SelfGNN: Self-Supervised Graph Neural Networks for Sequential Recommendation

Yuxi Liu
School of Electronic and Computer Engineering, Peking University
Shenzhen, China
liuyuxi.tongji@gmail.com

Lianghao Xia
Computer Science Department, University of Hong Kong
Hong Kong, China
aka_xia@foxmail.com

Chao Huang*
Computer Science Department & IDS, University of Hong Kong
Hong Kong, China
chaohuang75@gmail.com

ABSTRACT

Sequential recommendation effectively addresses information overload by modeling users' temporal and sequential interaction patterns. To overcome the limitations of supervision signals, recent approaches have adopted self-supervised learning techniques in recommender systems. However, there are still two critical challenges that remain unsolved. Firstly, existing sequential models primarily focus on long-term modeling of individual interaction sequences, overlooking the valuable short-term collaborative relationships among the behaviors of different users. Secondly, real-world data often contain noise, particularly in users' short-term behaviors, which can arise from temporary intents or misclicks. Such noise negatively impacts the accuracy of both graph and sequence models, further complicating the modeling process. To address these challenges, we propose a novel framework called Self-Supervised Graph Neural Network (SelfGNN) for sequential recommendation. The SelfGNN framework encodes short-term graphs based on time intervals and utilizes Graph Neural Networks (GNNs) to learn short-term collaborative relationships. It captures long-term user and item representations at multiple granularity levels through interval fusion and dynamic behavior modeling. Importantly, our personalized self-augmented learning structure enhances model robustness by mitigating noise in short-term graphs based on long-term user interests and personal stability. Extensive experiments conducted on four real-world datasets demonstrate that SelfGNN outperforms various state-of-the-art baselines. Our model implementation codes are available at https://github.com/HKUDS/SelfGNN.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential Recommendation, Self-Supervised Learning, Graph Neural Networks, Collaborative Filtering, Recommender Systems

*Chao Huang is the Corresponding Author.

SIGIR'24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07…$15.00
https://doi.org/10.1145/3626772.3657716

ACM Reference Format:
Yuxi Liu, Lianghao Xia, and Chao Huang. 2024. SelfGNN: Self-Supervised Graph Neural Networks for Sequential Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'24), July 14–18, 2024, Washington, DC, USA.* ACM, Austin, TX, USA, 10 pages. https://doi.org/10.1145/3626772.3657716

1 INTRODUCTION

Recommender systems have emerged as a crucial field for addressing the information overload issue, providing benefits to both users and service providers such as online shopping platforms (*e.g.*, Tmall, Amazon) [9], video websites (*e.g.*, TikTok, YouTube) [38, 45], and social networks (*e.g.*, Gowalla, Facebook) [24, 47]. A wide range of solutions has been proposed, such as factorization-based methods [29, 30] and neural collaborative filtering models [13, 14]. Recent research has introduced Graph Neural Networks (GNNs) to capture high-order relations over user-item interaction graphs for encoding user preferences. For instance, NGCF [36] utilizes Graph Convolutional Networks (GCNs) to model user-item interactions, and LightGCN [12] simplifies and enhances the power of GCNs. Recent efforts have also been dedicated to dynamic GNNs in the context of recommendation systems. For instance, DGCF [21] introduces dynamic graphs into recommendation, while DGSR [46] connects different user sequences through dynamic graph structures.

One important line of research in recommender systems is sequential recommendation, which involves analyzing users' temporal interaction patterns and predicting their future behaviors. Various deep learning methods have been developed to address this task. For example, GRU4Rec [15] utilizes recurrent neural networks (RNN) to model sequential patterns. Similarly, SASRec [18] and Bert4Rec [31] leverage self-attention mechanisms to capture item-item pairwise correlations. To address the data scarcity challenge,

Despite the significant progress made in graph and sequential recommenders with SSL, there are still several challenges that remain unaddressed (illustrated in Fig 1). Firstly, existing long-short-term sequential recommenders [3, 48] often mainly focus on encoding single user sequence, which overlook the essential periodical collaborative relationships between users [7]. Traditional GNNs solely rely on static data. Although some work has constructed dynamic and temporal GNNs, they overlook the consideration of both long and short-term aspects in GNNs, nor do they address the construction of a more robust dynamic learning paradigm.

