length of the item sequence $\mathcal{I}_u$. Moreover, we denote the set of all the observed user behavior sequences by $\mathcal{D}_s$. The main objective of a sequential recommendation model is to first predict the probability that a user u interacts with an item i, given the list of the user's historical items $\mathcal{I}_u$. Then, the candidate items are ranked based on the predicted dependency probabilities in descending order, and the N top-ranked items are recommended to the user. Table 1 summarizes the mostly used notations in this article.

With all the observed user behavior sequences $\mathcal{D}_s$, we first use a directed dependency graph $\mathcal{G}$ to describe the sequential dependency relationship between two items in the user behavior sequences. Specifically, we use $\mathbf{T}_{i,j}$ to denote item j's dependency on item i, and define it as follows:

$$\mathbf{T}_{i,j} = \frac{\sum_{u \in \mathcal{U}} \mathbb{I}[t_{u,i} < t_{u,j}]}{\sum_{u \in \mathcal{U}} \sum_{k \in \mathcal{I}} \mathbb{I}[t_{u,i} < t_{u,k}]}, \tag{1}$$

where $t_{u,i}$ is the timestamp when user u interacts with item i, and $\mathbb{I}(\cdot) = 1$ if the condition is true, and 0 otherwise. In Equation (1), a higher value of $\mathbf{T}_{i,j}$ indicates a higher chance that a user would like to interact with the item j, given that she has interacted with the item i. Then, if $\mathbf{T}_{i,j} > 0$, then we build an edge from item i to item j in $\mathcal{G}$. It is worth noting that $\mathbf{T}$ serves as the transition probability of any user clicking item j after she has clicked item i, and the item dependency (transition) matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ is an asymmetric row-stochastic matrix, i.e., the sum of each row is 1, where n is the number of items.

We further define the k-hop item transition matrix $\mathbf{T}^{(k)} = \prod_{i=1}^{k} \mathbf{T}$, and we denote the element in the ith row, jth column of $\mathbf{T}^{(k)}$ as $\mathbf{T}_{i,j}^{(k)}$, which also means the transition probability of any user to click item j after he/she has clicked item i kth steps ago. When k increases, the number of k-hop neighbours increases exponentially, and we may obtain a large amount of k-hop neighbours with low transition probability $\mathbf{T}_{i,j}^{(k)}$. Small transition probability indicates low dependencies, thus such neighbours are not informative enough for message passing, and they can be dropped during training to increase the model processing speed. To this end, in this work, we propose to choose the top h neighbours for each hop to increase the processing speed, while capturing enough information from the global context. Thus, to compute $\mathbf{T}^{(k)}$, we calculate by $\mathbf{T}^{(k)} = \sigma(\mathbf{T}^{(k-1)})\sigma(\mathbf{T})$, where $\sigma(\cdot)$ denotes the filter method to select the top h elements for each row.

Note that the k-hop transition matrix can be computed in the data prepossessing stage. As $\sigma(\mathbf{T}^{(k-1)})\sigma(\mathbf{T})$ is a sparse matrix multiplication, the complexity for each matrix multiplication can be bounded by $\min(O(n^{2+o(1)}), O(n^{2.38}))$[70].

Table 1. List of Notations

Notations	Description
$\mathcal{U}, \mathcal{I}$	Set of users and items
$\mathcal{I}_u$	Item sequence interacted by u
$\mathcal{D}_s$	Set of observed user behavior sequences
$\mathcal{G}$	Item-item correlation graph
$\mathbb{E}[\cdot]$	Expectation
L_u	Sequence length of user u
L	Maximum input length to the model
d	Dimension of latent representations
$\tau_{u,i}^{(k)}$	Transition probability from item i to item j at k steps
$r_{u,i}$	Interaction score of the $\{u, i\}$ pair
$\mathbf{u}_k$	kth view of user representations
θ	Neural network parameters
θ_k	Neural network parameters for the kth view
$\mathbf{T}^{(k)}$	k-hop item dependency (transition) matrix
$\mathbf{T}_{i,j}^{(k)}$	k-hop item dependency score from item i to item j
$\mathcal{N}_i^{(k)}$	k-hop neighbours of the item i
$\mathbf{e}_i$	Embedding of node i
$\{\alpha_k\}_{k=1}^K$	The set of hyper-parameters for the Dirichlet distribution
$\mathcal{D}(x\|\alpha)$	Dirichlet distribution for random variable x

3.2 Overall Structure

Figure 2 shows the overall architecture of the proposed sequential recommendation framework. For a given item sequence $\mathcal{I}_u$, rather than directly applying sequence encoding methods to obtain the sequence representation, we first extract sub-graphs from $\mathcal{G}$ to augment the sequence $\mathcal{I}_u$. For each item $i \in \mathcal{I}_u$, we extract the sub-graph of $\mathcal{G}$ containing all the 1 to K hop neighbors of i, and then generate different "views" of item i's neighborhood. The sub-graph is also the ego-graph with the node i as the central node, and each view of the ego-graph consists of the nodes that have the same distance to the central node. Specifically, the kth view only contains the k-hop neighbours of item i.

Figure 3 shows an example of how to build multi-views of the neighborhood of each item in a user behavior sequence. One challenge in searching for k-hop neighbors is that the number of such neighbors grows exponentially as k increases. This results in a more complex model as k gets larger. To address this issue, we have developed the Dirichlet weight sampling method, which allows for the selection of important k-hop neighbors while avoiding overfitting on the selected neighbors. Details will be elaborated in Section 3.2.2.

Then, for the kth view, we use an efficient graph aggregation method to extract the local graph information of each item and apply a Transformer [45] module to capture the sequential information among all the items in the sequence $\mathcal{I}_u$ and obtain the sequence embedding from the k views. In this way, the sequence embedding can capture both the sequential behavior patterns of a specific user and the collaborative information among different users through the item dependency sub-graphs.

In the following step, we aggregate multiple views through attention network to generate a unified representation of the user behavior sequence for predicting the next potential interaction

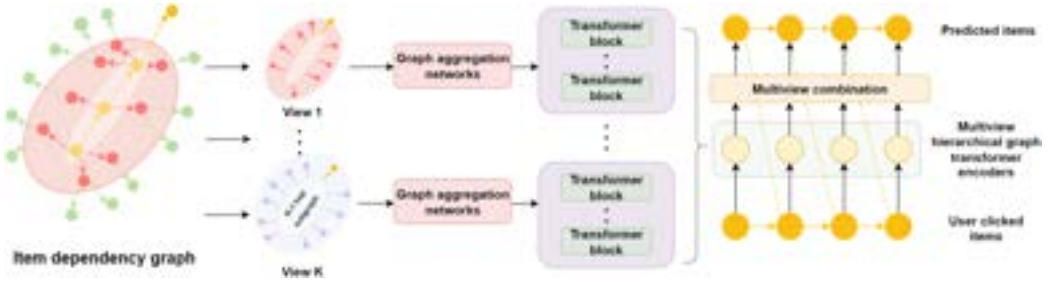

Fig. 2. Architecture of the proposed framework. The input contains the user-clicked item sequence together with the sub-graphs of the items from the item dependency graph. We form multiple views for the input item sequence, and each view is passed through the hierarchical graph aggregation networks followed by the transformer encoders. Finally, we combine the representations of the user preference from multiple views to predict the user's next preferred item.

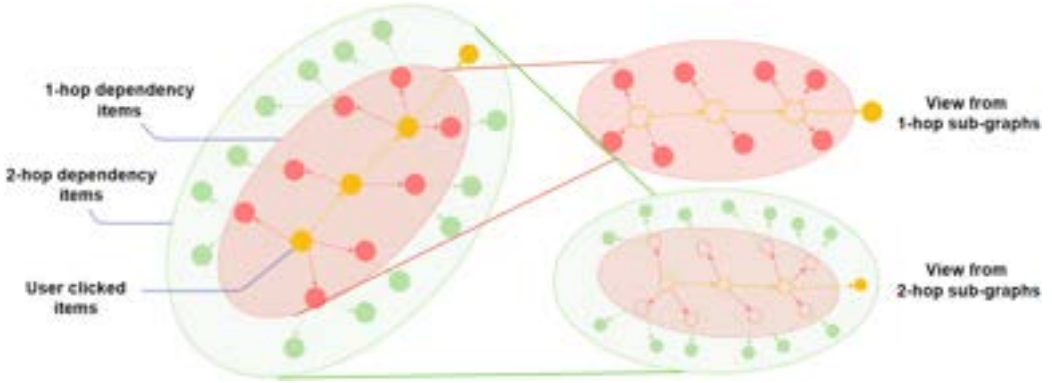

Fig. 3. Item dependency graph, where the yellow-colored trace represents the sequence of the user's clicked items. The red-colored nodes (i.e., 1-hop neighbours) represent the items that have strong dependencies with the user-clicked items, which form the view of the first-hop neighbours. The green-colored nodes (i.e., 2-hop neighbours) represent the items that have strong dependencies with the red-colored nodes, which form the view of the second-hop neighbours.

items. Embeddings from different views are further aggregated using an attention network to generate a unified representation of the user behavior sequence for predicting the next potential interaction items.

