Information Sciences 648 (2023) 119510

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Graph-coupled time interval network for sequential recommendation

Bin Wu[a], Tianren Shi[a], Lihong Zhong[b], Yan Zhang[c,d,*], Yangdong Ye[a]

[a] *School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, 450001, China*
[b] *School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou, 450001, China*
[c] *Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518028, China*
[d] *Intelligent Terminal Key Laboratory of Sichuan Province, Yibin 644000, China*

ARTICLE INFO

Keywords:
Sequential recommendation
Graph neural network
Time interval
Self-attention mechanism

ABSTRACT

Modeling the dynamics of sequential patterns (*i.e.*, sequential recommendation) has obtained great attention, where the key problem is how to infer the next interesting item according to users' historical actions. Owing to high efficiency and accuracy, several Transformer-like frameworks have successfully achieved this task without adopting complicated recurrent or convolutional operations. Nevertheless, they focus only on the user-item bipartite graph and forgo other auxiliary information, which is non-trivial to attain satisfactory performance especially under long-tail distribution scenarios. In modeling short-term user interests, they fail to capture the time intervals between the recent actions and the target timestamp, which may result in the suboptimal performance. To settle such two problems, we propose a novel architecture for the task of sequential recommendation, namely *graph-coupled time interval network* (GCTN). Specifically, by means of item category information, we devise a category-aware graph propagation module to better learn user and item embeddings. Furthermore, we design a time-aware self-attention mechanism, which explicitly captures the effect of the time interval between two actions for next item prediction. To integrate these two parts into an organic whole, we introduce a personalized gating strategy to differentiate the importance of each part under the special context. Extensive experiments demonstrate the effectiveness and efficiency of GCTN over recent state-of-the-art methods on four real-world datasets, seamlessly combining the advantages of graph neural networks and Transformers.

1. Introduction

With the rapid evolution of the Internet, the ever-growing amount of user behavior data is exacerbating the information overload problem. Personalized recommendation, as a vital and indispensable tool for alleviating such issue, has been deployed in many business applications such as streaming micro-video and e-commerce services. Collaborative filtering (CF), which is the most prevalent technique, provides personalized services by modeling the user-item bipartite graph [41]. The past decade has witnessed the evolution of such direction, spanning from the early neighborhood-based methods [26] and matrix factorization (MF) [34,38] to the recent graph neural networks (GNN) based recommenders [4,32]. Nonetheless, they concentrate only on the user-item bipartite graph and

* Corresponding author at: Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518028, China.
E-mail address: yixianqianzy@gmail.com (Y. Zhang).

https://doi.org/10.1016/j.ins.2023.119510
Received 24 September 2022; Received in revised form 12 June 2023; Accepted 8 August 2023

Available online 14 August 2023

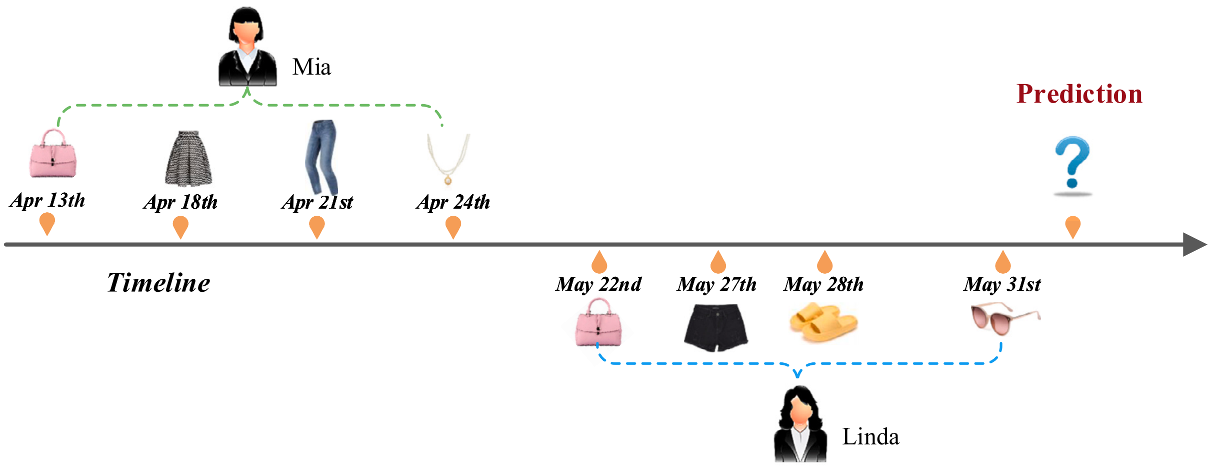

Fig. 1. An illustration of the time interval information for sequential recommendation.

forgo other auxiliary information such as timestamp, item category, and user profile. While providing a general solution for building personalized services, these CF-based methods are non-trivial to yield satisfactory performance and could be significantly improved by considering this important information.

As user interests are dynamic by nature, inferring what items a user will visit next time is more practical than learning user's generic taste. Such problem is referred to as sequential recommendation [3,13], which aims to model and distinguish long-term and short-term user interests. As the pioneering solution, Rendle et al. [25] formalized the factorized personalized Markov chain (FPMC) method, complementing a matrix factorization model (MF) with capturing the transition pattern between the recently visited item and the next item. While a first-order Markov assumption should be obeyed in FPMC, several subsequent works attempt to learn sequential patterns from high-order Markov chains, such as HGN [22] and ATM [35]. With the "renaissance" of deep learning techniques, Tang et al. [30] treated the high-order MCs as an "image" and developed a convolutional sequence embedding recommender, which employs different convolutional filters to capture hierarchically sequential patterns. Alternatively, inspired by the advance of sequence modeling in natural language understanding, various recurrent neural networks (RNN) based approaches [6,7,9] have also been introduced to excavate item-to-item transitions. Though achieving promising results, these RNN-based methods are difficult to capture non-adjacent transition patterns and fully leverage the parallel computation of modern hardware (*e.g.,* GPU).

Instead of adopting complicated recurrent or convolutional operations, recent self-attention mechanism in Transformer has achieved significant performance and efficiency in many tasks, such as machine translation, computer vision, and sequential recommendation [11,14]. Kang et al. [11] proposed the self-attentive sequential recommender (SASRec) and achieved comparable results; it embeds a positional encoding strategy to capture the relative order information, and applies the self-attention mechanism to excavate item transition patterns. Later on, Li et al. [14] extended the SASRec framework by exploring the time interval information between two adjacent items. Recently, Zhu et al. [49] performed GCN on a hybrid item relation graph to produce smoothed embeddings, and equipped SASRec with these embeddings. Despite effectiveness, current Transformer-based models still suffer from the following limitations:

- *Non-trivial to learn high-quality embeddings from the user-item bipartite graph.* In real-world scenarios, many user-item bipartite graphs are long-tailed. As shown in Fig. 3, the head nodes (*i.e.*, active users and popular items) have enough action records, meanwhile the tail nodes (*i.e.*, inactive users and unpopular items) have limited actions. To alleviate the long-tail problem, one representative solution is to learn head and tail node embeddings separately [21,48]. Nonetheless, its performance depends heavily on the segmentation manner, which may hurt the model's flexibility.
- *Neglecting the effect of the time intervals between recently successive actions and the target timestamp.* Previous time-aware sequential recommenders focus on the time interval between two adjacent items. Nevertheless, they fail to notice an intuition that the closer an interacted item is to the user's current timestamp, the more important this item is to infer the next one. To interpret such defect, Fig. 1 provides a toy example in the E-Commerce scenario. The Mia's consumed records are $\{(Handbags, Apr\ 13^{th})$, $(Skirt, Apr\ 18^{th})$, $(Jeans, Apr\ 21^{st})$, $(Jewelry, May\ 24^{th})\}$, and the Linda's recent purchase records are $\{(Handbags, May\ 22^{nd})$, $(Shorts, May\ 27^{th})$, $(Slippers, May\ 28^{th})$, $(Sunglasses, May\ 31^{st})\}$. Although the time interval between two adjacent items is identical in two users, the same item "Handbags" should contribute more information for Linda's short-term interest at time step t than for Mia's.
- *Non-trivial to settle the semantic gap between collaborative signals and sequential patterns.* After exploiting collaborative signals and capturing sequential patterns, previous sequential recommenders usually employ a concatenation or summation operation to make a prediction. While such two manners are straightforward, they fail to differentiate the significance of each part under the special context.

To settle these three deficiencies, we contribute a graph-coupled time interval network (GCTN) for sequential recommendation, which focuses on item category and time interval information to improve the recommendation quality. To be specific, we devise a category-aware graph propagation module to attain long-term user preference. By this way, we could explicitly encode collaborative signals in the user-item-category tripartite graph and greatly alleviate the long-tail issue. Furthermore, we introduce a time-aware self-attention mechanism, which could explore long-range item transition patterns and capture the effect of the time intervals between recently successive actions and the target item. Herein the key challenge is how to settle the semantic gap between these two parts and fuse them into an organic whole. In light of this, we develop a personalized gating strategy to distinguish the importance of each part under the special context. All in all, our contributions are as follows:

- We propose a novel recommendation solution called GCTN, which seamlessly combines the advantages of GNNs and Transformers. To our best knowledge, this paper is the first to highlight the importance of jointly exploring item category and time interval information for next item prediction.
- We design a category-aware graph propagation module to learn robust user and item embeddings. Furthermore, a time-sensitive self-attention mechanism is designed to explore the influence of the time interval information for predicting user preference.
- We develop a personalized gating strategy to eliminate the semantic gap between collaborative signals and sequential patterns. In this way, GCTN could determine which module is more important for inferring the next item.
- We perform extensive experiments under four recommendation scenarios, demonstrating that GCTN has remarkable performance over recent state-of-the-art methods. Further ablation studies justify the rationality and necessity of each component.

The rest of this article is organized as follows. Section 2 describes previous studies related to our proposed GCTN. In Section 3, we formulate the solved problem and present our Graph-Coupled Time Interval Network. We conduct comprehensive experiments on four large-scale datasets to verify the effectiveness of GCTN for sequential recommendation in Section 4. Finally, we conclude this paper and discuss future work in Section 5.

2. Related work

2.1. General recommendation

Learning informative user and item embeddings from historical behavior data is the core of a general recommendation. Among these general recommenders, CF-based methods have been widely utilized in various applications, such as location-based social networks and e-commerce services. MF-based models, as one of the most popular CF techniques, have achieved great success in the early stage. Theoretically, a standard MF-model usually maps each user/item to a dense vector (*i.e.*, embedding), and produces a prediction based on the inner product of these two embeddings. Beyond merely depending on the one-hot encoding of a user (an item), several follow-up works augment the user/item embedding learning with the representations of his/her neighbor nodes. For instance, Kabbur et al. [10] proposed a factored item similarity model (FISM), regarding the average representations of historical items as a user's embedding. Afterwards, He et al. [5] extended the FISM method by using an attentive strategy to distinguish the contribution of each historical item; and Chen et al. [2] augmented the user embedding learning by applying a hierarchical attention on the historical items and their multimedia content. When reorganizing all users' action records as a bipartite graph, the performance gains are derived from the usage of the subgraph structure of a user.

To deepen the user-item interaction graph, another relevant research line is exploring GNNs for recommender systems. Inspired by vanilla GCN [12], Wang et al. [32] developed the neural graph CF (NGCF) framework, performing multiple message propagation layers to encode collaborative signals into the embedding learning. To speed up the model convergence, He et al. [4] omitted nonlinear feature transformation in NGCF and developed a lightweight graph convolutional network (LightGCN), which obtained impressive performance gains over NGCF. Afterwards, Wu et al. [40] further improved the LightGCN framework by leveraging contrastive learning to strengthen the model's robustness. Recently, Cai et al. [1] performed singular value decomposition (SVD) on the user-item bipartite graph to generate augmentation view, and devised a local-global contrastive learning framework for recommender systems.

However, the aforementioned general recommenders depend only on the historical behaviors to model long-term user preference. As described in the introduction, many real-world recommendation scenarios confront the long-tailed problem. In other words, a large fraction of nodes has very limited neighbors. In this case, it is very hard to get informative embeddings for tail nodes (*i.e.*, inactive users and unpopular items). More importantly, these methods overlook the sequential information, leading to the suboptimal recommendation results.