Secondly, the impressive results achieved by existing SSL methods heavily depend on high-quality data [41]. However, real-world data often contains noise, such as users' misclicks, temporary intents, or interest shifts [28]. Existing models tend to amplify such

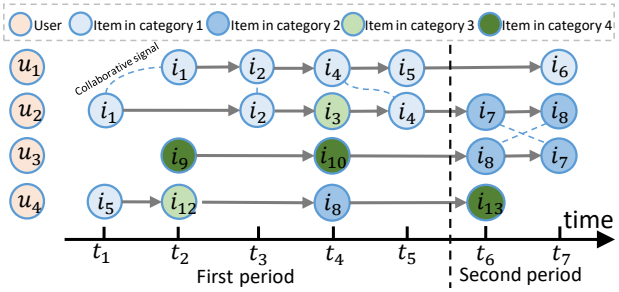


Figure 1: Illustrative examples reflecting collaborative relations (between u_1 and u_2 in first period, between u_2 and u_3 in second period), temporal intents (u_2 adopting i_3), interest shift (u_3), and user with variable interests (u_4).

noises through multi-hop graph message propagation or multi-layer pairwise self-attention, which can greatly impact the representation of other nodes [32]. It is worth noting that the notion of "noise" varies for different users, as some users have diverse interests while others have more stable preferences. Current denoising approaches lack the ability to effectively handle user-specific noise and often rely on random data augmentation. These challenges hinder the accurate representation of user behaviors.

Contribution. In light of the aforementioned challenges, this work proposes a Self-Supervised Graph Neural Network (SelfGNN) for sequential recommendation. SelfGNN aims to capture user interests by incorporating dynamic collaborative information and personalizing short-term denoising through self-augmented learning. SelfGNN is built upon three key paradigms. (1) **Short-term Collaborative Graphs Encoding.** The global user-item graph is divided into several short-term graphs based on time intervals. GCNs are employed to propagate collaborative high-order information. For instance, in Figure 1, u_1 and u_2 have significant collaborative signals in the first period, while u_3 has similar behaviors with u_2 in the second period. In this way, we not only capture the collaborative patterns but also include short-term temporal information. (2) **Multi-level Long-term Sequential Learning.** We conduct dual-level sequence modeling, which forms the interest with different granularity complement each other. At the time interval level, the features learned from different short-term graphs are treated as sequences to generate long-term collaborative representations for users. At the instance level, the complete sequence of user behavior is modeled by self-attention mechanisms. (3) **Personalized Self-Augmented Learning for Denoising.** A self-augmented learning task is designed to correct the corresponding relationships in the short-term graphs based on the long-term user interests. This task learns personalized weights for users and adjusts the noise level according to users' interest stability (*e.g.*, u_2 with a stable tendency and u_4 with variable interests in Figure 1). Consequently, the SelfGNN can discriminate abnormal interactions in the short term and mitigate their impact by adapting to different users.

The key contributions can be summarized as follows:

- We introduce a novel self-supervised graph recommender framework, which effectively captures dynamic user interests by integrating interval-level periodical collaborative relationship learning and attentive instance-level sequential modeling.
- Our method incorporates a personalized self-augmented learning component, enabling effective denoising of noisy interactions in sparse short-term graphs. It also generates personalized weights to adapt to different users, enhancing the model's ability to cater to individual preferences and stability.
- Our extensive experiments on real-world datasets demonstrate that SelfGNN outperforms various baseline models. Furthermore, the robust performance of our framework is thoroughly validated, further underscoring its effectiveness and practical utility.

2 RELATED WORK

2.1 GNNs for Recommendation

Recent studies have introduced diverse Graph Neural Network architectures for encoding user-item interactions within graph-structured data [25, 33]. Networks like GC-MC [34] and NGCF [36] are adept at capturing high-order collaborative signals through extensive information propagation. Innovations such as LightGCN [12] and GCCF [4] have streamlined and enhanced the foundational GCN architecture. These have also inspired temporal GNN methods for sequential recommendation, exemplified by SRGNN [40], GCE-GNN [37], and TGSREC [8], each contributing novel graph construction and attention mechanisms. Dynamic graph design has been another area of focus, with models like DGCF [21] and TG-MC [2] integrating temporal dynamics into user and item representation learning. DGSR [46] and SURGE [3] further advance the field by explicitly modeling dynamic user-item interaction graphs.

However, existing methods have yet to fully address the integration of long and short-term user features within a robust sequential GNN model. Our proposed SelfGNN tackles this challenge by revisiting GNN with a self-augmented learning paradigm. Our proposed new SelfGNN, grounded in long-term feature supervision, aims to reduce noise in short-term graphs and fortify the model's ability to learn from both immediate and enduring user preferences.