3.3 Graph Aggregation Networks

After extracting the local sub-graph for each item in the item sequence, we aim to obtain an embedding for each item by aggregating its local dependency sub-graph information. One potential solution is to use existing GNNs, e.g., **Graph Convolutional Networks (GCNs)** [24] and **Graph Attention Networks (GATs)** [51], for message passing in the local sub-graph of each item. However, these existing GNNs are not preferred to be used in the proposed framework. Because they need to process the whole graph by adding convolutional layers or attention layers to each node of the whole graph, and obtain the next-hop embedding iteratively. Such a learning procedure is usually time-consuming (refer to Section 3.7 for more discussions), and could hardly process a large amount of frequently changing sub-graphs efficiently.

3.3.1 Intra-hop Aggregation. To improve the efficiency of message passing in the local sub-graphs of items in $\mathcal{I}_u$, we propose the following intra-hop aggregation procedure. For each item $i \in \mathcal{I}_u$, we treat it as the central node and denote its k-hop neighbors in $\mathcal{G}$ by $\mathcal{N}_i^{(k)}$. Then, we perform the kth intra-hop aggregation via

$$\mathbf{e}_i^{(k)} = \sum_{j \in \mathcal{N}_i^{(k)}} \mathbf{T}_{i,j}^{(k)} \mathbf{e}_j, \tag{2}$$

where $\mathbf{e}_j$ is the initial embedding of node j. Based on Equation (2), the $(k+1)$th hop aggregation can be computed as follows:

$$\begin{aligned} \mathbf{e}_i^{(k+1)} &= \sum_{j \in \mathcal{N}_i^{(k)}} \mathbf{T}_{i,j}^{(k)} \sum_{m \in \mathcal{N}_j^{(k+1)}} \mathbf{T}_{j,m}^{(1)} \mathbf{e}_m \\ &= \sum_{j \in \mathcal{I}} \mathbf{T}_{i,j}^{(k)} \sum_{m \in \mathcal{I}} \mathbf{T}_{j,m}^{(1)} \mathbf{e}_m = \sum_{m \in \mathcal{N}_i^{(k+1)}} \mathbf{T}_{i,m}^{(k+1)} \mathbf{e}_m, \end{aligned} \tag{3}$$

which corresponds to Equation (2) by increasing k to $k+1$. Equation (3) indicates that to perform the kth hop aggregation for $k \in \{1, 2, \ldots, K\}$, we can neglect the 1-hop to $(k-1)$th hop sub-graphs and directly aggregate the kth hop nodes using their dependency scores as the weights.

Compared with existing GCN-based models that concatenate k layers to process the kth hop sub-graphs, the proposed aggregation method can directly process the specific hop of the sub-graph. Moreover, the proposed aggregation method is more suitable for the sequential recommendation task. Because the item embeddings may be frequently changed in each batch of training samples. In contrast, GCNs need to re-process the kth hop embeddings of the entire item graph for each batch. Using GCN-based methods may significantly increase the computation time.

3.3.2 Dirichlet Weight Sampling. Since $\mathbf{T}_{i,j}^{(k)}$ is an empirical estimation of the overall dependency between items. For a specific user, such dependency may not reflect the exact relationships between items in the user-specific behavior sequence. Especially, when the interaction data is relatively sparse, the item dependency might be biased due to the shortage of data.

Similar issues have been reported in many related works such as SSEPT [61] and SGCN [4], where the stochastic masking is applied to the input data to avoid over-fitting due to data sparsity. Intuitively, the model should not over-fit on specific $\mathbf{T}_{i,j}^{(k)}$ on top h neighbours. To mitigate the inductive bias of the model and avoid over-fitting on transition pattern on the top h neighbours, we propose the Dirichlet sampling method to stochastically sample the transition probability, i.e., the weight parameters $\mathbf{T}_{i,j}^{(k)}$.

For a set of Dirichlet random variable $\{\mathbf{x}_1, \ldots, \mathbf{x}_K\} \sim \mathcal{D}(\mathbf{x}|\alpha)$, where α represents a set of hyperparameters $\{\alpha_k\}_{k=1}^K$, has the following probability density function,

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_K; \alpha_1, \ldots, \alpha_K) = \frac{1}{\mathrm{B}(\alpha)} \prod_{k=1}^{K} \mathbf{x}_i^{\alpha_k - 1}, \tag{4}$$

where $\mathrm{B}(\cdot)$ denotes the Beta function and $\mathbf{x}_k$ denotes the Dirichlet random variable of the kth hop neighbours. The expectation of the random variable $\{\mathbf{x}\}_{k=1}^K \sim \mathcal{D}(\mathbf{x}|\alpha)$ is $\mathbb{E}[\mathbf{x}_k] = \frac{\alpha_k}{\sum_k \alpha_k}$. Furthermore, large hyper-parameters $\{\alpha\}_{k=1}^K$ imply that $\{\mathbf{x}\}_{k=1}^K$ are concentrated around their expected value, while small $\{\alpha\}_{k=1}^K$ give high variance to $\{\mathbf{x}\}_{k=1}^K$ to make them less concentrated.

To be specific, based on the dependency score $\mathbf{T}_{i,j}^{(k)}$, we can construct the Dirichlet distribution so that the expectation of the random variable $x_{(i,j,k)}$ that follows a Dirichlet distribution equals

to $\mathbf{T}_{i,j}^{(k)}$, i.e., $\mathbb{E}[x_{(i,j,k)}] = \mathbf{T}_{i,j}^{(k)}$. This Dirichlet sampling is unbiased according to $\mathbf{T}_{i,j}^{(k)}$. Finally, we use $x_{(i,j,k)}$ to replace $\mathbf{T}_{i,j}^{(k)}$ to represent the kth hop transition probability from item i to item j in practice.

The advantage of using Dirichlet sampling to sample multinomial weight distribution rather than using stochastic masking is that Dirichlet sampling can provide randomness to the transition probability while maintaining unbiased sampling. In contrast, stochastic masking does not change the transition probability but only provides binary selection on neighbours.

3.4 Sequence Embedding via Transformers

For each view, we first obtain the sub-graph representation of each item in the sequence by the graph aggregation model. Then, we apply a set of transformer blocks on the item sequence to obtain the representation of the item sequence at the kth view. For the item sequence whose length $n_u < L$, where L is the maximum length among all the sequences, we pad zero values at the beginning of the sequence to make the lengths of all the sequences to be L.

3.4.1 Positional Encoding. We apply learnable positional embedding $\mathbf{P} \in \mathbb{R}^{L\times d}$ referred to the previous work [23, 45], and the user's item sequence embedding at kth view $E_{u,k}$ is performed by an element-wise addition of item embedding and positional embedding,

$$\mathbf{E}_{u,k} = \begin{bmatrix} \mathbf{e}_{s_1}^{(k)} + \mathbf{p}_1 \\ \mathbf{e}_{s_2}^{(k)} + \mathbf{p}_2 \\ \cdots \\ \mathbf{e}_{s_L}^{(k)} + \mathbf{p}_L \end{bmatrix}. \tag{5}$$

3.4.2 Multi-head Self-attention Blocks. We stack multiple self-attention blocks based on the embedding layer to capture the sequential feature of the user $u \in \mathcal{U}$. The conventional scaled dot-product attention for the kth view is defined as $\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}$, where $\mathbf{Q} = \mathbf{E}_u\mathbf{W}^Q$ represents the query, $\mathbf{K} = \mathbf{E}_u\mathbf{W}^K$ is the key, $\mathbf{V} = \mathbf{E}_u\mathbf{W}^V$ is the value and d is the latent dimension acting as the scaled factor to regularize the dot-product value. The learnable projection matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d\times d}$ are used to project the item sequence embedding to $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. In this work, we design the multi-view multi-head self-attention blocks, which provide multi-head attention for each view independently. Each self-attention block for the kth view contains multiple attention heads, and the formula is defined as follows:

$$\begin{aligned} \text{MultiHeadAtt}\big(\mathbf{E}_{u,k}^l\big) &= [\text{head}_{1,k}, \text{head}_{2,k}, \ldots, \text{head}_{h,k}]W_k^H, \\ \text{head}_{i,k} &= \text{Attention}(\mathbf{Q}_{i,k}, \mathbf{K}_{i,k}, \mathbf{V}_{i,k}), \end{aligned} \tag{6}$$

where $\mathbf{Q}_{i,k} = \mathbf{E}_{u,k}^l\mathbf{W}_{i,k}^Q$, $\mathbf{K}_{i,k} = \mathbf{E}_{u,k}^l\mathbf{W}_{i,k}^K$, and $\mathbf{V}_{i,k} = \mathbf{E}_{u,k}^l\mathbf{W}_{i,k}^V$ are the query, key, and value, respectively. $\mathbf{W}_{i,k}^Q, \mathbf{W}_{i,k}^K, \mathbf{W}_{i,k}^V \in \mathbb{R}^{d\times d/h}$. $\mathbf{E}_{u,k}^l \in \mathbb{R}^{L\times d}$ is the input to the kth view, lth self-attention block. $\mathbf{W}_k^H \in \mathbb{R}^{d\times d}$ is the kth view projection matrix to obtain the output $\mathbf{E}_{u,k}^{l+1}$, which can be the input to the next self-attention block. The output is then processed by the feed-forward layer.