2.2. Sequential recommendation

Compared with general recommendation that models user interests in a time-independent fashion, the goal of sequential recommendation is to predict the next item that a user will visit in future work [13,37]. During the early phase, FPMC-based method combines a MF model with a first-order MC to simultaneously model long-term user preference and short-term dynamic interest. Subsequently, the translation-based recommender (TransRec) projected all items into a metric space and modeled item-to-item transitions by regarding the target user as a transition vector. Wu et al. [35] further improved TransRec by modeling high-order MCs with a position-specific attention strategy. Despite simplicity, such methods only capture single-way transition patterns between two items

Table 1
Notations.

Symbol	Explanation
$\mathcal{U}$, $\mathcal{I}$, $\mathcal{C}$	User set, item set and category set
$\mathcal{I}_u$	The set of items that user u has consumed
$\mathcal{U}_i$	The set of users who have visited item i
$\mathcal{S}^u$	A subsequence of user u: $\{\mathcal{S}^u_{t-L}, \cdots, \mathcal{S}^u_{t-1}\}$
$\mathcal{S}^u_{t-1}$	The item that user u interacted at time step $t-1$
$\boldsymbol{E}, \boldsymbol{M}$	Embedding matrices associated with users and items
$\boldsymbol{H}$	Embedding matrix associated with categories
T^u_t	The absolute timestamp that user u consumed an item
$\hat{r}_{u,t,i}$	The likelihood that user u accesses item i at time step t
D	The embedding size

and neglect complex transition patterns within a sequence (*e.g.,* many-to-one). With the "renaissance" of deep learning techniques, several CNN-based methods regard high-order MCs as an "image" and employ different convolution kernels to capture sequential patterns, including convolutional sequence embedding recommender (Caser) [30], convolutional generative network [45], 2D convolutional recommender [44], among others. Alternatively, inspired by the progress of sequence modeling in machine translation, some researchers have also realized the advantages of applying RNN in session-based recommendation. In particular, GRU4Rec [7] and GRU4Rec+ [6] utilized gated recurrent units to capture short-term user interests. Li et al. [15] introduced a neural attentive recommender machine, which simultaneously embeds a global and local RNN to model user's short sessions. Ji et al. [9] proposed a multi-hop time-aware attentive memory network (MTAM), unifying the strength of user memory network at modeling long-term and the power of the time-aware GRU network at capturing short-term intent. Recently, Xu et al. [43] developed the recurrent convolutional network for next-item recommendation, which unifies the advantage of RNN at capturing long-term dependencies and the power of CNN at modeling short-term sequential patterns. Nevertheless, it is well known that RNN-based methods fail to capture transition patterns of non-adjacent items and are difficult to leverage parallel computation of modern hardware.

More recently, Transformer as a new sequential architecture sheds light on alleviating this efficiency issue, and has obtained remarkable results in many machine learning tasks. Instead of adopting recurrent or convolutional module, it is a classical encoder-decoder structure to capture global dependencies between input and output. In particular, borrowing the self-attention mechanism in Transformer, Kang et al. [11] introduced a self-attentive sequential recommender (SASRec), which composes of multiple self-attention layers, point-wise feed-forward network and residual connections. Afterwards, Lin et al. [20] augmented the SASRec framework with an item similarity model, where the capturing of users' global representation plays an important role in improving the mode performance. Li et al. [14] presented a time interval aware self-attention network for sequential recommendation, which investigates the impact of the time interval between two adjacent items for predicting the next action. Recently, Li et al. [16] replaced the self-attention mechanism in SASRec with a pure MLP operation and employed a tri-directional mixer to capture sequential patterns. The solution that is most relevant to this paper is [49], which presents a graph-enhanced embedding smoothing method for next-item recommendation; it first performs GCN on an item relation graph to generate informative item embeddings, and then equips SASRec with these embeddings to improve its performance.

Our GCTN follows this thread but is distinct in three points: (1) to alleviate the long-tailed problem, we devise a category-aware graph propagation module to learn informative user and item embeddings; (2) we design a time-sensitive self-attention mechanism, which explicitly exploits the time intervals between recently successive actions and the target item; (3) we introduce a personalized gating strategy, which could dynamically determine the contribution of each part in GCTN under different contexts.

3. Graph-coupled time interval network

This section first provides the definition of user-item-category tripartite graph and formalizes the solved problem. Afterwards, we elaborate the Graph-Coupled Time Interval Network (GCTN) to explore item category and time interval information. As illustrated in Fig. 2, our solution consists of two key modules: the upper part corresponds to the category-aware graph propagation module that models long-term user preference, and the bottom part corresponds to the time-sensitive self-attention mechanism that captures item transition patterns. Finally, we present the optimization criterion and theoretical analysis in detail.

3.1. Problem formulation

In later sections, we use lowercase letters to represent scalars (*e.g.,* u) and bold lowercase letters to denote vectors (*e.g.,* $\boldsymbol{e}$). All matrices and sets are described by bold uppercase letters (*e.g.,* $\boldsymbol{W}$) and uppercase calligraphic letters (*e.g.,* $\mathcal{I}$), respectively. Table 1 lists some crucial notations and their descriptions.

Definition 1. *(User-Item-Category Tripartite Graph).* Let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items, respectively. Suppose $\mathcal{C}$ is the set of item categories, where each item belongs to one category. We use $\boldsymbol{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ to denote user-item interaction matrix, and each element indicates whether user u has visited item i. According to $\mathcal{C}$ and $\boldsymbol{R}$, we could construct a tripartite graph $G = (\mathcal{U} \cup \mathcal{I} \cup \mathcal{C}, \mathcal{E})$, where $\mathcal{E}$ stands for the set of edges between two nodes.

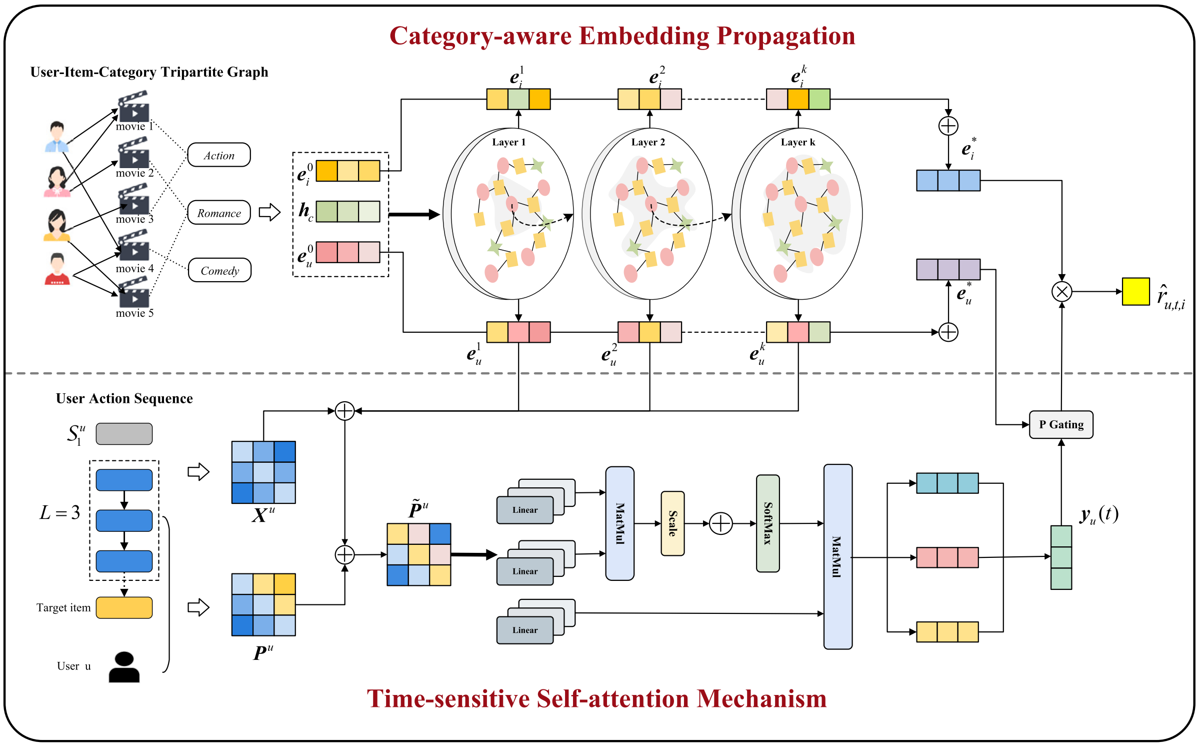

Fig. 2. A visualization of our proposal.

Definition 2. *(User Action Sequence).* For each user u, we have a behavior set $\mathcal{I}_u$. Based on the absolute timestamp, we sort his/her historical actions in chronological order, and attain a sequence $\left\{S_1^u, \dots, S_t^u, \dots, S_{|S^u|}^u\right\}$, where $S_t^u \in \mathcal{I}$ implies user u consumed an item at time step t. On the top of this, we extract a subsequence $\mathcal{S}^u = \left\{S_{t-L}^u, \dots, S_{t-1}^u\right\}$, composing of the recently successive L items before time step t.

According to the above definitions, we could model long-term user preference and short-term dynamic interest. Finally, we provide the description of our solved task as follows:

Definition 3. (Sequential Recommendation).
Input: *User-item-category tripartite graph G and each user $u \in \mathcal{U}$ that has the subsequence $\mathcal{S}^u \subseteq \mathcal{I}$.*
Output: *Provide a top-N recommendation list from $\mathcal{I} \backslash \mathcal{I}_u$ that user u has never visited but may be interested in.*

3.2. Embedding initialization

Following previous GNN-based recommenders [4,32,40], we map a user (an item) ID to a D-dimensional embedding $\boldsymbol{e}_u \in \mathbb{R}^D$ ($\boldsymbol{m}_i \in \mathbb{R}^D$). Analogously, we also project a category ID into an embedding ($\boldsymbol{h}_c \in \mathbb{R}^D$). As such, the embeddings of all nodes in user-item-category tripartite graph can be expressed by three matrices as follows:

$$\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|\mathcal{U}|}], \boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_{|\mathcal{I}|}], \boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_{|C|}]. \tag{1}$$

In order to capture sequential patterns, we follow other sequential recommenders [11,22] and additionally encode each item with a vector $\boldsymbol{p}_i \in \mathbb{R}^D$.

3.3. Category-aware graph propagation module

Previous Transformer-based recommenders [11,14,18] usually employ the initial embedding (*i.e.*, $\boldsymbol{e}_u$) to capture long-term user interest. While such manner is straightforward, it is non-trivial to attain satisfactory performance. The key obstacle is that they fail to explicitly encode collaborative signals, whereas they really exist in user-item bipartite graph. Careful readers might say it is a choice to mitigate this obstacle by implementing GNNs, such as NGCF [32] and LightGCN [4]. Nonetheless, the node degrees in the user-item bipartite graph follow a long-tailed distribution. In other words, a significant fraction of users has very few actions, and massive items were visited by a few users. If we directly apply GNNs on this long-tailed interaction graph, it is very hard to produce high-quality embeddings especially for tail nodes. To alleviate this issue, one representative solution is to learn head and tail node embeddings separately [21,48], whereas how to select the segmentation threshold is highly sensitive to domain-aware knowledge. Alternatively, item categorical features reveal the intrinsic characteristics, and shed light on settling this issue. Intuitively, we can regard each item category as a single node, and apply relational GCN [27] on the user-item-category tripartite graph. Since there exists the information richness between user-item relationships and item-category relations, it is non-trivial to harvest robust node embeddings. In light of this, we introduce a category-aware graph propagation module, which regards item category as context information and seamlessly embeds it into the embedding propagation rule. To be more specific, such module composes of two ingredients: category-aware embedding propagation and embedding readout.

1) *Category-aware embedding propagation*: Recent GNN-based methods have made significant advancements in recommendation, such as CF [4], session-based recommendation [42], and social recommendation [17]. The success of GNN lies in the power representation ability by explicitly modeling high-order neighbor information over graph-structural data. Albeit in fruitful progress, these GNN-based recommenders usually execute embedding propagation based upon user-item bipartite graph and confront the long-tail issue. By contrast, we build message propagation upon user-item-category tripartite graph. Formally, for a triplet (u, c, i), we define the message propagation rule at the kth layer from user u to item i as:

$$\boldsymbol{m}_i^{(k)} = f(\boldsymbol{e}_u^{(k-1)}, \boldsymbol{m}_i^{(k-1)}, \boldsymbol{h}_c), \tag{2}$$

where $f(\cdot)$ is a generic propagation function. If we ignore the term $\boldsymbol{h}_c$, several common graph propagation functions can be immediately implemented, such as GCN with an average aggregator [12], NGCF with a bi-interaction aggregator [32], and LightGCN with a linear aggregator [4]. Nevertheless, the introduction of item category information brings us light as well as challenge. In this paper, we implement the category-aware embedding propagation by:

$$\boldsymbol{m}_i^{(k)} = \sum_{u \in \mathcal{U}_i} \frac{1}{\sqrt{|\mathcal{I}_u||\mathcal{U}_i|}} \left(\boldsymbol{e}_u^{(k-1)} + \boldsymbol{e}_u^{(k-1)} \odot \boldsymbol{h}_c\right), \tag{3}$$

where $\frac{1}{\sqrt{|\mathcal{I}_u||\mathcal{U}_i|}}$ aims to avoid the overly large scale of vectors. $\odot$ stands for the element-wise product between two vectors. Distinct from the recent GNNs [4,19,39] that only consider the contribution of $\boldsymbol{e}_u$, herein we additionally inject the user interest towards the theme of item i into the embedding learning process. Analogously, we can also get the message propagation rule from item i to user u as:

$$\boldsymbol{e}_u^{(k)} = \sum_{i \in \mathcal{I}_u} \frac{1}{\sqrt{|\mathcal{U}_i||\mathcal{I}_u|}} \left(\boldsymbol{m}_i^{(k-1)} + \boldsymbol{m}_i^{(k-1)} \odot \boldsymbol{h}_c\right). \tag{4}$$

2) *Embedding readout*: After implementing the above part with the K times, we can get an embedding set for each user and item, *i.e.*, $\{\boldsymbol{e}_u^{(0)}, \cdots, \boldsymbol{e}_u^{(K)}\}$ and $\{\boldsymbol{m}_i^{(0)}, \cdots, \boldsymbol{m}_u^{(K)}\}$, respectively. Note that the subscript "0" denotes the initial embedding. To enrich the model ability, we integrate these embeddings to construct the final node representation. In our experiments, we employ the most simple strategy– average pooling. For other more complicated operations such as LSTM and attention mechanism, we explore them in future work. Formally, we can get

$$\boldsymbol{e}_u^* = \frac{1}{K+1} \sum_{k=0}^{K} \boldsymbol{e}_u^{(k)}, \quad \boldsymbol{m}_i^* = \frac{1}{K+1} \sum_{k=0}^{K} \boldsymbol{m}_i^{(k)}. \tag{5}$$

3.4. Time-sensitive self-attention mechanism

After modeling long-term user preference, how to excavate sequential patterns from the recently successive items (*i.e.*, $\mathcal{S}_u$) is the key to improving the recommendation quality. To capture such patterns, researchers have made a great deal of effort in two directions: CNN-based and RNN-based. CNN-based methods represent $\mathcal{S}_u$ as an image-like embedding matrix and extract sequential patterns by adopting various convolutional filters, such as Caser [30], CosRec [44], and NextItNet [45]. Meanwhile, RNN-based recommenders aim to capture sequential patterns of adjacent items, and generally perform better in denser datasets due to the higher model complexity. Instead of adopting recurrent or convolutional units, recent self-attention mechanism in Transformer has been successfully deployed to sequential recommendation, due to its power to excavate long-range dependencies within a user sequence. However, most of Transformer-like recommenders regard the user's action records as an ordered sequence, and forgo the absolute timestamps. While TiSASRec [14] and MTAM [9] explicitly exploited the time interval between two adjacent items, they dissatisfied an intuition that the closer the absolute timestamp of an interacted item is to the user's current time, the more important such item is for inferring future user actions. Continuing the early example in Fig. 1, the Mia's purchase history is $\{(Handbags, Apr\ 13^{th}), (Skirt, Apr\ 18^{th}), (Jeans, May\ 21^{st}), (Jewelry, Apr\ 24^{th})\}$, and the Linda's recent purchase records are $\{(Handbags, May\ 22^{nd}), (Shorts, May\ 27^{th}), (Slippers, May\ 28^{th}), (Suglasses, May\ 31^{st})\}$. Although the item "Handbags" lies in the same sequential position in two users, it is obvious that such item should contribute more to Linda's short-term interest than to Mia's. In light of this, we develop a time-sensitive self-attention mechanism to capture the complicated item transition patterns. To be more specific, it consists of three ingredients: time interval aware embedding, time-sensitive self-attention layer, and point-wise feed-forward layer.

1) *Time interval aware embedding*: For user u, we extract the recently successive L items to form a fixed-length sub-sequence, which is abbreviated as $\mathcal{S}^u = \{\mathcal{S}_{t-L}^u, \cdots, \mathcal{S}_{t-1}^u\}$. If the number of a user's actions is less than L, we repeatedly pad a "virtual" item to the right

until $|\mathcal{S}^u| = L$. Herein, we regard an invariant zero vector $\mathbf{0}$ as the representation of the "virtual" item. As such, the sub-sequence $\mathcal{S}^u$ can be expressed by an item embedding matrix $\boldsymbol{P}^u = [\boldsymbol{p}_{\mathcal{S}^u_{t-L}}, \cdots, \boldsymbol{p}_{\mathcal{S}^u_{t-1}}] \in \mathbb{R}^{L\times D}$. If we immediately apply $\boldsymbol{P}^u$ to capture long-range dependencies between items, our solution will confront two defects. On one hand, when two users have the same sub-sequence, our solution would model short-term user interest in an un-personalized manner. On the other hand, it is not aware of the time interval between a candidate item in $\mathcal{S}^u$ and the target item. To settle these limitations, we first inject the user embedding $\boldsymbol{e}^*_u$ into the embedding matrix $\boldsymbol{P}^u$:

$$\hat{\boldsymbol{P}}^u = \begin{bmatrix} \boldsymbol{p}_{\mathcal{S}^u_{t-L}} \oplus \boldsymbol{e}^*_u \\ \cdots \\ \boldsymbol{p}_{\mathcal{S}^u_{t-2}} \oplus \boldsymbol{e}^*_u \\ \boldsymbol{p}_{\mathcal{S}^u_{t-1}} \oplus \boldsymbol{e}^*_u \end{bmatrix} \in \mathbb{R}^{L\times d}, \tag{6}$$

where $\oplus$ denotes the additive operation. Furthermore, suppose the current timestamp of user u is T^u_t, we can achieve a time interval feature based on the timestamp T^u_{t-l} of item $\mathcal{S}^u_{t-L}$:

$$\boldsymbol{x}^u_{t-l} = \tanh(\boldsymbol{w}_x log(T^u_t - T^u_{t-l}) + \boldsymbol{b}_x), \tag{7}$$

where $\boldsymbol{w}_x, \boldsymbol{b}_x \in \mathbb{R}^D$ are the trainable parameters. Based on the action sequence $\mathcal{S}^u$, we can get the time interval matrix $\boldsymbol{X}^u = \{\mathbf{x}^u_{t-L}, \cdots, \mathbf{x}^u_{t-1}\}$. Furthermore, we can get the time interval aware embedding by injecting $\mathbf{X}^u$ into $\hat{\boldsymbol{P}}^u$:

$$\tilde{\boldsymbol{P}}^u = \begin{bmatrix} \hat{\boldsymbol{p}}_{\mathcal{S}^u_{t-L}} \oplus \mathbf{x}^u_{t-L} \\ \cdots \\ \hat{\boldsymbol{p}}_{\mathcal{S}^u_{t-2}} \oplus \mathbf{x}^u_{t-2} \\ \hat{\boldsymbol{p}}_{\mathcal{S}^u_{t-1}} \oplus \mathbf{x}^u_{t-1} \end{bmatrix} \in \mathbb{R}^{L\times d}. \tag{8}$$

2) *Time-sensitive self-attention layer*: Inspired by SASRec that adopts relative position self-attention mechanism, we introduce a time-sensitive self-attention to exploit the time interval between each candidate item in $\mathcal{S}^u$ and the target one. Specifically, we convert $\tilde{\boldsymbol{P}}^u$ to three matrices, *i.e.*, queries $\boldsymbol{Q} = \tilde{\boldsymbol{P}}^u \boldsymbol{W}_Q$, keys $\boldsymbol{K} = \tilde{\boldsymbol{P}}^u \boldsymbol{W}_K$ and values $\boldsymbol{V} = \tilde{\boldsymbol{P}}^u \boldsymbol{W}_V$, and feed them into an attentive operation:

$$\boldsymbol{S} = SA(\tilde{\boldsymbol{P}}^u) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right) \cdot \boldsymbol{V}, \tag{9}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{D\times D}$ are the weight matrices. $\sqrt{D}$ aims to avoid the overly large values, especially when the embedding size is very large. Notably, due to the nature of sequences, a sequential recommender should leverage solely the previous $t-1$ items when inferring the t-th item. Nevertheless, the $(t-1)$-th output of Eq. (9) consists of all input information. As such, following previous studies [11,14], we also employ the causality mask to prevent information leaks, *i.e.*, adjusting the attention by prohibiting all links between $\boldsymbol{Q}_i$ and $\boldsymbol{K}_j$ $(j > i)$.

3) *Point-wise feed-forward layer*: although the time-sensitive self-attention layer could adaptively aggregate the recently interacted items' embeddings, it is non-trivial to capture the complicated item transition patterns due to the linearity. As such, we further apply a point-wise feed-forward layer to explore interactions between different factors:

$$\boldsymbol{F} = \mathrm{FFN}(\boldsymbol{S}) = \mathrm{ReLU}(\boldsymbol{S}\boldsymbol{W}' + \boldsymbol{b}')\boldsymbol{W}'' + \boldsymbol{b}'', \tag{10}$$

where $\boldsymbol{W}', \boldsymbol{W}'' \in \mathbb{R}^{D\times D}$ and $\boldsymbol{b}', \boldsymbol{b}'' \in \mathbb{R}^D$ are the learnable model parameters. In order to avoid overfitting and vanishing gradients, we follow [11,14] and also apply the layer normalization and dropout regularization techniques. After performing Eq. (10), $\boldsymbol{F}$ essentially aggregates the recently interacted items' embeddings. Nonetheless, it may be beneficial to capture longer-range dependency patterns with a sequence by stacking multiple self-attention layers. In particular, we implement the above two layers with the b times via the following rules:

$$\boldsymbol{S}^{(b)} = \mathrm{SA}(\boldsymbol{F}^{(b-1)}), \quad \boldsymbol{F}^{(b)} = \mathrm{FFN}(\boldsymbol{S}^{(b)}), \tag{11}$$

where $\boldsymbol{S}^{(1)} = \boldsymbol{S}$ and $\boldsymbol{F}^{(1)} = \boldsymbol{F}$. After performing the b times, we adopt the output vector $\boldsymbol{y}_u(t) = \boldsymbol{f}^{(b)}_L$ to model short-term user interest.

3.5. Model prediction

After implementing the aforementioned two modules, we get long-term interest $\boldsymbol{e}^*_u$ that encodes collaborative signals and short-term interest $\boldsymbol{y}_u(t)$ that capture sequential patterns. Naturally, a question arises: how to settle the semantic gap between two information and integrate them into an organic whole? Intuitively, concatenation and summation strategies can be immediately applied to this issue, and they are described as follows.