Feed-forward Layer. The feed-forward layer provides non-linearity to the transformer and supports interaction between dimensions. It consists of two affine transformations with a **Gaussian Error Linear Unit (GELU)** in between, to process the input data x_k from the kth view as follows:

$$\text{FFN}(\mathbf{x}_k) = \text{GELU}(\mathbf{x}_k\mathbf{W}_{1,k} + \mathbf{b}_{1,k})\mathbf{W}_{2,k} + \mathbf{b}_{2,k}, \tag{7}$$

where $\mathbf{W}_{1,k}$, $\mathbf{W}_{2,k}$, $\mathbf{b}_{1,k}$, and $\mathbf{b}_{2,k}$ are learnable parameters for the kth view, which are not shared across layers.

3.5 Multi-view Aggregation

After performing the aforementioned steps, for each view $k \in \{0, 1, \ldots, K\}$, where $k = 0$ denotes the view from the input item sequence without item-dependency graphs, we obtain an embedding $\mathbf{u}_k$, which is the output from the FFN layer, for a user sequence $\mathcal{I}_u$. In the following, we further merge the embeddings from all the $(K + 1)$ views to generate an overall representation for each specific user to capture his/her preference and context-aware intention. The aggregation is done by using the attention mechanism via $\hat{\mathbf{u}} = \sum_{k=0}^{K} \frac{\exp(\mathbf{h}_k^\top \mathbf{u}_k)\mathbf{u}_k}{\sum_{k=0}^{K} \exp(\mathbf{h}_k^\top \mathbf{u}_k)}$, where $\hat{\mathbf{u}}$ denotes the overall embedding for user u based on the set of $(K+1)$ views, and $\mathbf{h}_k$ is a learnable parameter. It is worth noting that u_k is the output of the last FFN(x_k) layer. With the user embedding, a predicted interaction score of user u and item i is computed as, $\hat{r}_{u,i} = \hat{\mathbf{u}}\mathbf{e}_i^\top$, where $\mathbf{e}_i$ is the embedding of item i.

3.6 Loss Functions

We use $\mathcal{D}_s$ to represent the training set of a recommender system, where $\mathcal{D}_s := \{(r_{u,i}, u, i) | u \in \mathcal{U}, i \in \mathcal{I}\}$, and $r_{u,i}$ is the interaction score of the (u, i) pair. As our goal is to correctly predict the potential user-item interactions, the main objective of the proposed model can be formulated as follows:

$$\min_{\theta} \sum_{(u,i)\in\mathcal{D}_s} \ell_{\text{main}}(r_{u,i}, \hat{r}_{u,i}), \tag{8}$$

where $r_{u,i}$ is the ground-truth rating of item i given by user u, and $\hat{r}_{u,i}$ is the predicted rating. We denote $\hat{r}_{u,i} = f_\theta(\mathcal{I}_u, i)$, where f_θ is our proposed model with a set of learnable parameters θ, and $\mathcal{I}_u$ is the item sequence of user u. We adopt the binary-cross-entropy loss for ℓ_{main}, that is

$$\ell_{\text{main}} = \sum_{(u,i)\in\mathcal{D}_s} \big[- r_{u,i}\log(\sigma(f_\theta(\mathcal{I}_u, i))) - (1 - r_{u,i})\log(1 - \sigma(f_\theta(\mathcal{I}_u, i)))\big]. \tag{9}$$

Besides the main loss in Equation (9), we also aim to make use of multiple view information to construct auxiliary losses to provide more discriminative information into our model learning. As we obtain a representation of a user for each view before performing multi-view aggregation in Section 3.5, we can construct an individual prediction model for each view $k \in \{0, 1, \ldots, K\}$, denoted by $f_{\theta_k}(\mathcal{I}_u, i)$, where $\theta_k \subseteq \theta$ is the subset of parameters for view k. $\sigma(\cdot)$ denotes the activation function, and we use the Sigmoid function similar as SASRec [23].

Thus, the final prediction of $\hat{r}_{u,i}$ can be further decomposed into a convex combination of the predicted results generated from each view: $\hat{r}_{u,i} = \sum_{k=0}^{K} w_k f_{\theta_k}(\mathcal{I}_u, i)$. Here, $w_k \in [0, 1]$ represents the attention weight of kth view that can be obtained by the attention mechanism as elaborated in Section 3.5, and $f_{\theta_k}(\mathcal{I}_u, i)$ denotes prediction score with the kth view's transformer model. Therefore, the binary-cross-entropy loss can be re-written as follows:

$$\ell_{\text{main}} = \sum_{(u,i)\in\mathcal{D}_s} \left[-r_{u,i}\log\left(\sigma\left(\sum_k w_k f_{\theta_k}(\mathcal{I}_u, i)\right)\right) - (1 - r_{u,i})\log\left(1 - \sigma\left(\sum_k w_k f_{\theta_k}(\mathcal{I}_u, i)\right)\right)\right]. \tag{10}$$

However, the above loss function causes biased updates for each individual model: the lower the attention weight, the lesser the individual view's model f_{θ_k} is optimized. To solve this issue, we propose an auxiliary loss ℓ_{indiv} that focuses on updating the network parameters in each individual view, shown as follows:

$$\ell_{\text{indiv}} = \sum_k \sum_{(u,i)\in\mathcal{D}_s} \left[-r_{u,i}\log(\sigma(f_{\theta_k}(\mathcal{I}_u, i))) - (1 - r_{u,i})\log(1 - \sigma(f_{\theta_k}(\mathcal{I}_u, i)))\right]. \tag{11}$$

3.6.1 Mutual Information Maximization. Note that the auxiliary loss function in Equation (11) can be optimized by minimizing each individual loss independently. However, minimizing Equation (11) or Equation (9) does not guarantee the consensus of each view. Thus, we encode the consensus constraint into the predictions of different individual views on the same user-item pair, by introducing another objective to maximize the mutual information between different views. We propose to maximize the mutual information among the multiple views as follows:

$$\max_{\theta} \mathbb{E}_{\mathbf{u}_k} I_\theta(\mathbf{u}_k; \mathbf{U} - \mathbf{u}_k), \tag{12}$$

where $\mathbf{u}_k$ is the representations from the kth view, $k \in \{0, \ldots, K\}$. $\mathbf{U} - \mathbf{u}_k = \{\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_K\}$ denotes the representations from all views except $\mathbf{u}_k$, and $I_\theta(\mathbf{u}_k; \mathbf{U} - \mathbf{u}_k) = \sum_{\mathbf{u}_k} p_\theta(\mathbf{u}_0, \ldots, \mathbf{u}_K) \log \frac{p_\theta(\mathbf{u}_k | \mathbf{U} - \mathbf{u}_k)}{p_\theta(\mathbf{u}_k)}$ computes the mutual information between the view $\mathbf{u}_k$ and other views. By maximizing Equation (12), we expect that the multiple views will converge to a consistent representation of the user's preference. However, the mutual information is difficult to compute. As $I_\theta(\mathbf{u}_k, \mathbf{U} - \mathbf{u}_k) \propto \frac{p(\mathbf{u}_k | \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_k)}$, following Reference [34], we minimize the noise contrastive loss as follows:

$$\ell_{\text{contrast}} = -\frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}_{\mathbf{u}_k} \log \frac{f(\mathbf{u}_k, \mathbf{U} - \mathbf{u}_k)}{f(\mathbf{u}_k, \mathbf{U} - \mathbf{u}_k) + \sum_{\mathbf{u}_{neg}} f(\mathbf{u}_{neg}, \mathbf{U} - \mathbf{u}_k)}, \tag{13}$$

where $\mathbf{u}_{neg}$ is the sampled negative view's representations from the batch. $f(\mathbf{u}_k, \mathbf{U} - \mathbf{u}_k) = \text{Sigmoid}(\langle \mathbf{u}_k, \frac{\sum_{j \neq k} \mathbf{u}_j}{K} \rangle)$ denotes the function used to calculate the similarity of the view $\mathbf{u}_k$ to other views, and $\langle \cdot, \cdot \rangle$ denotes the inner product. In our methodology, negative views are derived from the behavior sequences of users within the same batch, excluding the target user. For each user, the total number of negative samples is $(K+1) \times (b-1)$, where b is the batch size in training. Since the number of total negative samples is relatively large, in practice, for each user in each view, we randomly sample b negative samples from the negative sample set with size $(K+1) \times (b-1)$.