1) Concatenation: To extract high-level semantic features, such operation concatenates both kinds of information and transforms them to a D-dimensional representation by a weight matrix $\boldsymbol{W}_c \in \mathcal{R}^{D\times 2D}$. Formally, this strategy can be written as follows:

$$\boldsymbol{z}_u(t) = \boldsymbol{W}_c \begin{bmatrix} \boldsymbol{e}^*_u \\ \boldsymbol{y}_u(t) \end{bmatrix}. \tag{12}$$

2) Summation: To aggregate both kinds of embeddings, this strategy constructs a D-dimensional vector by an additive operation. Such strategy could be formulated as follows:

$$\boldsymbol{z}_u(t) = \boldsymbol{e}_u^* + \boldsymbol{y}_u(t). \tag{13}$$

Although such two strategies are simple and straightforward, they have two defects. First, at a certain time, they fail to distinguish the significance of each part for different users. For instance, at inferring the next interesting item, long-term interest $\boldsymbol{e}_u^*$ plays a key role for a stubborn man, while short-term interest $\boldsymbol{y}_u(t)$ is more important for a fickle girl. Second, even for the same user, the contribution ratio of each part should vary significantly at different moments. In summary, the above two manners fail to adaptively determine the effect of each part, leading to the suboptimal recommendation results. As such, inspired by the design of long short-term memory [8], we design a personalized gating strategy to adaptively assign weights for these two ingredients. Formally, we could derive the representation of user u at time step t as follows:

$$\boldsymbol{z}_u(t) = \alpha \boldsymbol{e}_u^* + (1-\alpha)\boldsymbol{y}_u(t), \quad \alpha = \sigma(\boldsymbol{W}_g[\boldsymbol{e}_u^* || \boldsymbol{y}_u(t)]), \tag{14}$$

where $\sigma(\cdot)$ is a sigmoid function, $||$ represents the concatenation operation, and $\boldsymbol{W}_g$ is the learnable parameter matrix. Taking the above representation as input, we can compute the likelihood of item i being the next interesting one as follows:

$$\hat{r}_{u,t,i} = \boldsymbol{z}_u(t)\boldsymbol{e}_i^{*\top}. \tag{15}$$

3.6. Model optimization

Based on the above predictor, we utilize the binary cross entropy (BCE) loss function, a prevalent optimization criterion for recommender systems, to learn the model parameters. Mathematically, GCTN could be optimized by calculating the following likelihood:

$$\arg\max_{\Theta} \prod_{u\in\mathcal{U}} \prod_{t=L+1}^{|\mathcal{I}^u|} \sigma\left(\hat{r}_{u,t,i}\right) \prod_{j\neq i} \left(1-\sigma\left(\hat{r}_{u,t,j}\right)\right) \Pr(\Theta). \tag{16}$$

By minimizing the negative logarithm of BCE, Eq. (16) is rewritten as follows:

$$\mathcal{L} = -\sum_{u\in\mathcal{U}} \sum_{t=L+1}^{|\mathcal{I}^u|} \log\left(\sigma\left(\hat{r}_{u,t,i}\right)\right) + \sum_{j\in\mathcal{I}\setminus S^u} \log\left(1-\sigma\left(\hat{r}_{u,t,j}\right)\right) + \lambda\|\Theta\|^2, \tag{17}$$

where Θ stands for all the trainable parameters and λ is a regularization coefficient. The negative instance j is randomly chosen from all the unobserved items. Other more complicated sampling strategies (*e.g.*, dynamic sampling [38] and adversarial sampling [29]) may facilitate the model accuracy and are left in future work. Following LightGCN [4] and HGN [22], we utilize *Adam* as the optimizer and describe the optimization procedure in Algorithm 1.

Algorithm 1: Model optimization for GCTN.

```
   Input: User-item-category tripartite graph $G$, the action sequence $S^u$, the layer $K$, the block $B$, the order $L$, the hyper-parameter $\lambda$;
   Output: $\Theta_1 = \{\boldsymbol{E}, \boldsymbol{M}, \boldsymbol{P}\}$,
   $\Theta_2 = \{\boldsymbol{H}, \boldsymbol{w}_x, \boldsymbol{b}_x, \boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}', \boldsymbol{W}'', \boldsymbol{b}', \boldsymbol{b}'', \boldsymbol{W}_g\}$;
1  Initialize $\Theta_1$ and $\Theta_2$;
2  while stopping criteria is not met do
3      for each layer $k \in \{1,2,\dots,K\}$ do
4          calculate $\boldsymbol{m}_i^{(k)}$ and $\boldsymbol{e}_u^{(k)}$ w.r.t. Eq. (3) and Eq.( (4), respectively;
5      obtain $\boldsymbol{m}_u^*$ and $\boldsymbol{e}_u^*$ w.r.t. Eq. (5);
6      for each user $u \in \mathcal{U}$ do
7          $\boldsymbol{x}_{t-l}^u = \tanh(\boldsymbol{w}_x log(T_t^u - T_{t-l}^u) + \boldsymbol{b}_x)$;
8      for each block $b \in \{1,2,\dots,B\}$ do
9          obtain $\boldsymbol{S}^{(b)}$ w.r.t. Eq. (9) and Eq. (11);
10         obtain $\boldsymbol{F}^{(b)}$ w.r.t. Eq. (10) and Eq. (11);
11     obtain $\hat{r}_{u,t,i}$ w.r.t. Eq. (15);
12     calculate $\mathcal{L}$ w.r.t. Eq. (17);
13     update parameters w.r.t. $\Theta_1$ and $\Theta_2$ with Adam
14 return $\Theta$
```

1) *Model size*: From Algorithm 1, it observes that our model parameters compose of two portions. The first portion Θ_1 derives from the initial embeddings of users and items (*i.e.*, $\boldsymbol{E}, \boldsymbol{M}, \boldsymbol{P}$), which is proportional to the scale of users and items (*i.e.*, $(|\mathcal{U}| + 2|\mathcal{I}|)D$). For the second portion Θ_2, they are completely independent of the scale of users and items (*i.e.*, $(|\mathcal{C}| + 6)D + 5D^2$). Theoretically, the model size of our solution is comparable to the recent recommender HGN (*i.e.*, $(|\mathcal{U}| + 2|\mathcal{I}| + L + 1)D + |\mathcal{I}| + 2D^2$), and much smaller than the pioneer solution FPMC (*i.e.*, $(|\mathcal{U}| + 3|\mathcal{I}|)D$). Taking the CDs&Vinyl dataset as an example (*i.e.*, $|\mathcal{U}|$ =26872, $|\mathcal{I}| = 64375$ and $|\mathcal{C}| = 466$ in Table 2), when D is fixed to 64, HGN and FPMC have 9.57M and 13.42M parameters respectively, meanwhile GCTN

Table 2
Statistics of four benchmarks.

Property	Beauty	Movies&TV	CDs&Vinyl	ML-1M
#Users($\|\mathcal{U}\|$)	5,123	40,842	26,872	6,040
#Items($\|\mathcal{I}\|$)	11,654	49,965	64,375	3,706
#Categories($\|\mathcal{C}\|$)	226	29	466	18
#Actions($\|\mathcal{R}\|$)	91,824	1,174,697	792,115	1,000,209
$\|\mathcal{R}\|/\|\mathcal{U}\|$	17.92	28.76	29.47	165.59
$\|\mathcal{R}\|/\|\mathcal{I}\|$	7.87	23.51	12.30	269.88
Sparsity	99.84%	99.94%	99.95%	95.53%

has 9.54M parameters. In a nutshell, the memory cost of the second portion is perfectly acceptable, considering $|\mathcal{C}|$ and D are much smaller than $|\mathcal{U}|$ and $|\mathcal{I}|$.

2) *Time complexity*: The time complexity of GCTN derives from the category-aware graph propagation module and the time-sensitive self-attention mechanism. The time cost of the former is similar to LightGCN [4], and producing the final embeddings $\boldsymbol{e}_u^*$ and $\boldsymbol{m}_i^*$ has computation complexity $O(DK|\mathcal{E}|)$. For the second part, multiple self-attention operations and point-wise feed-forward layers are involved, for which the computational complexity is $O(L^2D + LD^2)$. Overall, the total time cost of learning GCTN per epoch is $O\left(DK|\mathcal{E}| + L^2D + LD^2\right)$. We will provide the efficiency analysis of GCTN and other baselines in Section 4.10.

4. Experiments

In this section, we perform experiments to justify the superiority of GCTN under different recommendation scenarios. The empirical results aim to answer four questions:

RQ1 Does GCTN provide better results than state-of-the-arts including CF-based/sequential-aware recommenders?
RQ2 How is the robustness of our method?
RQ3 What is the influence of different ingredients in the GCTN framework?
RQ4 How does GCTN perform with different fusion strategies?
RQ5 How do the key hyper-parameters affect the GCTN's results?
RQ6 What is the scalability of GCTN compared to other baselines?

4.1. Description of datasets

We perform experiments under different application scenarios: (1) **Amazon.** Amazon is the world's most popular e-commerce platform, where customers could purchase various kinds of products and provide numerical ratings for them. Due to the space reason, we select three representative categories, *i.e.*, "Beauty", "Movies&TV" and "CDs&Vinyl" as the experimental datasets. (2) **ML-1M.** This dataset is collected by GroupLens, which consists of a large number of user ratings towards movies in 2000.

Following previous studies [28,36,47], we treat all user ratings as positive samples and the rest unobserved items as negative samples. For each user, we ensure that his action records are no less than 10. Table 2 displays the basic properties of the preprocessing datasets. To further understand the data characteristic, we display data distribution *w.r.t.* user and item in Fig. 3 (log–log plot). Clearly, item popularity exhibits a long-tail distribution, *i.e.*, a significant fraction of items has very few action records, meanwhile massive interactions are dominated by a small fraction of items. Analogously, we can observe that user actions also follow a long-tail distribution. Such phenomenon makes a recommender difficult to attain informative representations for inactive users and tail items, and motivates us to alleviate this challenge via item category information.

4.2. Evaluation metrics

We utilize the *fold-out* evaluation setting, *i.e.*, for a user action sequence, we hold 70% as the training set, the next 10% of action records as the validation set, and the rest actions for testing. Four popular ranking-oriented metrics [34,39,46] are utilized to verify the superiority of GCTN, namely Precision, NDCG, Recall and MAP. Formally, we provide the detailed definition as follows:

$$\text{Precision@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{\left|l_{\text{rec}}^u \bigcap l_{\text{tes}}^u\right|}{\text{N}},\ \text{NDCG@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{\sum_{a=1}^{\text{N}}\frac{2^{\delta u(a)}-1}{\log_2(a+1)}}{\sum_{a=1}^{\min\left\{\text{N},\left|l_{\text{tes}}^u\right|\right\}}\frac{1}{\log_2(a+1)}},$$
$$\text{Recall@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{\left|l_{\text{rec}}^u \bigcap l_{\text{tes}}^u\right|}{\left|l_{\text{tes}}^u\right|},\ \text{MAP@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{\sum_{a=1}^{\text{N}} P_u(a)\times\delta_u(a)}{\min\left\{\text{N},\left|l_{\text{tes}}^u\right|\right\}}, \tag{18}$$

where l_{rec}^u represents the top-N recommendation results and l_{tes}^u denotes the really interacted items. $P_u(a)$ indicates the precision of the a-th position for user u, and $\delta_u(a)$ is a binary indicator function that is 1 if an item at rank a is consumed, and 0 otherwise.

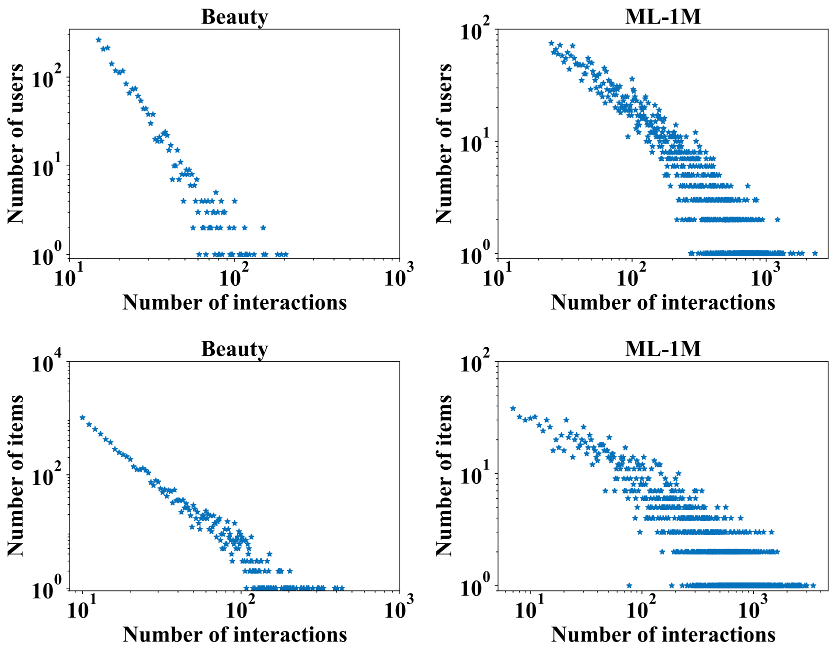

Fig. 3. Data distribution *w.r.t.* user and item.

4.3. Baselines

To justify the effectiveness of our proposed GCTN framework, we choose seven representative methods, consisting of two general recommenders and five sequential recommenders:

- **BPRMF:** It is the simplest CF-based model, which employs MF to produce user and item embeddings and adopts the Bayesian personalized ranking criteria to optimize the relative ranking of all the unobserved items [24].
- **FPMC:** It is the pioneer solution for sequential recommendation, which extends BPRMF by adopting a first-order MC to explicitly capture short-term user interest [25].
- **Caser:** It is a CNN-based sequential recommender, which models high-order MCs by using horizontal and vertical convolution filters [30].
- **LightGCN:** This is the classical GNN-based method, which abandons nonlinear feature transformation in GNNs and devises a linear graph convolution block [4].
- **LightGCL:** This is the recently introduced self-supervised learning method, which adopts the singular value decomposition to reconstruct user-item bipartite graph and introduces a lightweight contrastive learning framework for learning user and item representations [1].
- **SASRec:** Instead of employing complicated convolutional module, this method applies a positional-aware self-attention mechanism and point-wise feed-forward network to capture long-range dependency patterns between items [11].
- **MLP4Rec:** It is a pure MLP-based framework, which develops a tri-directional mixing MLP block to capture complicated correlations [16].
- **TiSASRec:** This baseline is closest to our solution, which presents a time interval aware method for next-item recommendation; it leverages the time interval information between adjacent items and introduces a time interval self-attention layer to dynamically assign weights for different items [14].
- **GES-SASRec:** It is the closest to our work [49], which integrates GNN and SASRec into a unified solution; it first performs GCN on a hybrid item graph to produce smoothed item representations and then enhances SASRec with these representations.
- **HGN:** This is the recently proposed sequential recommender based on a hierarchical gating module, where personalized feature gating learns union-level sequential patterns, while instance gating captures individual-level sequential patterns [22].

4.4. Hyper-parameter settings

For two classical recommenders (*i.e.*, BPRMF and FPMC), we implement them by $NeuRec$[1] — an opensource library. For the rest baselines, we deploy them according to the released source codes by the corresponding author. For all methods, the learning rate is tuned in $\{5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$ and the regularization coefficient λ is selected from {1e-5,1e-4,1e-3,1e-2,0.1,1,10}. For HGN

[1] https://github.com/wubinzzu/NeuRec.

Table 3
The performance of all approaches with $N = 10$ (higher is better).

Dataset	Method	$D=16$				$D=64$			
		Precision	Recall	NDCG	MAP	Precision	Recall	NDCG	MAP
Beauty	BPRMF	1.0716	3.0621	2.0706	1.0939	1.2227	3.6618	2.4651	1.3244
	FPMC	1.7841	4.3419	3.3021	1.7689	2.3306	6.1013	4.7928	2.6493
	Caser	1.7606	4.2325	3.2812	1.7692	2.0945	5.1591	3.9907	2.3803
	LightGCN	1.2001	3.5041	2.3754	1.2752	1.3663	4.2374	2.9216	1.6156
	LightGCL	1.2844	3.8272	2.6815	1.5350	1.4198	4.4488	3.0516	1.7189
	SASRec	1.8265	4.3649	3.3084	1.7898	2.3387	6.1221	4.6192	2.6669
	MLP4Rec	1.8491	4.4151	3.3565	1.8028	2.3651	6.2183	4.7957	2.7019
	TiSASRec	1.8819	4.5337	3.4146	1.8229	2.4153	6.3163	4.9101	2.7368
	GES-SASRec	2.0185*	5.1968*	3.9245*	2.1402*	2.5179*	6.4295*	4.9557*	2.8036
	HGN	2.0124	4.9559	3.6841	2.0593	2.4399	6.2446	4.8881	2.8294*
	GCTN	**2.1238**	**5.4789**	**4.1411**	**2.3517**	**2.6917**	**7.0901**	**5.5084**	**3.1629**
	Improve	5.22%	5.43%	5.52%	9.88%	6.91%	10.27%	11.15%	11.79%
Movies &TV	BPRMF	0.7007	1.7515	1.2836	0.6028	0.8329	2.1369	1.5841	0.7759
	FPMC	0.7789	1.9086	1.5278	0.7792	1.0254	2.6889	2.0441	1.0451
	Caser	0.7918	1.9091	1.5488	0.7811	1.0783	2.6897	2.0506	1.0527
	LightGCN	0.9475	2.3094	1.7649	0.8507	1.2568	3.2522	2.4976	1.2869
	LightGCL	1.0745	2.3978	1.8923	0.9189	1.3038	3.4891	2.5787	1.3257
	SASRec	1.0499	2.3269	1.8662	0.9057	1.3327	3.2934	2.5442	1.3039
	MLP4Rec	1.0596	2.3319	1.8697	0.9103	1.3346	3.3481	2.5730	1.3168
	TiSASRec	1.0748	2.3586	1.8891	0.9183	1.3833	3.4327	2.6068	1.3339
	GES-SASRec	1.1301*	2.7215*	2.0344*	1.0706*	1.5832*	3.8612*	3.1948*	1.5057*
	HGN	1.0916	2.4905	1.8931	0.9239	1.5237	3.5621	2.7854	1.4226
	GCTN	**1.1912**	**2.8629**	**2.2056**	**1.0867**	**1.6640**	**4.0475**	**3.2791**	**1.6858**
	Improve	5.41%	5.20%	8.42%	1.51%	5.05%	4.82%	2.64%	11.96%
CDs&Vinyl	BPRMF	0.9634	2.2519	1.7799	0.8691	1.2396	3.0235	2.3617	1.1833
	FPMC	0.9671	2.2973	1.7973	0.8953	1.5362	3.9597	3.3311	1.6815
	Caser	1.0304	2.3040	1.8715	0.9062	1.2974	3.1312	2.5012	1.2386
	LightGCN	1.0437	2.3114	1.9328	0.9134	1.4567	3.6872	3.2812	1.5189
	LightGCL	1.1953	2.9212	2.2667	1.1493	1.5708	3.9904	3.3221	1.6316
	SASRec	1.0606	2.3387	1.9481	0.9381	1.6488	4.0363	3.3359	1.7314
	MLP4Rec	1.0715	2.3647	1.9612	0.9484	1.6613	4.0936	3.3906	1.7647
	TiSASRec	1.0940	2.4059	2.0182	0.9712	1.7171	4.1567	3.4889	1.8557
	GES-SASRec	1.4187*	3.3038*	2.4177*	1.3039*	2.0330*	4.7574*	3.8706*	1.9816*
	HGN	1.2969	3.0014	2.3587	1.1937	1.8557	4.2532	3.5358	1.9771
	GCTN	**1.4506**	**3.4076**	**2.7291**	**1.3764**	**2.1152**	**5.1627**	**4.1733**	**2.2425**
	Improve	2.25%	3.14%	12.88%	5.56%	4.04%	4.82%	7.82%	13.16%
ML-1M	BPRMF	0.1195	0.0608	0.1331	0.0655	0.1203	0.0615	0.1341	0.0661
	FPMC	0.1649	0.0919	0.1862	0.0949	0.1657	0.0931	0.1871	0.0963
	Caser	0.2005	0.1111	0.2243	0.1206	0.2098	0.1228	0.2361	0.1298
	LightGCN	0.1257	0.0618	0.1381	0.0695	0.1278	0.0628	0.1419	0.0722
	LightGCL	0.1286	0.0686	0.1457	0.0736	0.1324	0.0699	0.1463	0.0748
	SASRec	0.2001	0.1069	0.2202	0.1192	0.2052	0.1221	0.2318	0.1254
	MLP4Rec	0.2031	0.1077	0.2229	0.1213	0.2091	0.1235	0.2362	0.1274
	TiSASRec	0.2042	0.1121	0.2270	0.1228	0.2121	0.1245	0.2405	0.1299
	GES-SASRec	0.2083*	0.1151*	0.2329*	0.1282*	0.2247*	0.1294*	0.2526*	0.1403
	HGN	0.2056	0.1139	0.2318	0.1271	0.2224	0.1269	0.2511	0.1415*
	GCTN	**0.2131**	**0.1167**	**0.2395**	**0.1335**	**0.2434**	**0.1435**	**0.2771**	**0.1595**
	Improve	2.31%	1.41%	2.83%	4.13%	8.32%	10.90%	9.74%	12.72%

and Caser, the length of a subsequence L is chosen from {2,3,4,5,6,7}. For Caser, we adjust the height h of the horizontal convolution filter in $\{1,\cdots,L\}$, and select the number of convolutional filters in {4,8,16,32,64}. For SASRec and TiSASRec, we select the number of self-attention blocks from {0,1,2,3} and the dropout rate from {0.2,0.3,0.4,0.5,0.6}. For TiSASRec, we tune the time span in {2,4,8,16,32,64,128,256,512}. For LightGCL, the temperature η and the regularization weight λ_1 are tuned from {0.3,0.5,1,3,10} and {1e-5,1e-6,1e-7}, respectively. For GES-SASRec, we search the number of graph convolutional layers from {1,2,3,4,5}. For GCTN, we also implement it with NeuRec and optimize it with *Adam* optimizer. The number of graph propagation layers K is selected from {1,2,3,4,5,6,7}. The impact of different L values will be discussed below.

4.5. Comparison with state-of-the-arts (RQ1)

We first provide the overall performance comparison of GCTN and all baselines. Table 3 displays the performance with different embedding sizes $D \in \{16, 64\}$. Furthermore, we conduct the ANOVA test [31] to measure the significance of the performance gains between GCTN and other baselines. The ANOVA results indicate these gains are non-contingent and statistically significant (*i.e.*,

$p < 0.001$), but are removed due to the space reason. Note that the results on three Amazon datasets have been multiplied by 100 for convenience. From the table, we have seven key conclusions:

- As D increases ($16 \rightarrow 64$), the results of all methods tend to be better. This indicates that larger dimensional embeddings could capture more characteristics of users and items, which is helpful for enhancing the recommendation quality.
- The classical model BPRMF obtains the worst performance, which illustrates that the one-hot encoding technique is non-trivial to provide sufficient information for user and item embedding learning. Furthermore, LightGCN consistently surpasses BPRMF in all cases, demonstrating that exploiting high-order neighbor information is beneficial for guiding the embedding learning of nodes in user-item bipartite graph.
- Compared with LightGCN, LightGCL achieves consistently better results. It demonstrates that SVD-guided graph contrastive learning has the ability to alleviate data sparsity and popularity bias issues.
- With the same optimization criterion, FPMC significantly outperforms than BPRMF under different recommendation scenarios. This indicates the importance of explicitly capturing sequential patterns by exploring first-order MC.
- Caser generally achieves better results than FPMC in most cases, demonstrating the benefits of modeling high-order MCs for sequential recommendation. Alternatively, instead of the complicated convolutional operation, SASRec can obtain better performance on Amazon datasets. Presumably, CNN-based methods require more parameters to capture item transition patterns, whereas the well-designed self-attention mechanism is more effective in high-sparsity applications.
- Compared to SASRec, TiSASRec shows consistent performance gains on four benchmarks. This indicates that leveraging the time interval information between adjacent items is more important than capturing the sequential order information in $\mathcal{S}^u$.
- Instead of the self-attention mechanism in SASRec, MLP4Rec adopts a tri-directional MLP-mixer block to capture sequential patterns and obtains slight performance gains. In contrast, GES-SASRec achieves substantial improvements over SASRec. This shows that by performing GCN on a hybrid item graph, GES-SASRec is capable of generating smoothed item embeddings for sequential recommenders.
- Compared with SASRec and TiSASRec, HGN obtains considerably better performance in all cases. Such significant gains may be attributed to the hierarchical gating mechanism, which can capture sequential patterns both at feature-level and instance-level.
- As expected, GCTN achieves the best results on four datasets. For instance, when we fix $D = 64$, GCTN achieves significant performance gains over the "*" results *w.r.t.* MAP@10 by 11.79%, 11.96%, 13.26%, and 12.72% in Beauty, Movies&TV, CDs&Vinyl, and ML-1M, respectively. Such considerable improvements stem from three aspects: (1) by implementing the category-aware graph propagation module, our method could greatly alleviate the long-tail issue when learning the embeddings of users and items; (2) the time-sensitive self-attention mechanism captures item transition patterns in a more reasonable fashion, which is highly sensitive to the time interval between each candidate item in $\mathcal{S}^u$ and the target one; (3) given the special context, the devised personalized gating strategy could adaptively determine the contribution of each part for predicting future user decision makings.