Proposition 1. *Suppose $\ell_{contrast}$ is driven in Equation (13) and let N be the total number of training samples from the training set, the following inequality holds:*

$$\ell_{contrast} \geq -\mathbb{E}_{\mathbf{u}_k} I_\theta(\mathbf{u}_k; \boldsymbol{U} - \mathbf{u}_k) + log(N). \tag{14}$$

Proof. Since the optimal value of $I(\mathbf{u}_k, \mathbf{U} - \mathbf{u}_k)$ is given by $\frac{p(\mathbf{u}_k | \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_k)}$, due to the proportional property as we discussed in Section 3.6.1, we can insert the ratio back to Equation (13) and obtain

$$\begin{aligned}
\ell_{\text{contrast}} &= -\frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}_{\mathbf{u}_k} \log \left[\frac{\frac{p(\mathbf{u}_k | \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_k)}}{\frac{p(\mathbf{u}_k | \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_k)} + \sum_{\mathbf{u}_{\text{neg}}} \frac{p(\mathbf{u}_{\text{neg}} | \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_{\text{neg}})}} \right] \\
&= \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}_{\mathbf{u}_k} \log \left[1 + \frac{p(\mathbf{u}_k)}{p(\mathbf{u}_k \mid \mathbf{U} - \mathbf{u}_k)} \sum_{\mathbf{u}_{\text{neg}}} \frac{p(\mathbf{u}_{\text{neg}} \mid \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_{\text{neg}})} \right] \\
&\approx \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}_{\mathbf{u}_k} \log \left[1 + \frac{p(\mathbf{u}_k)}{p(\mathbf{u}_k \mid \mathbf{U} - \mathbf{u}_k)} (N-1) \mathop{\mathbb{E}}_{\mathbf{u}_{\text{neg}}} \frac{p(\mathbf{u}_{\text{neg}} \mid \mathbf{U} - \mathbf{u}_k)}{p(\mathbf{u}_{\text{neg}})} \right] \\
&= \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}_{\mathbf{u}_k} \log \left[1 + \frac{p(\mathbf{u}_k)}{p(\mathbf{u}_k \mid \mathbf{U} - \mathbf{u}_k)} (N-1) \right]
\end{aligned}$$

$$\geq \frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}_{\mathbf{u}_k}\log\left[\frac{p(\mathbf{u}_k)}{p(\mathbf{u}_k \mid \mathbf{U}-\mathbf{u}_k)}N\right] \tag{15}$$
$$= -\mathbb{E}_{\mathbf{u}_k} I(\mathbf{u}_i; \mathbf{U}-\mathbf{u}_k) + \log(N).$$

This proof partially refers to the ideas from the literature [34]. It shows that minimizing Equation (13) results in maximizing the lower bound of the mutual information of each view $\mathbf{u}_k$ to the other views $\{\mathbf{U}-\mathbf{u}_k\}$, $k \in \{0, \ldots, K\}$. Note that a pairwise multi-view contrastive learning model was proposed in Reference [49]. However, the time complexity of the pairwise contrastive model grows exponentially with the number of views, which is not practical when the number of views is large. In contrast, the complexity of Equation (13) is linear with the number of views. □

Overall loss. To sum up, we combine the main loss function ℓ_{main} in Equation (9), the sum of individual losses from different views ℓ_{indiv} in Equation (11), and the contrastive loss ℓ_{contrast} in Equation (13) to construct an overall loss to train the proposed model,

$$\ell_{\text{combine}} = \ell_{\text{main}} + \lambda_1 \ell_{\text{indiv}} + \lambda_2 \ell_{\text{contrast}}, \tag{16}$$

where λ_1 and λ_2 are the hyper-parameters used to control the weights of the regularization components.

3.7 Time Complexity Discussion

We compare the training time complexity of the proposed model with existing GNN-based models for sequential recommendation. To compare the complexity under the sequential recommendation setting, we apply GNN models to process the user-item graph, obtain the embeddings of items, and input the embeddings into the transformer model. Suppose we have a graph $\mathcal{G}$ with n nodes and e edges in total. We train the model with batch size s. The number of training samples is M. The number of neighbours being sampled for each node is h. The number of GNN layers is K. The time complexity for a fixed-length transformer is constant T. The dimension of embedding size is d. Thus, the time complexity of the proposed model in each epoch is $O(hKdM + MKT)$, where hK represents the intra-hop aggregation. The proposed model's time complexity linearly increases with the expansion of the number of sampled neighbours h, and the sub-graph size K.

To show the time complexity comparison of our model with the state-of-the-art graph-based models, such as GCN [24], GraphSage [13], FastGCN [5], and Cluster-GCN [6], in the same sequential recommendation settings, we take the aforementioned models as the graph-backbone. For each user behavior, the graph-backbone takes the user-item interaction graph as the input and outputs the embedding of that behavior. Then, we input the sequence of behavior embeddings into a transformer model to predict the next behavior of that user. In Table 2, we show the comparison of the training complexity of our model with the aforementioned models. Moreover, $K \ll h \ll n \ll m$, and $b \ll n$ are in common recommendation scenarios. Therefore, compared to the GCN+SR (SR stands for sequential recommendation with transformers), GraphSage+SR, FastGCN+SR and Cluster-GCN+SR, our model performs significantly faster. Furthermore, we record the time spent of the aforementioned methods for training one batch of data from the ML-1M dataset in Table 2, with the batch size of 256 and the hidden dimension of 128 on a single 1080Ti GPU. It is worth noting that our model achieves 6.2× and 8.1× faster compared to FastGCN + transformers and GCN + transformers, respectively.

The reduction in time complexity is primarily attributed to the use of our proposed multi-hop graph aggregation networks. The graph aggregation is primarily achieved through multi-hop aggregation, wherein the main computation involves finding k-hop neighbors, which can be performed during the pre-processing phase. Additionally, employing sparse multi-hop aggregation

Table 2. Time Comparison of Training Each Epoch for GNN+SR (Sequential Recommendation) Training Algorithms

Model	Time Complexity	Time/batch
GCN[24] with sequential recommendation	$O\left(\frac{(Kmd+Knd^2)M}{b}+MT\right)$	0.5347 s
GraphSage [13] with sequential recommendation	$O\left(\frac{(h^K nd^2)M}{b}+MT\right)$	23.4390 s
FastGCN [5] with sequential recommendation	$O\left(\frac{(Khnd^2)M}{b}+MT\right)$	0.4126 s
Cluster-GCN [6] with sequential recommendation	$O\left(\frac{(Kmd+Knd^2)M}{b}+MT\right)$	0.5119 s
Ours	$O(hKdM+MKT)$	0.0659 s

n is the total number of nodes. M is the total number of training samples. m is the total number of edges. K is the number of layers. b is the batch size. h is the number of neighbors being sampled for each node. For simplicity, the dimensions of the node hidden features remain constant, denoted by d. The time complexity for processing each fixed length sequence by the transformer is constant, denoted by T.

can boost performance speed for rapidly changing graphs, where the embedding of each graph node is updated after every training step.

Moreover, the utilization of the Dirichlet sampling method does not increase the complexity of the training process, as it primarily influences the sampling weights of each k-hop neighbor and does not add computation cost to the model. In comparison to other baseline methods, our multi-hop graph aggregation networks can be easily integrated into other backbone models, which we will discuss in detail in Section 5.1.5.

Overall, our proposed model boasts numerous advantages, such as reduced time complexity, increased performance speed, adaptability to various backbone models, and the potential for enhanced recommendation accuracy. The incorporation of the Dirichlet sampling method enables our approach to strike a balance between exploration and exploitation during the training process, resulting in more stable robustness. The combination of these benefits, along with the potential for wider applications across different domains, establishes our proposed method as a promising solution for recommendation systems. In the following section, we will elaborate on extensive experiments to further evaluate our model's performance and showcase the effectiveness of our proposed approach.

4 EXPERIMENTS

In this section, we present comprehensive experiments to showcase the advantages and effectiveness of our model and its individual components. We assess our model using five publicly available datasets: MovieLens-1M,[1] Yelp,[2] Amazon Video Games,[3] Amazon CDs,[4] and Tmall.[5] MovieLens datasets serve as widely recognized benchmarks for recommendation systems, containing movie reviews from a prominent movie review website. The Amazon Video Games and Amazon CDs datasets comprise user reviews for products in the video game and CD categories on Amazon, respectively. Meanwhile, the Yelp dataset features user ratings for various businesses on Yelp. Last, the Tmall dataset includes user feedback from the e-commerce platforms Tmall and Taobao. Table 3 provides detailed statistics for each dataset.

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://www.yelp.com/dataset
[3] https://jmcauley.ucsd.edu/data/amazon/
[4] https://jmcauley.ucsd.edu/data/amazon/
[5] https://tianchi.aliyun.com/dataset/dataDetail?dataId=53

Table 3. Statistics of the Experimented Data

Dataset	User #	Item #	Interaction #	Avg. actions/user #
Yelp	25,677	25,815	0.7M	26.5
Movielens-1M	6,040	3,707	1.0M	163.5
Video Game	24,303	10,674	207k	8.51
CD	75,258	64,444	1.0M	13.3
Tmall	48,618	40,729	335k	6.9

4.1 Evaluation Metrics

To evaluate the performance of the recommendation models, we adopt the commonly used leave-one-out evaluation, i.e., next-item prediction task [15, 23]. We split the dataset into train, validation and test set: for each user, we take the last item of the behavior sequence as the test set, the second-last as the validation set, and the rest as the train set. We evaluate all the methods in terms of ***Hit Ratio*** **(HR)** and ***Normalized Discounted Cumulative Gain*** **(NDCG)**. HR@N measures the average number of positive items being retrieved in the generated top-N recommendation list for each user. NDCG@N extends HR@N by considering the positions of retrieved positive items in the top-N recommendation list. A higher value in the metric reflects a higher performance.

In this work, we opt for the full-ranking strategy over the sampling-based ranking strategy when evaluating recommendation performance, as suggested by Reference [25]. Specifically, for each user, we sort all the candidate items in descending order based on the predicted scores and choose N top-ranked items as the top-N recommendation list. In the experiments, we empirically set N to 5, 10, and 20. As we only have one ground truth item for each user in the testing data, HR@N is equivalent to Recall@N and proportional to Precision@N.

4.2 Baseline Methods

To show the effectiveness of our method, we include two groups of recommendation baselines[6]: graph-based recommendation methods, including:

(a) Light-GCN [14] uses Graph Convolutional Networks to learn users' preferences on items from the bipartite user-item interaction graph.
(h) DHCN [63] designs Hypergraph Convolutional Networks to capture both session graph and global graph information.
(i) SUGER [3] proposes Interest-fusion Graph Convolutional Layers and Interest-extraction Graph Pooling Layers to extract long-/short-term interest for sequential prediction.
(j) GES [79] conducts graph convolution on the hybrid item graph to generate smoothed item embeddings.
(l) SRJGraph [72] proposes a GNN-based CTR model to apply to both search and recommendation scenarios.
(m) SAPL [37] designs a reinforcement learning strategy for learning users' sentiments on items to correct recommendations.
(n) ReMR [57] adopts a reinforcement learning framework for multi-level recommendation reasoning.
(r) KERL [52] fuses knowledge graph information into a RL framework for sequential recommendation.

[6]The serial numbers of the methods correspond to Table 4.

We also adopt the following well-known sequential recommendation methods, which are not based on graph architectures:

(b) Caser [46] employs CNN in both horizontal and vertical ways to model high-order Markov chains of users' behaviors.
(c) SASRec [23] employs a transformer model to capture the user's sequential behavior and predict the next item for the recommendation.
(d) GRU4REC [17] adopts Recurrent Neural Networks with multiple GRU layers to sequentially predict users' next behaviors.
(e) HGN [32] hierarchically combines gating networks to capture user's intentions based on the user's features and sequential behaviors.
(f) Bert4Rec [45] is based on SARec [23] architecture. It uses the BERT [9] architecture instead of transformers to extract user's intentions and conduct sequential predictions.
(g) STOSA [10] uses the Wasserstein self-attention networks to capture users' behavior patterns from their sequential behavior histories. It is one of the state-of-the-art non-graph-based sequential recommendation methods.
(k) DIEN [75] designs the interest evolution network for CTR prediction.
(o) RKSA [22] is a sequential recommendation model that uses relation-aware kernelized self-attention to enhance the prediction.
(p) MT4SR [10] adopts a multi-relational transformer to capture the auxiliary item relationships in sequential recommendations.
(q) DFAR [28] uses a dual-interest disentangling method to factorize the relation between different types of feedback and decouple positive and negative interests before performing disentanglement on their representations.

4.3 Experiment Settings

For a fair comparison, we adopt the code provided by the corresponding authors or implement the method if the code is not provided. All other hyper-parameters and initialization strategies are according to the suggestions of the methods' authors. We also tune the parameters to make the baseline methods perform well on different datasets using the validation set. We implement our model using PyTorch and train the model using Adam optimizer with a learning rate of 0.001. We set the maximum sequence length $L = 100$ for all datasets. We choose graph size from $\{1, 2, 3\}$, the number of transformer layers from $\{1, 2, 3\}$ and the number of attention heads from $\{1, 2, 4\}$. We vary the embedding size in $\{32, 64, 128\}$. We further conduct paired per-user significance tests according to the method in the literature [44], verifying that all the improvements are statistically significant for $p < 0.001$. All models are trained on the NVIDIA GeForce GTX 1080 Ti GPU.

5 RESULTS AND ANALYSIS

The experiment results for all the methods on the five datasets with HR@N and NDCG@N are shown in Tables 4, 5, 6, 7, and 8, respectively. From the results, we can infer that, in general, our sequential recommendation method can outperform the state-of-the-art sequential recommendation methods. The results show that our method outperforms all the baselines in various datasets. The performance of our method largely improves on datasets ML-1M and Yelp. On Yelp, our model achieved 0.0738 on HR@20, compared to SOTA graph-based sequential recommendation models such as SURGE, which is 0.0692 on HR@20 and GES, which is 0.0596 on HR@20, our model achieves 6.7% and 23.8% improvements. Since ML-1M and Yelp datasets are based on user's reviews, the connections between items are more correlated and, therefore, a large amount of information can be stored in the item-to-item connections. Therefore, applying items' sub-graphs can help

Table 4. Recommendation Performance Achieved by Different Methods in Terms of HR and NDCG on the Yelp Dataset

Methods	Yelp					
	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
(a) Light-GCN	0.0191	0.0118	0.0395	0.0174	0.0586	0.0225
(b) Caser	0.0185	0.0110	0.0361	0.0178	0.0597	0.0239
(c) SASRec	0.0224	0.0138	0.0425	0.0214	0.0689	0.0281
(d) GRU	0.0196	0.0118	0.0343	0.0165	0.0558	0.0220
(e) HGN	0.0231	0.0147	0.0401	0.0198	0.0652	0.0261
(f) BERT4Rec	0.0198	0.0113	0.0387	0.0189	0.0599	0.0241
(g) STOSA	0.0223	0.0138	0.0414	0.0204	0.0657	0.0265
(h) DHCN	0.0179	0.0112	0.0319	0.0155	0.0532	0.0208
(i) SURGE	0.0239	0.0113	0.0429	0.0219	0.0692	0.0288
(j) GES	0.0187	0.1106	0.0343	0.0203	0.0596	0.0246
(k) DIEN	0.0205	0.0107	0.0371	0.0187	0.0674	0.0268
(l) SRJGraph	0.0204	0.0131	0.0421	0.0208	0.0675	0.0277
(m) SAPL	0.0257	0.0161	0.0446	0.0217	0.0716	0.0287
(n) ReMR	0.0248	0.0156	0.0437	0.0215	0.0711	0.0283
(o) RKSA	0.0231	0.0147	0.0438	0.0218	0.0694	0.0282
(p) MT4SR	0.0258	0.0165	0.0440	0.0221	0.0723	0.0284
(q) DFAR	0.0229	0.0145	0.0429	0.0213	0.0691	0.0276
(r) KERL	0.0224	0.0138	0.0424	0.0210	0.0692	0.0279
(s) Ours	**0.0276**	**0.0174**	**0.0453**	**0.0230**	**0.0738**	**0.0299**

The best results are in **boldface**, and the second-best results are underlined. The improvements achieved by our model over baseline methods are significant with p-value smaller than 0.001.

Table 5. Recommendation Performance Achieved by Different Methods in Terms of HR and NDCG on the ML-1M Dataset

Methods	ML-1M					
	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
(a) Light-GCN	0.1145	0.0695	0.1610	0.0857	0.2631	0.1146
(b) Caser	0.1272	0.0808	0.2017	0.1048	0.2930	0.1261
(c) SASRec	0.1288	0.0805	0.1877	0.0950	0.2611	0.1153
(d) GRU	0.0984	0.0635	0.1522	0.0804	0.2341	0.1012
(e) HGN	0.1281	0.0825	0.1899	0.0964	0.2836	0.1231
(f) BERT4Rec	0.1152	0.0712	0.1658	0.0894	0.2345	0.1091
(g) STOSA	0.1302	0.0826	0.1881	0.1013	0.2741	0.1221
(h) DHCN	0.1290	0.0812	0.1697	0.0941	0.2329	0.1025
(i) SURGE	0.1378	0.0875	0.2037	0.1102	0.2957	0.1278
(j) GES	0.1320	0.0815	0.1737	0.0955	0.2474	0.1085
(k) DIEN	0.1175	0.0754	0.1792	0.0912	0.2540	0.1106
(l) SRJGraph	0.1217	0.0784	0.1817	0.0927	0.2618	0.1147
(m) SAPL	0.1483	0.0942	0.2247	0.1086	0.3017	0.1338
(n) ReMR	0.1451	0.0931	0.2135	0.1050	0.2986	0.1314
(o) RKSA	0.1357	0.0824	0.1978	0.1024	0.2847	0.1201
(p) MT4SR	0.1507	0.0869	0.2314	0.1132	0.3124	0.1411
(q) DFAR	0.1368	0.0841	0.1982	0.1031	0.2862	0.1224
(r) KERL	0.1371	0.0836	0.1994	0.1047	0.2934	0.1253
(s) Ours	**0.1629**	**0.1075**	**0.2519**	**0.1352**	**0.3620**	**0.1635**

The best results are in **boldface**, and the second-best results are underlined. The improvements achieved by our model over baseline methods are significant with p-value smaller than 0.001.