Though Table 3 investigates the top-10 recommendations, we are still uncertain how GCTN performs under different lengths of the top-N list. Towards this end, Fig. 4 shows the performance comparison of GCTN and several representative baselines *w.r.t.* Precision@N and NDCG@N (*i.e.*, N $\in \{5, 10, 15, 20\}$). The results in rest metrics are omitted due to the same observations. We interpret the results with the following three findings:

- GCTN and other competitors perform the similar trend on four datasets. Concretely, when increasing the recommendation length, the Precision values steadily decline, meanwhile the NDCG values gradually increase.
- SASRec consistently outperforms BPRMF in all cases. This indicates that capturing short-term user interest is very important for accurately predicting user preference towards an item. On the top of SASRec, GES-SASRec achieves further improvements, which shows the power of graph convolutions at generating smoothed item embeddings.
- Our method consistently achieves significant performance gains over other competitors. This again verifies the advantage of adopting the category-aware graph propagation module to learn high-quality user and item embeddings, the effectiveness of using the time-sensitive self-attention mechanism to exploit the time interval information, and the rationality of applying the personalized gating strategy to dynamically integrate long-term and short-term user interests.

4.6. Robustness analysis (RQ2)

In this section, the robustness of GCTN will be investigated from two aspects.

(1) *Performance comparison under different user groups*: Table 3 and Fig. 4 have displayed the overall comparison between GCTN and other competitors. Nonetheless, how is the robustness of our solution under different sparsity settings? To answer this question, we group all users into four parts, and each one has the same number of user-item interactions. Herein we concentrate on a sparser dataset (*i.e.*, Beauty) and a denser dataset (*i.e.*, ML-1M). Taking Beauty as an example, the sequence length per user of four groups is $\{(0, 12], (12, 18], (18, 33], (33, 204]\}$, wherein $(12, 18]$ indicates that the sequence length of each user in this group is greater than 12 yet smaller than 18. Fig. 5 shows the results *w.r.t.* Precision@10 and NDCG@10 on different user groups. We interpret these results with the following observations:

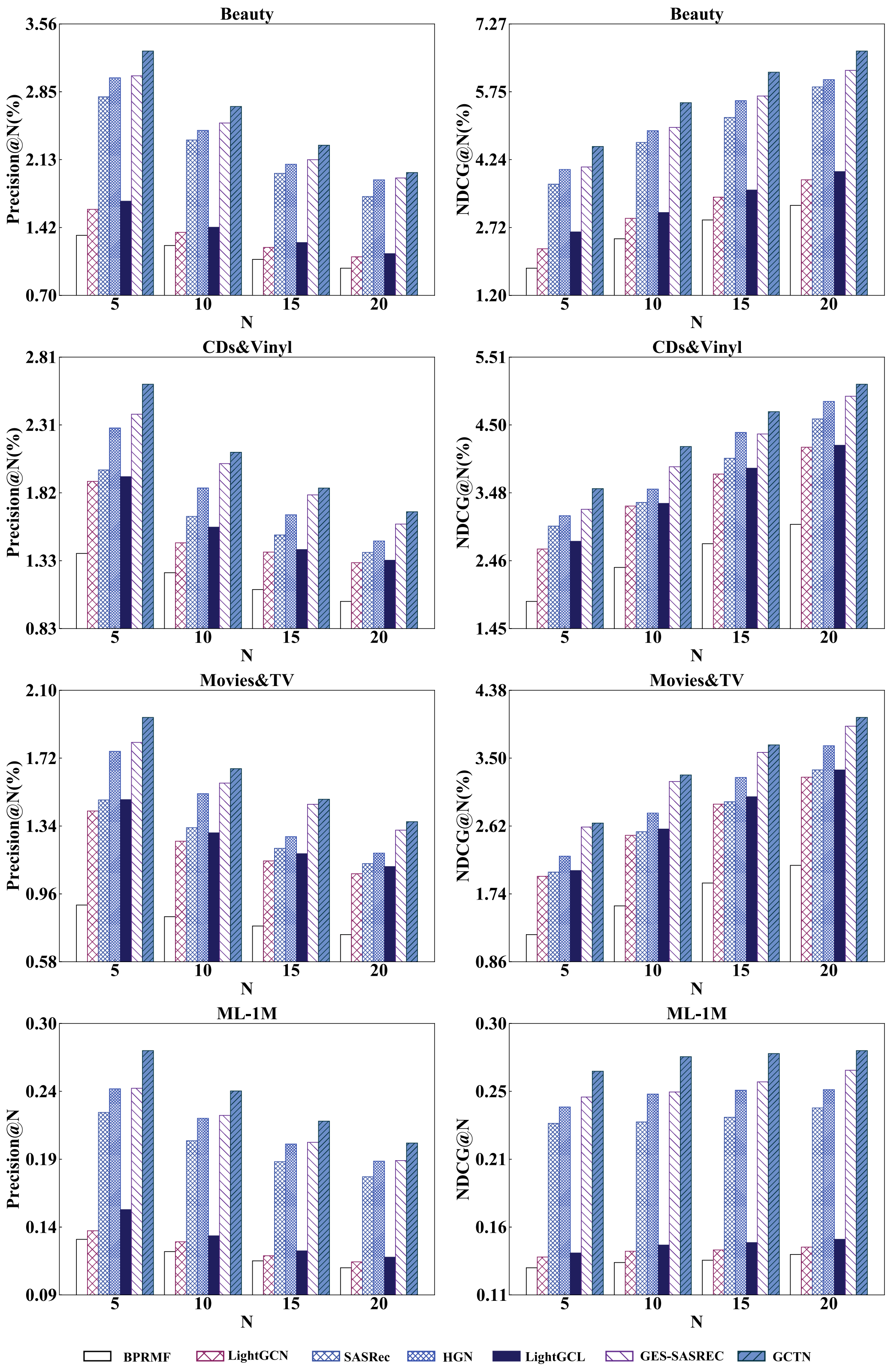

Fig. 4. The performance comparison with different values of N.

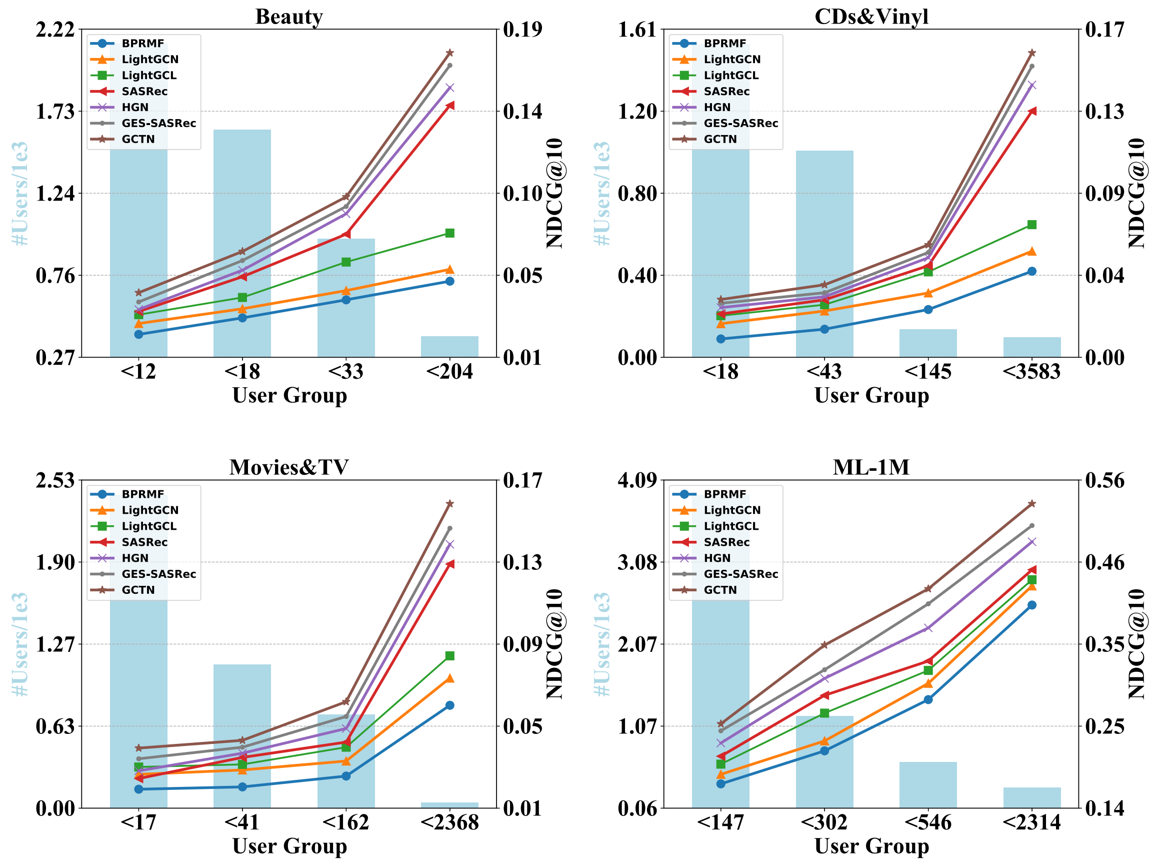

Fig. 5. Performance comparison under different interaction sparsity levels on representative datasets.

- With the same scale of model parameters and the same learner, LightGCN consistently achieves better results over BPRMF in all groups. This validates that explicitly exploring collaborative signals is beneficial to refine the user embedding learning.
- When users have more behavior data, the recommendation quality tends to become better as more observed instances present more evidence for user preference learning. Moreover, GCTN still obtains the best results over other competitors. This justifies the robustness of our solution across different types of users.

(2) Long-tail recommendation: In large-scale industrial scenarios, the ability of alleviating the long-tail issue decides the practicality of a recommender system. In light of this, we conduct a micro analysis of the recommendation results on the Beauty and ML-1M datasets. To be specific, we first sort all items based on their popularity. Afterwards, all items are put into 10 groups and each of which has the identical action number. Note that the more the number of items is, the larger the group ID is. In this way, the items in GroupID 1 are the most popular and the ones in GroupID 10 are the most unpopular (*i.e.*, long-tailed). For a recommendation method, we can compute the recommendation ratio (RR) by:

$$RR(g,m) = \frac{\sum_{i \in GroupID\ \ g} RC(i,m)}{\sum_{i \in GroupID\ \ 1 \to 10} RC(i,m)}, \tag{19}$$

where $RC(i,m)$ is a counter that a recommender m recommends item i in the top-10 list. Ideally, the probability that each group is recommended should be identical (*i.e.*, $RR(g,m) = 0.1$). Fig. 6 displays the recommendation ratio of GCTN and two representative methods. We have two major observations:

- From GroupID 1 to GroupID 10, the RR values of LightGCN decrease quickly. Compared with the ideal RRs, the gap between probabilities to recommend GroupID 1 and GroupID 10 becomes larger. By capturing the dynamics of sequential patterns, GES-SASRec alleviates bias amplification to a certain extent.
- Compared with other baselines, the RR line of GCTN is flatter and the nearest to the ideal. Moreover, for the items in GroupID 10, our solution has a larger RR than GES-SASRec. This demonstrates that our solution could provide the relatively fair recommendation list.

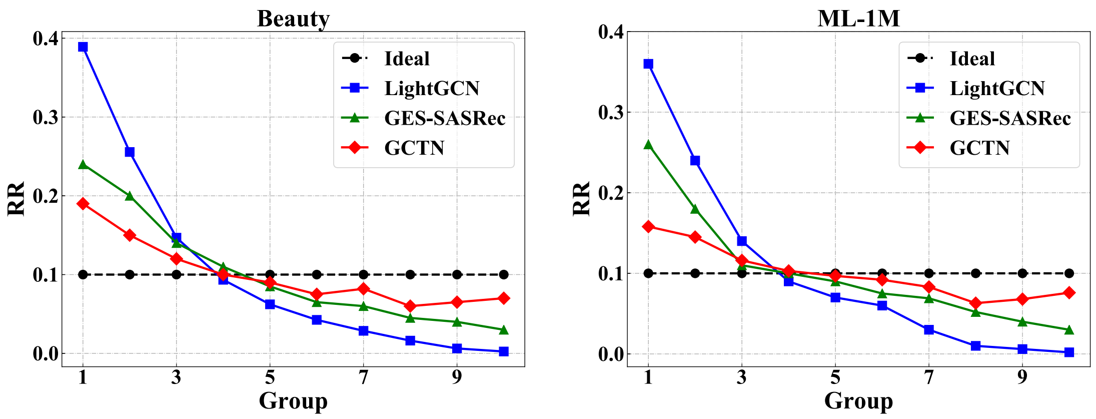

Fig. 6. Recommendation ratio comparison over different item groups.