Table 6. Recommendation Performance Achieved by Different Methods in Terms of HR and NDCG on the Video Games Dataset

Methods	Video Games					
	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
(a) Light-GCN	0.0171	0.0116	0.0265	0.0152	0.0459	0.0201
(b) Caser	0.0182	0.0126	0.0281	0.0156	0.0455	0.0198
(c) SASRec	0.0573	0.0352	0.0836	0.0435	0.1238	0.0554
(d) GRU	0.0436	0.0277	0.0705	0.0361	0.1109	0.0463
(e) HGN	0.0456	0.0293	0.0760	0.0391	0.1177	0.0492
(f) BERT4Rec	0.0555	0.0348	0.0815	0.0425	0.1185	0.0535
(g) STOSA	0.0602	0.0391	0.0962	0.0501	0.1427	0.0622
(h) DHCN	0.0470	0.0295	0.0794	0.0399	0.1222	0.0507
(i) SURGE	0.0587	0.0378	0.0875	0.0415	0.1246	0.0542
(j) GES	0.0618	0.0375	0.1031	0.0503	0.1421	0.0614
(k) DIEN	0.0528	0.0334	0.0803	0.0412	0.1185	0.0506
(l) SRJGraph	0.0514	0.0327	0.0756	0.0392	0.1071	0.0459
(m) SAPL	0.0608	0.0381	0.0912	0.0485	0.1419	0.0610
(n) ReMR	0.0601	0.0374	0.0903	0.0480	0.1397	0.0601
(o) RKSA	0.0586	0.0367	0.0849	0.0455	0.1281	0.0572
(p) MT4SR	0.0614	0.0389	0.0916	0.0487	0.1421	0.0608
(q) DFAR	0.0593	0.0387	0.0862	0.0476	0.1304	0.0595
(r) KERL	0.0598	0.0381	0.0877	0.0496	0.1327	0.0598
(s) Ours	**0.0650**	**0.0416**	**0.1035**	**0.0536**	**0.1577**	**0.0669**

The best results are in **boldface**, and the second-best results are underlined. The improvements achieved by our model over baseline methods are significant with p-value smaller than 0.001.

Table 7. Recommendation Performance Achieved by Different Methods in Terms of HR and NDCG on the CD Dataset

Methods	CD					
	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
(a) Light-GCN	0.0098	0.0061	0.0152	0.0079	0.0254	0.0105
(b) Caser	0.0086	0.0053	0.0145	0.0072	0.0248	0.0098
(c) SASRec	0.0342	0.0203	0.0594	0.0272	0.0814	0.0364
(d) GRU	0.0084	0.0053	0.0144	0.0072	0.0239	0.0096
(e) HGN	0.0289	0.0187	0.0456	0.0239	0.0697	0.0303
(f) BERT4Rec	0.0335	0.0195	0.0568	0.0252	0.0795	0.0332
(g) STOSA	0.0324	0.0215	0.0494	0.0271	0.0742	0.0331
(h) DHCN	0.0176	0.0107	0.0298	0.0147	0.0476	0.0191
(i) SURGE	0.0315	0.0205	0.0482	0.0261	0.0722	0.0337
(j) GES	0.0351	0.0202	0.0620	0.0282	0.0971	0.0371
(k) DIEN	0.0304	0.0181	0.0567	0.0245	0.0779	0.0338
(l) SRJGraph	0.0317	0.0194	0.0574	0.0263	0.0792	0.0347
(m) SAPL	0.0407	0.0253	0.0649	0.0324	0.0925	0.0412
(n) ReMR	0.0390	0.0243	0.0638	0.0316	0.0901	0.0402
(o) RKSA	0.0367	0.0224	0.0613	0.0289	0.0847	0.0384
(p) MT4SR	0.0391	0.0246	0.0643	0.0317	0.0918	0.0412
(q) DFAR	0.0382	0.0237	0.0631	0.0296	0.0869	0.0390
(r) KERL	0.0385	0.0241	0.0621	0.0295	0.0858	0.0392
(s) Ours	**0.0439**	**0.0283**	**0.0682**	**0.0360**	**0.1004**	**0.0440**

The best results are in **boldface**, and the second-best results are underlined. The improvements achieved by our model over baseline methods are significant with p-value smaller than 0.001.

Table 8. Recommendation Performance Achieved by Different Methods in Terms of HR and NDCG on the Tmall Dataset

Methods	Tmall					
	HR@5	NDCG@5	HR@10	NDCG@10	HR@20	NDCG@20
(a) Light-GCN	0.1741	0.1432	0.2047	0.1605	0.2323	0.1792
(b) Caser	0.1399	0.1233	0.1544	0.1280	0.1678	0.1314
(c) SASRec	0.3549	0.2749	0.4347	0.3047	0.4747	0.3149
(d) GRU	0.2741	0.2394	0.3004	0.2480	0.3253	0.2543
(e) HGN	0.1324	0.1071	0.1610	0.1156	0.1910	0.1232
(f) BERT4Rec	0.3349	0.2618	0.4275	0.2975	0.4579	0.3067
(g) STOSA	0.4039	0.3011	0.4334	0.3308	0.4802	0.3326
(h) DHCN	0.4126	0.3175	0.4576	0.3329	0.5164	0.3467
(i) SURGE	0.3743	0.2847	0.4480	0.3146	0.4848	0.3185
(j) GES	0.4014	0.3090	0.4565	0.3345	0.5108	0.3380
(k) DIEN	0.3067	0.2473	0.3961	0.2758	0.4536	0.2904
(l) SRJGraph	0.2935	0.2278	0.3847	0.2646	0.4419	0.2851
(m) SAPL	0.3930	0.3023	0.4584	0.3206	0.5036	0.3317
(n) ReMR	0.3859	0.2981	0.4512	0.3168	0.4954	0.3253
(o) RKSA	0.3692	0.2847	0.4453	0.3124	0.4821	0.3196
(p) MT4SR	0.3970	0.3012	0.4602	0.3225	0.5127	0.3380
(q) DFAR	0.3758	0.2931	0.4503	0.3152	0.4921	0.3247
(r) KERL	0.3727	0.2914	0.4531	0.3209	0.4952	0.3277
(s) Ours	**0.4169**	**0.3236**	**0.4785**	**0.3433**	**0.5313**	**0.3583**

The best results are in **boldface**, and the second-best results are underlined. The improvements achieved by our model over baseline methods are significant with p-value smaller than 0.001.

improve the model's representation capabilities. For e-commerce datasets, e.g., Amazon CD and Video Games, we can observe relatively high improvements in the HR and NDCG scores. In the Tmall dataset, the improvement is relatively limited, since in this dataset we consider each session as an independent user and the user histories are sparse, the user preference is relatively random with respect to the item dependency due to the sparsity.

5.1 Ablation Study

We perform ablation studies on our model by showing how different components of our model can affect the recommendation performance in the following aspect:

— The usage of item dependency sub-graphs: whether the sub-graph improves the recommendation accuracy and how the sub-graph size affects the model performance.
— Dirichlet sampling: how the Dirichlet sampling with hyper-parameter α affects the performance.
— Hyperparameters: how the hyper-parameters, the embedding size, λ_1 and λ_2, affect the model's performance.

We also investigate two advantage aspects of our model:

— The adaptation of our proposed graph-based multi-view model: Can our model be adapted to other sequential user behavior encoders except for the transformer networks?
— Can other types of item correlations be adopted in our model?

5.1.1 Impact of Using Item Dependency Sub-graphs. To justify the effectiveness of applying item sub-graphs to the sequential recommendation, we show how the sub-graph's size can affect the performance of our model. The results of using different sub-graph sizes on the four datasets are shown in Table 9. From the experiment, we can infer that with the sub-graphs, our model

Table 9. Ablation Study for HR@10 and NDCG@10 with Different Sub-graph Sizes

Dataset	Yelp		ML-1M		Video Games		CD	
Metric	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
0-hop	0.0425	0.0214	0.1877	0.0950	0.0836	0.0435	0.0594	0.0272
1-hop	**0.0454**	0.0225	0.2434	0.1330	0.0942	0.0478	0.0655	0.0341
2-hop	0.0453	**0.0230**	**0.2583**	**0.1409**	**0.1068**	**0.0543**	**0.0682**	**0.0360**
3-hop	0.0450	0.0218	0.2519	0.1297	0.1010	0.0527	0.0671	0.0350

0-hop refers to the SASRec method.

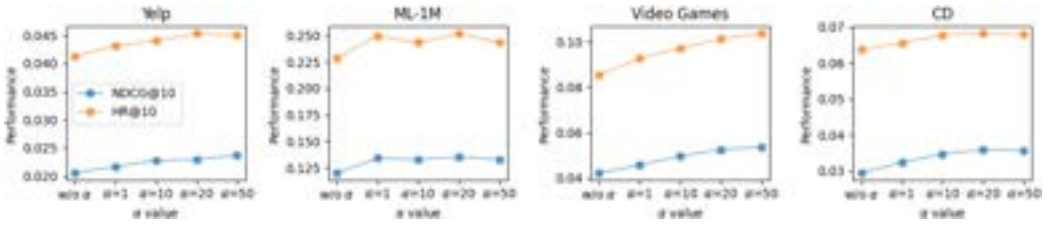

Fig. 4. Ablation study for HR@10 and NDCG@10 with different α values.

outperforms the SASRec model, which does not employ an item dependency graph to help the model make predictions. In addition, our model achieves high performance at around 2-hop sub-graph size. Using too large sub-graphs may hinder the performance, since when the sub-graph is too large, the dependencies between the centre item and border items of the sub-graph are not informative enough for predicting users' preferences.