Table 4
Ablation study.

Dataset	Metric@10	BE	BE+H	BE+C	BE+A	BE+S	BE+C+S	GCTN
Beauty	Precision	1.2277	1.5789	1.6630	2.4179	2.4343	2.6191	**2.6917**
	NDCG	2.4649	3.1230	3.1615	4.9321	4.9745	5.2387	**5.5084**
Movies&TV	Precision	0.8332	1.2871	1.3238	1.4219	1.6341	1.6489	**1.6641**
	NDCG	1.5841	2.6281	2.6603	2.8715	3.1633	3.2356	**3.2791**
CDs&Vinyl	Precision	1.2396	1.5329	1.6217	1.7319	1.8209	2.0553	**2.1152**
	NDCG	2.3617	3.3001	3.3439	3.5021	3.6278	4.0284	**4.1733**
ML-1M	Precision	0.1195	0.1458	0.1660	0.2173	0.2266	0.2426	**0.2434**
	NDCG	0.1331	0.1665	0.1872	0.2491	0.2568	0.2764	**0.2771**

4.7. Ablation study (RQ3)

To study how different ingredients impact the recommendation quality, we perform an ablation study in Table 4. Specifically, BE represents the most concise MF model with the popular binary cross entropy loss. Instead of the MF module, BE+H regards item category as a single node and applies relational GCN [27] on the user-item-category tripartite graph. In contrast to BE+H, BE+C views item category as context information and adopts the category-aware graph propagation module to generate the node embeddings. Same as TiSASRec [14] and MTAM [9], BE+A explores the time interval between two adjacent items and employs the standard self-attention mechanism to model short-term user interest. Compared to BE+A, BE+S utilizes the time-sensitive self-attention mechanism to capture item transition patterns. BE+C+S represents the tight integration between our designed two components. Finally, we provide the complete GCTN method to demonstrate the necessity of integrating different ingredients. From Table 4, we make four key conclusions:

- As the simplest variant, BE obtains the worst results in all cases. This again verifies that depending only on the directly observed data is non-trivial to learn high-quality node embeddings especially for inactive users and tail items.
- Through jointly analyzing Table 3 and 4, BE+H consistently surpasses LightGCN, which shows leveraging item categorical information is beneficial for improving the node embedding learning. Moreover, the variant BE+C obtains the further improvements over BE+H, demonstrating that it is more reasonable to integrate item category into the embedding propagation rule.
- Compared with the SASRec's results in Table 3, BE+A achieves the better performance, which verifies that exploring the time interval information in a user's action sequence is helpful for modeling short-term user interest. Furthermore, BE+S significantly outperforms SASRec and BE+A. These remarkable performance gains validate the necessity and rationality of explicitly capturing the time interval between each candidate item in S^u and the target timestamp for predicting the next action.
- Compared with other variants, BE+C+S shows further performance gains, indicating that our devised two modules are complementary for attaining satisfactory recommendation results.
- As expected, GCTN obtains substantial performance gains over all variants, which indicates the necessity of dynamically integrating long-term and short-term user interests. These results also demonstrate the superiority of our solution.

4.8. Effect of fusion strategy (RQ4)

To investigate how different fusion strategies affect GCTN's performance, we conduct this experiment with three variants of GCTN. Table 5 shows the empirical results of these variants and we have two observations.

Table 5
Performance comparison with different fusion strategies ($D = 64$ and $N = 10$).

Dataset	C		S		G	
	Precision	NDCG	Precision	NDCG	Precision	NDCG
Beauty(%)	2.5513	5.1862	2.6191	5.2387	**2.6917**	**5.5084**
Movie&TV(%)	1.6279	3.1513	1.6489	3.2356	**1.6640**	**3.2791**
CD&Vinyl(%)	2.0437	3.9386	2.0553	4.0284	**2.1152**	**4.1733**
ML-1M	0.2309	0.2635	0.2426	0.2764	**0.2434**	**0.2771**

"C" stands for concatenation, "S" represents summation, and "G" denotes the personalized gating strategy.

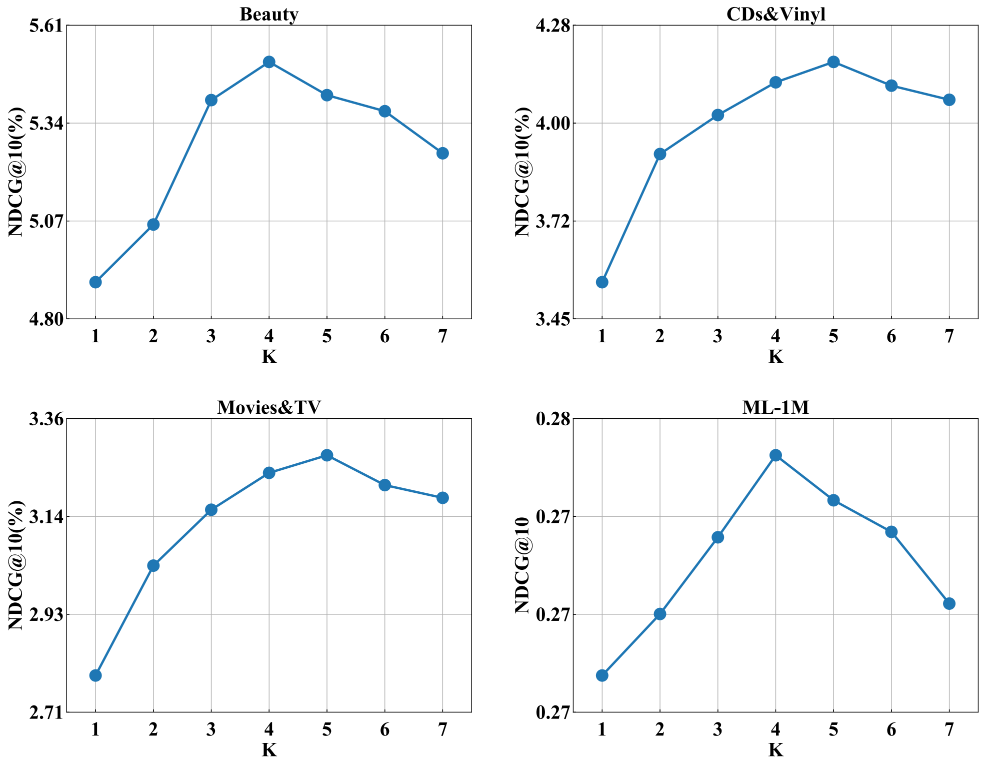

Fig. 7. NDCG@10 (y-axis) vs. the layer K (x-axis).

- GCTN with the concatenation strategy underperforms the variant with summation. Probably, it is non-trivial to distill the informative embedding by solely a weight matrix $\boldsymbol{W}_c$.
- As expected, GCTN with the personalized gating strategy significantly outperforms other variants. These substantial performance gains justify that it is necessary to dynamically determine the importance of each module at different time steps.

4.9. *Hyper-parameter sensitivity (RQ5)*

In this section, we investigate the impact of two vital hyper-parameters for GCTN.

1) *Effect of layer number K*: We select K from {1,2,3,4,5,6,7} while keeping the rest optimal hyper-parameters fixed. Fig. 7 shows the NDCG@10 values on the test data and we have two observations:

- As K increases, the performance of GCTN significantly goes better especially in the sparser datasets. For example, GCTN-2 achieves substantial performance gains over GCTN-1. Probably, it is helpful to mitigate the data sparsity issue by explicitly exploiting second-order neighbor information in user-item-category tripartite graph.
- For each dataset, more graph propagation layers do not always achieve better performance. GCTN obtains its best results when we properly select the number. The performance gets worse with a larger number due to over smoothing.

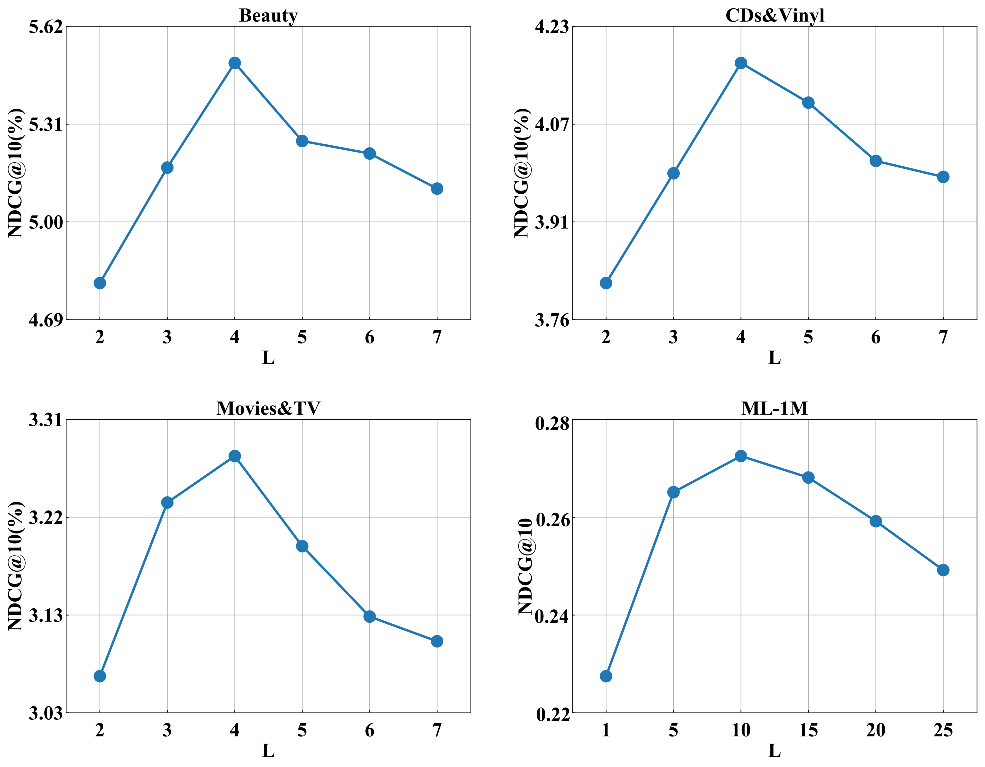

Fig. 8. NDCG@10 (y-axis) vs. the length L (x-axis).

2) *Effect of the length* L: Previous studies have shown that modeling the recently successive actions is essential for improving the recommendation quality. Consequently, it is worthwhile to investigate the performance of GCTN with varying lengths L. Fig. 8 shows the GCTN's results *w.r.t.* NDCG@10, and we make two key conclusions:

- With the increase of L, the performance gain of GCTN is significant. And when L reaches a certain value, GCTN obtains the best result. Taking a closer look at Fig. 8, the optimal values on three Amazon datasets are smaller than on ML-1M dataset. We argue the key reason is caused by the large variance of the length of user action sequences (see Table 2).
- Once L crosses the optimal value, the performance of GCTN declines markedly. This demonstrates that modeling short-term user interest with a longer sequence would bring in several irrelevant items for inferring the current user taste.

4.10. Training efficiency (RQ6)

Apart from high accuracy, the training speed of a modern recommendation is also an important factor of its practicality. As such, we conduct this experiment to study the training efficiency of each recommender from four dimensions, including the GPU memory cost, the time cost per epoch, the number of iterations, and the total training time. For fair comparison, we set the embedding size $D = 64$ and apply the optimal values for rest hyper-parameters. All experiments are implemented on a personal computer (AMD Ryzen 7 5800H CPU, Geforce NVIDIA RTX 3060, 16GB memory and Windows 10). Table 6 summarizes the efficiencies of different recommenders on two larger datasets, and we have four key observations:

- Compared to BPRMF, GNN-based recommenders usually consume more memory due to the usage of multiple graph convolution layers. In the meantime, sequential methods also need more memory, as they capture item transition patterns.
- For each recommender, the time cost per epoch on Movies&TV is more than CD&Vinyl as the former dataset includes more action records.
- For each dataset, LightGCN requires more training time than BPRMF, since the former method embeds three linear graph propagation layers to refine user and item embedding learning.
- Compared to LightGCN, LightGCL needs more training time per epoch due to the existence of the SVD-guided augmentation view; nonetheless, it converges to the optimal results with the less number of epochs, contributing by the contrastive learning.
- FPMC takes more time cost than BPRMF due to the modeling of short-term user interest. Furthermore, Caser, SASRec, TISASRec, and HGN are remarkably slower than FPMC. It is reasonable, FPMC only explores first-order MC, while other three methods adopt more complicated operations (*e.g.,* convolution) to model high-order MCs.