5.1.2 Impact Dirichlet Sampling Parameter α. The Dirichlet Sampling Parameter $\{\alpha\}_{k=1}^{K}$ controls the variance of the randomly sampled dependency scores $\mathbf{T}_{i,j}^{(k)}$, which serves as an important hyper-parameter to mitigate the inductive bias and enhance our model's prediction accuracy.

Figure 4 shows the performance of our 2-hop sub-graph model with different Dirichlet sampling parameters α. We calculate the Dirichlet parameter for each item in the sub-graph as $\alpha_j^{(k)} = \alpha \cdot \mathbf{T}_{i,j}^{(k)}$ for items j in the kth hop of the sub-graph centred around item i. During the training, the items transition probability $\mathbf{T}$ is sampled from $\mathcal{D}(\alpha_1, \ldots, \alpha_K)$. With high α, the sampling is more concentrated around the expectation of the Dirichlet distribution, which is the true weight distribution obtained from the item-item dependency graph. From the results, we can infer that by creating perturbations to the item-item dependency graph, the model can achieve higher accuracy in the test set than the model without perturbations. Meanwhile, the perturbations need to be concentrated around the empirical estimation of the transition probability $\mathbf{T}_{i,j}^{(k)}$ to reduce the inaccuracy of the sampling.

5.1.3 Impact of the Dimensionality of Hidden Variables. We conduct experiments with embedding dimensionality from $\{16, 32, 64, 128\}$, to test the robustness and better understand the effect of dimensionality on the performance of our model. The results on four datasets are shown in Figure 5. It can be observed that with a small dimension of embedding size, e.g., 16, our model achieved inferior performance on all the datasets. Thus, a small embedding size is not sufficient to express the latent features of users and items. By increasing the dimension of the embedding, the model's performance improves and approaches stability when the dimension reaches 128.

5.1.4 Impact of λ_1 and λ_2. In Equation (16), λ_1 and λ_2 are hyper-parameters to control the weight of losses from individual views and contrastive losses among the views, respectively. Properly controlling the weight of these two terms can enhance the performance of the model. In

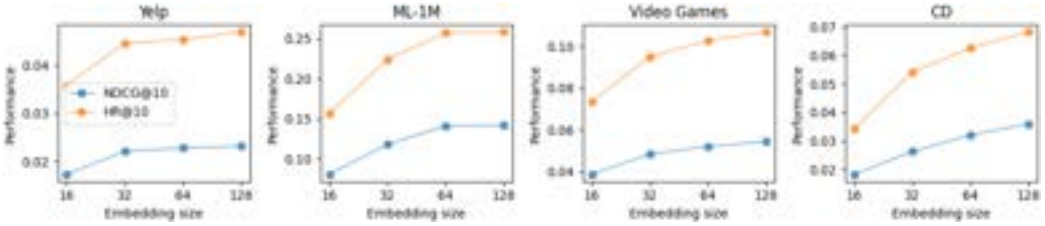

Fig. 5. Ablation study for HR@10 and NDCG@10 with different embedding sizes: 16, 32, 64, and 128.

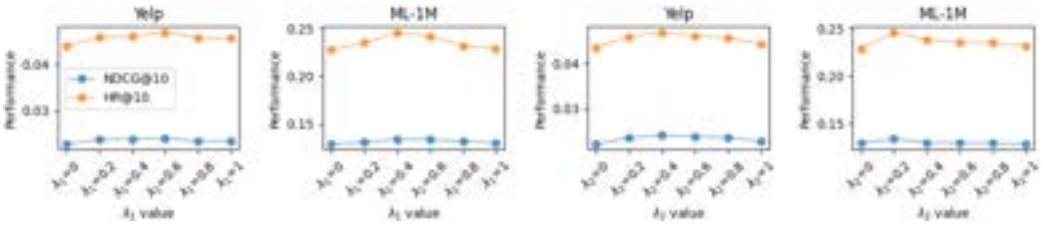

Fig. 6. Ablation study for HR@10 and NDCG@10 with different λ_1 and λ_2 values. The left two figures show the performance for different λ_1 values, and the right two figures show the performance for different λ_2 values.

Table 10. Ablation Study on the Adaptability of Our Model

Dataset	Yelp		ML-1M		Video Games		CD	
Metric	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Transformer	0.0425	0.0214	0.1877	0.0950	0.0836	0.0435	0.0594	0.0272
Transformer + graph multi-view	**0.0454**	**0.0230**	**0.2583**	**0.1409**	**0.1068**	**0.0543**	**0.0682**	**0.0360**
HGN	0.0401	0.0198	0.1899	0.0964	0.0760	0.0391	0.0456	0.0239
HGN + graph multi-view	**0.0428**	**0.0212**	**0.2081**	**0.1083**	**0.0816**	**0.0415**	**0.0483**	**0.0246**
GRU	0.0343	0.0165	0.1522	0.0804	0.0705	0.0361	0.0144	0.0072
GRU + graph multi-view	**0.0368**	**0.0179**	**0.1710**	**0.0927**	**0.0758**	**0.0380**	**0.0249**	**0.0127**
Caser	0.0361	0.0178	0.2018	0.1048	0.0280	0.0156	0.0145	0.0072
Caser + graph multi-view	**0.0382**	**0.0185**	**0.2246**	**0.1137**	**0.0492**	**0.0251**	**0.0286**	**0.0146**

We fit our proposed multi-view graph model into different encoder networks: HGN, GRU, and Caser.

Figure 6, we show the impact of varying different values of λ_1 and λ_2 to the performance of our model on the datasets Yalp and ML-1M. When $\lambda_1 = 0$ or $\lambda_2 = 0$, the corresponding loss term is removed from our proposed model. As can be seen a moderate weight on the regularization terms enhances the performance of our model.

5.1.5 Adaptability to Different Encoder Networks. Our proposed graph-based sequential recommendation model can not only be applied to transformers backbone but also able to enhance representation learning for other sequential networks such as RNNs. In Table 10, we show the adaptivity of our model to several commonly used sequential encoders including user gating networks (i.e., HGN), RNN module (i.e., GRU), and convolution networks (i.e., Caser). In this experiment, we replace the default transformer networks with the aforementioned networks, while keeping other parts the same as our proposed model. The results show that our graph-based multi-view model enhances the performance of these architectures. Specifically, the Caser model's performance increases from 0.028 to 0.0496 and from 0.0145 to 0.0286 on HR@10 in video games and CD datasets, respectively.

Table 11. Ablation Study for Different Item Graph Construction Methods

Dataset	ML-100k		Games		Steam		ML-1M	
Metric	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
(a) Time sequence	**0.0453**	**0.0230**	**0.2519**	**0.1352**	**0.1020**	**0.0536**	**0.0682**	**0.0360**
(b) Co-occurrence	0.0446	0.0221	0.2506	0.1345	0.0948	0.0483	0.0539	0.0287

We compare our proposed item graph construction through time sequence, shown in (a); as well as the item graph construction through co-occurrence, shown in (b).

5.1.6 Forms of Item Correlation Graphs. In Table 11(b), we adopt another commonly used item-correlation graph: If two items are clicked by the same user, then we add an edge to the item pair, and if the edge has been added, then we add +1 to the edge's weight. After constructing the graph, we normalized the weights to make the adjacency matrix symmetrically column-stochastic, i.e., the sum of each column of the adjacency matrix is one. Since the adjacency matrix is symmetric, it is also row-stochastic. In the following steps, we use the normalized item-correlation graph to train our model, and the rest of the procedures follow our method in Section 3. Compared to Table 11(a), which represents our default proposed graph construction method, we can recognize that constructing item-dependency graphs based on the time sequence of user's behaviors achieves higher performance, which also serves as a key aspect in the sequential recommendations.

To summarize, we mainly contribute the performance advantages of our model into four aspects,

— The transformer model can perfectly handle the item sequences and learn the item embeddings with respect to the user's trend of preference.
— The usage of item dependency sub-graphs and the hierarchical graph aggregation model can help to improve the representation capability and accuracy.
— The formation of multiple views and the application of mutual information maximization enhance the model's performance.
— The usage of Dirichlet sampling can mitigate the stiffness of the sub-graphs. By involving perturbations of the graph sampling, the model can learn a more robust estimation of the user's preference.

Together, as demonstrated in the ablation experiments, these components contribute to the model's overall effectiveness in addressing recommendation challenges and provide a powerful and efficient solution for predicting users' next preferred items in a general sequential recommendation setting.

6 CONCLUSIONS AND FUTURE WORK

In this article, we propose a multi-view graph-based sequential recommendation model, in which we design the hierarchical graph aggregation networks to aggregate local information of items from the item dependency graph, and we apply the transformer model to process the sub-graph representations of each user-clicked item. Finally, we combine the representations from multiple views to predict the next preferred item by the user. In our model, we create Dirichlet weight sampling and mutual information maximization techniques to enhance our model's accuracy. Dirichlet weight sampling enables our model to dynamically sample multi-hop neighbours with low time and memory costs while preventing our model from overfitting on specific sets of high-weighted neighbours. From the time complexity perspective, we demonstrate that our model is suitable for modeling user's sequential behaviors with evidently low time cost, compared to applying other types of GNN-based user behavior modeling in sequential recommendations.

To evaluate our model, we conduct extensive experiments using five publicly available recommendation datasets and compare its performance against multiple state-of-the-art baselines. Our model consistently outperforms the competing baselines. Ablation studies further confirm the

effectiveness of each component in our model. In future work, we plan to generalize our model for personalized recommendations that balance long-term and short-term user preferences, ensuring a more nuanced understanding of users' interests. Another promising area is to expand our model's applicability by incorporating general knowledge graphs, enabling the capture of a wider range of relationships and information for improved recommendations.

REFERENCES

[1] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Renqin Cai, Jibang Wu, Aidan San, Chong Wang, and Hongning Wang. 2021. Category-aware collaborative sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 388–397.

[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'21)*. 378–387.

[4] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured graph convolutional networks with stochastic masks for recommender systems. In *Proceedings of the Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'21)*. 614–623.

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. Retrieved from https://arXiv:1801.10247

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 257–266.

[7] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. Retrieved from https://arXiv:1409.1259.

[8] Yue Cui, Hao Sun, Yan Zhao, Hongzhi Yin, and Kai Zheng. 2021. Sequential-knowledge-aware next POI recommendation: A meta-learning approach. *ACM Trans. Info. Syst.* 40, 2 (2021), 1–22.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 4171–4186.

[10] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S. Yu. 2022. Sequential recommendation via stochastic self-attention. Retrieved from arXiv:2201.06035.

[11] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Trans. Info. Syst.* 39, 1 (2020), 1–42.

[12] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, and Yeow Meng Chee. 2015. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. ACM, 2069–2075.

[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. Retrieved from https://arXiv:1706.02216

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 639–648.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the World Wide Web Conference (WWW'17)*. 173–182.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'18)*. 843–852.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[18] Balázs Hidasi and Domonkos Tikk. 2016. General factorization framework for context-aware recommendations. *Data Min. Knowl. Discov.* 30, 2 (2016), 342–371.

[19] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'19)*. 573–581.

[20] Liwei Huang, Yutao Ma, Yanbo Liu, Bohong Danny Du, Shuliang Wang, and Deyi Li. 2023. Position-enhanced and time-aware graph convolutional network for sequential recommendations. *ACM Trans. Info. Syst.* 41, 1 (2023), 1–32.

[21] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 306–310.
[22] Mingi Ji, Weonyoung Joo, Kyungwoo Song, Yoon-Yeong Kim, and Il-Chul Moon. 2020. Sequential recommendation with relation-aware kernelized self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4304–4311.
[23] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'18)*.
[24] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. Retrieved from https://arXiv:1609.02907
[25] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20)*. 1748–1757.
[26] Chenliang Li, Xichuan Niu, Xiangyang Luo, Zhenzhong Chen, and Cong Quan. 2019. A review-driven neural model for sequential recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*.
[27] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'20)*. 322–330.
[28] Guanyu Lin, Chen Gao, Yu Zheng, Jianxin Chang, Yanan Niu, Yang Song, Zhiheng Li, Depeng Jin, and Yong Li. 2023. Dual-interest Factorization-heads Attention for Sequential Recommendation. In *Proceedings of the ACM Web Conference*. 917–927.
[29] Feng Liu, Qing Liu, Wei Guo, Huifeng Guo, Weiwen Liu, Ruiming Tang, Xutao Li, Yunming Ye, and Xiuqiang He. 2020. Inter-sequence Enhanced Framework for Personalized Sequential Recommendation. Retrieved from https://arXiv:2004.12118
[30] Yong Liu, Susen Yang, Chenyi Lei, Guoxin Wang, Haihong Tang, Juyong Zhang, Aixin Sun, and Chunyan Miao. 2021. Pre-training graph transformer with multimodal side information for recommendation. In *Proceedings of the ACM International Conference on Multimedia (ACM MM'21)*.
[31] Yong Liu, Susen Yang, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. 2021. Contextualized graph attention network for recommendation with item knowledge graph. *IEEE Trans. Knowl. Data Eng.* (2021).
[32] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 825–833.
[33] Muyang Ma, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Huasheng Liang, Jun Ma, and Maarten de Rijke. 2022. Improving transformer-based sequential recommenders through preference editing. *ACM Trans. Info. Syst.* (2022).
[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. Retrieved from https://arXiv:1807.03748
[35] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star graph neural networks for session-based recommendation. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'20)*. 1195–1204.
[36] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An intent-guided collaborative machine for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1833–1836.
[37] Sung-Jun Park, Dong-Kyu Chae, Hong-Kyun Bae, Sumin Park, and Sang-Wook Kim. 2022. Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 784–793.
[38] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Trans. Info. Syst.* 38, 3 (2020), 1–23.
[39] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'19)*.
[40] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. Gag: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*.
[41] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.
[42] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. Retrieved from https://arXiv:2005.10602.
[43] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the World Wide Web Conference (WWW'10)*. 811–820.
[44] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender Systems Handbook*. Springer, 257–297.

[45] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'19)*. 1441–1450.
[46] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'18)*. 565–573.
[47] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.
[48] Md Mehrab Tanjim, Congzhe Su, Ethan Benjamin, Diane Hu, Liangjie Hong, and Julian McAuley. 2020. Attentive sequential models of latent intent for next item recommendation. In *Proceedings of the Web Conference*. 2528–2534.
[49] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive multiview coding. Retrieved from https://arXiv:1906.05849
[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Adv. Neural Info. Process. Syst.* 30 (2017).
[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. Retrieved from https://arXiv:1710.10903
[52] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 209–218.
[53] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. 403–412.
[54] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[55] Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of the Web Conference*.
[56] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 165–174.
[57] Xiting Wang, Kunpeng Liu, Dongjie Wang, Le Wu, Yanjie Fu, and Xing Xie. 2022. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In *Proceedings of the ACM Web Conference*. 2098–2108.
[58] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'20)*. 169–178.
[59] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 495–503.
[60] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Trans. Knowl. Data Eng.* (2022).
[61] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'20)*. 328–337.
[62] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19)*. 346–353.
[63] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'21)*. 4503–4511.
[64] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *Proceedings of the International Conference on Data Engineering (ICDE'22)*. 1259–1273.
[65] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S. Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *Proceedings of the World Wide Web Conference*. 3398–3404.
[66] Mingming Xu, Fangai Liu, and Weizhi Xu. 2019. A survey on sequential recommendation. In *Proceedings of the International Conference on Software Engineering (ICISCE'19)*.
[67] Lyuxin Xue, Deqing Yang, Shuoyao Zhai, Yuxin Li, and Yanghua Xiao. 2022. Learning dual-view user representations for enhanced sequential recommendation. *ACM Trans. Info. Syst.* (2022).

[68] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1921–1924.

[69] Zeping Yu, Jianxun Lian, Ahmad Mahmoody, Gongshen Liu, and Xing Xie. 2019. Adaptive user modeling with long and short-term preferences for personalized recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*. 4213–4219.

[70] Raphael Yuster and Uri Zwick. 2005. Fast sparse matrix multiplication. *ACM Trans. Algor.* 1, 1 (2005), 2–13.

[71] Wei Zhang, Zeyuan Chen, Hongyuan Zha, and Jianyong Wang. 2021. Learning from substitutable and complementary relations for graph-based sequential product recommendation. *ACM Trans. Info. Syst.* 40, 2 (2021), 1–28.

[72] Kai Zhao, Yukun Zheng, Tao Zhuang, Xiang Li, and Xiaoyi Zeng. 2022. Joint learning of e-commerce search and recommendation with a unified graph neural network. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 1461–1469.

[73] Lin Zheng, Naicheng Guo, Weihao Chen, Jin Yu, and Dazhi Jiang. 2020. Sentiment-guided sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1957–1960.

[74] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19)*. 5941–5948.

[75] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[76] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*. 1059–1068.

[77] Kun Zhou, Hui Wang, Ji-Rong Wen, and Wayne Xin Zhao. 2023. Enhancing multi-view smoothness for sequential recommendation models. *ACM Trans. Info. Syst.* (2023).

[78] Nengjun Zhu, Jian Cao, Yanchi Liu, Yang Yang, Haochao Ying, and Hui Xiong. 2020. Sequential modeling of hierarchical user intention and preference for next-item recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 807–815.

[79] Tianyu Zhu, Leilei Sun, and Guoqing Chen. 2021. Graph-based Embedding Smoothing for Sequential Recommendation. *IEEE Trans. Knowl. Data Eng.* (2021).

Received 17 April 2023; revised 20 January 2024; accepted 26 January 2024