Table 6
Efficiency comparison of GCTN against all baselines (Hour/Minute/Second: h/m/s).

Method	Movies&TV				CDs&Vinyl			
	M	P	I	T	M	P	I	T
BPRMF	1.74GB	40.65s	232	2h37m10s	1.71GB	27.24s	267	2h01m13s
FPMC	2.06GB	59.16s	220	3h36m55s	2.00GB	45.37s	167	2h06m18s
Caser	3.82GB	86.17s	278	6h39m15s	3.79GB	62.06s	338	5h49m36s
LightGCN	2.85GB	202s	282	15h49m24s	2.82GB	198s	146	8h01m48s
LightGCL	3.22GB	244s	118	7h59m52s	3.00GB	212s	121	7h07m32s
SASRec	3.04GB	41.74s	482	5h35m18s	3.01GB	39.07s	438	4h43m06s
MLP4Rec	3.21GB	53.27	484	7h09m42s	3.11GB	43.56S	504	6h05m54s
TiSASRec	3.46GB	59.63s	514	8h30m49s	3.38GB	47.98s	404	5h23m02s
GES-SASRec	3.84GB	177s	117	5h45m09s	3.41GB	85.14s	106	2h30m24s
HGN	2.13GB	74.24s	547	11h16m32s	2.11GB	55.36s	389	5h59m20s
GCTN	3.17GB	474s	17	2h14m18s	3.09GB	298s	31	2h29m01s

"M", "P","I", and "T" represent the GPU memory cost, the training time taken per epoch, the number of iterations to converge, and the total running time, respectively.

- Compared with SASRec and TiSASRec, GES-SASRec needs to retain an additional hybrid item graph for information propagation and aggregation. Consequently, GES-SASRec needs more memory and takes larger time overhead for a single iteration.
- Compared to TiSASRec and GES-SASRec, GCTN consumes less memory for model training. It is because a smaller L is sufficient for our method to capture short-term user interest (*cf.* Fig. 8). To achieve satisfactory results, we retain a user-item-category tripartite graph for modeling long-term user preference. Consequently, GCTN takes larger time overhead in each iteration. Fortunately, our solution is much faster to converge than other baselines. For example, GCTN achieves the optimal performance at the 17th epoch, while GES-SASRec and HGN take 117 and 547 epochs, respectively. All in all, GCTN could obtain an optimal balance between effectiveness and efficiency, making it attractive in industrial deployments.

5. Conclusion and future work

In this paper, we contribute a novel sequential recommender called GCTN (graph-coupled time interval network), which successfully combines the advantages of GNNs and Transformers. In particular, such solution composes of: (1) a category-aware graph propagation module to learn robust user and item embeddings; (2) a time-sensitive self-attention mechanism to explore the time interval information within a user action sequence and capture item transition patterns; (3) a personalized gating strategy that dynamically determines the contribution of each part in GCTN. Extensive experiments have shown the effectiveness of GCTN over several state-of-the-art recommenders, and justified the rationality and necessity of different ingredients in our solution.

In future, we will further enhance GCTN's performance in two directions. On one hand, we are interested in inspecting how to integrate social relationships [36,39] into our GCTN and explore dynamic patterns both at sequential-level and social-level. On the other hand, we are also interested in embedding unstructured data (*e.g.,* visual signals [33] and textual reviews [23]) to develop a more trustworthy recommender system.

CRediT authorship contribution statement

Bin Wu: Conceptualization, Data curation, Methodology, Software, Writing – original draft, Writing – review & editing. **Tianren Shi:** Software, Validation, Writing – review & editing. **Lihong Zhong:** Formal analysis, Validation, Visualization. **Yan Zhang:** Methodology, Validation, Writing – review & editing. **Yangdong Ye:** Conceptualization, Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62102369, 62176239, 62002052), and by the Program for Science and Technology in Henan Province of China (Grant No. 232102211045).

References

[1] Xuheng Cai, Chaochao Huang, Lianghao Xia, Xiang Ren, Lightgcl: simple yet effective graph contrastive learning for recommendation, in: Proceedings of the 11th International Conference on Learning Representations, 2023, pp. 1–15.

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, Tat-Seng Chua, Attentive collaborative filtering: multimedia recommendation with item- and component-level attention, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 335–344.

[3] Mingkai He, Weike Pan, Zhong Ming, Bar: behavior-aware recommendation for sequential heterogeneous one-class collaborative filtering, Inf. Sci. 608 (2022) 881–899.

[4] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang, Lightgcn: simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2020, pp. 639–648.

[5] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, Tat-Seng Chua, Nais: neural attentive item similarity model for recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2354–2366.

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

[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk, Session-based recommendations with recurrent neural networks, in: Proceedings of the 4th International Conference on Learning Representations, 2015, pp. 1–9.

[8] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[9] Wendi Ji, Keqiang Wang, Xiaoling Wang, Tingwei Chen, Alexandra Cristea, Sequential recommender via time-aware attentive memory network, in: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, 2020, pp. 565–574.

[10] Santosh Kabbur, Xia Ning, George Karypis, Fism: factored item similarity models for top-n recommender systems, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 659–667.

[11] Wang-Cheng Kang, Julian McAuley, Self-attentive sequential recommendation, in: Proceedings of the 18th IEEE International Conference on Data Mining, 2018, pp. 197–206.

[12] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, 2017, pp. 1–14.

[13] Sara Latifi, Dietmar Jannach, Andrés Ferraro, Sequential recommendation: a study on transformers, nearest neighbors and sampled metrics, Inf. Sci. 609 (2022) 660–678.

[14] Jiacheng Li, Yujie Wang, Julian McAuley, Time interval aware self-attention for sequential recommendation, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 322–330.

[15] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, Jun Ma, Neural attentive session-based recommendation, in: Proceedings of the 26th ACM on Conference on Information and Knowledge Management, 2017, pp. 1419–1428.

[16] Muyang Li, Xiangyu Zhao, Chuan Lyu, Minghao Zhao, Runze Wu, Ruocheng Guo, Mlp4rec: a pure mlp architecture for sequential recommendations, in: Proceedings of the 31st International Joint Conference on Artificial Intelligence, 2022, pp. 2138–2144.

[17] Nian Li, Chen Gao, Depeng Jin, Qingmin Liao, Disentangled modeling of social homophily and influence for social recommendation, IEEE Trans. Knowl. Data Eng. (2022) 1–14.

[18] Yang Li, Tong Chen, Peng-Fei Zhang, Hongzhi Yin, Lightweight self-attentive sequential recommendation, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, 2021, pp. 967–977.

[19] Jie Liao, Wei Zhou, Fengji Luo, Junhao Wen, Min Gao, Xiuhua Li, Jun Zeng, Sociallgn: light graph convolution network for social recommendation, Inf. Sci. 589 (2022) 595–607.

[20] Jing Lin, Weike Pan, Zhong Ming, Fissa: fusing item similarity models with self-attention networks for sequential recommendation, in: Proceedings of 14th ACM Conference on Recommender Systems, 2020, pp. 130–139.

[21] Zemin Liu, Trung-Kien Nguyen, Yuan Fang, Tail-gnn: tail-node graph neural networks, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021, pp. 1109–1119.

[22] Chen Ma, Peng Kang, Xue Liu, Hierarchical gating networks for sequential recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019, pp. 825–833.

[23] Chen Ma, Peng Kang, Bin Wu, Qinglong Wang, Xue Liu, Gated attentive-autoencoder for content-aware recommendation, in: Proceedings of the 12th ACM International Conference on Web Search and Data Mining, 2019, pp. 519–527.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452–461.

[25] Steffen Rendle, Christoph Freudenthaler, Lars Schmidt-Thieme, Factorizing personalized Markov chains for next-basket recommendation, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 811–820.

[26] Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.

[27] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, Max Welling, Modeling relational data with graph convolutional networks, in: Proceedings of the 15th European Semantic Web Conference, 2018, pp. 593–607.

[28] Zhongchuan Sun, Bin Wu, Youwei Wang, Yangdong Ye, Sequential graph collaborative filtering, Inf. Sci. 592 (2022) 244–260.

[29] Zhongchuan Sun, Bin Wu, Yunpeng Wu, Yangdong Ye, Apl: adversarial pairwise learning for recommender systems, Expert Syst. Appl. 118 (mar 2019) 573–584.

[30] Jiaxi Tang, Ke Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.

[31] Abhishek Vijayvargiya, One-way analysis of variance, Journal of Validation Technology 15 (1) (2009) 62.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2019, pp. 165–174.

[33] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, Tat-Seng Chua, Mmgcn: multi-modal graph convolution network for personalized recommendation of micro-video, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1437–1445.

[34] Bin Wu, Xiangnan He, Yun Chen, Liqiang Nie, Kai Zheng, Yangdong Ye, Modeling product's visual and functional characteristics for recommender systems, IEEE Trans. Knowl. Data Eng. 34 (3) (2022) 1330–1343.

[35] Bin Wu, Xiangnan He, Zhongchuan Sun, Liang Chen, Yangdong Ye, Atm: an attentive translation model for next-item recommendation, IEEE Trans. Ind. Inform. 16 (3) (2020) 1448–1459.

[36] Bin Wu, Xiangnan He, Le Wu, Xue Zhang, Yangdong Ye, Graph-augmented co-attention model for socio-sequential recommendation, IEEE Trans. Syst. Man Cybern. Syst. (2023) 1–13.

[37] Bin Wu, Xiangnan He, Qi Zhang, Meng Wang, Yangdong Ye, Gcrec: graph-augmented capsule network for next-item recommendation, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–14.

[38] Bin Wu, Yangdong Ye, Yun Chen, Visual appearance or functional complementarity: which aspect affects your decision making?, Inf. Sci. 476 (2019) 19–37.

[39] Bin Wu, Lihong Zhong, Lina Yao, Yangdong Ye, Eagcn: an efficient adaptive graph convolutional network for item recommendation in social Internet of things, IEEE Int. Things J. 9 (17) (2022) 16386–16401.

[40] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, Xing Xie, Self-supervised graph learning for recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 726–735.

[41] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, Meng Wang, A survey on accuracy-oriented neural recommendation: from collaborative filtering to information-rich recommendation, IEEE Trans. Knowl. Data Eng. (2022) 1–20.

[42] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, Tieniu Tan, Session-based recommendation with graph neural networks, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 346–353.

[43] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S. Sheng, S. Sheng, Zhiming Cui, Xiaofang Zhou, Hui Xiong, Recurrent convolutional neural network for sequential recommendation, in: Proceedings of the 28th International Conference on World Wide Web, 2019, pp. 3398–3404.

[44] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, Julian McAuley, Cosrec: 2d convolutional neural networks for sequential recommendation, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2173–2176.

[45] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, Xiangnan He, A simple convolutional generative network for next item recommendation, in: Proceedings of the 12th ACM International Conference on Web Search and Data Mining, 2019, pp. 582–590.

[46] Qi Zhang, Bin Wu, Zhongchuan Sun, Yangdong Ye, Gating augmented capsule network for sequential recommendation, Knowl.-Based Syst. 247 (2022) 108817.

[47] Shuai Zhang, Lina Yao, Bin Wu, Xiwei Xu, Xiang Zhang, Liming Zhu, Unraveling metric vector spaces with factorization for recommendation, IEEE Trans. Ind. Inform. 16 (2) (2020) 732–742.

[48] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, Ed H. Chi, A model of two tales: dual transfer learning framework for improved long-tail item recommendation, in: Proceedings of the Web Conference 2021, 2021, pp. 2220–2231.

[49] Tianyu Zhu, Leilei Sun, Guoqing Chen, Graph-based embedding smoothing for sequential recommendation, IEEE Trans. Knowl. Data Eng. 35 (1) (2023) 496–508.