2.2 Sequential Recommender Systems

User interaction sequences play a pivotal role in recommender systems, as they highlight the chronological patterns of user behavior. These patterns provide critical insights into both the short-term dependencies of user interactions and the gradual shifts in user preferences. As a result, a multitude of studies have been devoted to uncovering the dynamics of user preferences over time, with sequential recommendation being a notable area, such as those developed using Recurrent Neural Networks like GRU4Rec [15]. In recent years, the transformative capabilities of the Transformer architecture have catalyzed the broad application of self-attention mechanisms. These mechanisms excel at capturing the intricate correlations between items within a user's sequence. Notable implementations include SASRec [18], Bert4Rec [31], TiSASRec [20] and MBHT [43]. Moreover, CLSR [48], while adept at discerning both short- and long-term user interests within sequential recommendations, falls short in thoroughly tackling the challenge of data noise. This aspect remains a significant area for improvement, as addressing it could lead to more accurate recommendation results.

2.3 Recommender Systems with SSL

Self-supervised learning has gained significant traction in bolstering the capabilities of recommendation systems, evident in both graph-based and sequential models [17, 27]. Its popularity stems from its capacity to produce auxiliary supervision signals that effectively tackle the prevalent challenge of data sparsity [16, 22, 44]. For instance, SGL [39] capitalizes on node self-discrimination through random augmentations of graph structures. CoTRec [42] employs a dual-level contrastive learning strategy across two similar item relation graphs. Meanwhile, AutoCF [41] designs autonomous masked autoencoding framework to offer robust self-supervised signals. In the realm of sequential recommendations, ICLRec [5] and CoSeRec [23] enhance interaction sequences by employing techniques such as cropping, masking, or reversing, followed by the application of contrastive learning across these diversified views. Distinct from these methodologies, our self-supervised learning framework is uniquely tailored to filter out noise from short-term interactions. This is achieved by embedding a deep understanding of long-term user behavior into the model, thereby refining the accuracy and relevance of the recommender system's output.

3 METHODOLOGY

In this section, we introduce our proposed SelfGNN framework and illustrate the overall model architecture in Figure 2.

3.1 Notations and Formalization

In this paper, we propose to combine the strength of GNNs in capturing high-order collaborative relations and the strength of sequence modeling for capturing users' preference shifts, with the help of self-supervised learning techniques. In such scenario, we define $\mathcal{U} = \{u_1, ..., u_i, ..., u_I\}(|\mathcal{U}| = I)$ and $\mathcal{V} = \{v_1, ..., v_j, ..., v_J\}$ $(|\mathcal{V}| = J)$ to represent the set of users and items respectively.

The method of segmenting user interaction data into different periods. We partition the entire time interval in the dataset according to timestamp, ranging from the earliest occurrence at time t_b to the latest occurrence at time t_e, into an average of T short-term time intervals, where T is a tunable hyperparameter. The duration of each time interval is $(t_e - t_b)/T$. The adjacent matrix $\mathcal{A}_t \in \mathbb{R}^{I\times J}$ indicates the implicit relationships between each user and the items they interacted with in the (t)-th period. Each entry $\mathcal{A}_{t,i,j}$ in $\mathcal{A}_t$ is set as 1 if user u_i has adopted item v_j, and $\mathcal{A}_{t,i,j} = 0$ otherwise.

Formalization. Given partitioned graph-structured interactions $\{\mathcal{A}_t|1 \le t \le T\}$, predict the future interactions $\mathcal{A}_{T+1}$, with the following abstracted optimization objective:

$$\begin{aligned} &\arg\min_{\Theta_f,\Theta_g} \ \mathcal{L}_{rec}(\hat{\mathcal{A}}_{T+1}, \mathcal{A}_{T+1}) + \mathcal{L}_{sal}(\mathbf{E}_s, \mathbf{E}_l) \\ &\hat{\mathcal{A}}_{T+1} = f(\mathbf{E}_l, \mathbf{E}_s; \mathbf{\Theta}_f), \quad \mathbf{E}_l, \mathbf{E}_s = g(\{\mathcal{A}_t\}; \mathbf{\Theta}_g) \end{aligned} \tag{1}$$

where $\mathcal{L}_{rec}$ denotes the loss function measuring the error between predictions $\hat{\mathcal{A}}_{T+1}$ and ground-truth $\mathcal{A}_{T+1}$, and $\mathcal{L}_{sal}$ denotes the SSL objective for aligning the short-term hidden embeddings $\mathbf{E}_s$ and the long-term embeddings $\mathbf{E}_l$. $\hat{\mathcal{A}}_{T+1}$ is acquired by the prediction function f with parameter set $\mathbf{\Theta}_f$, based on the short and long-term information $\mathbf{E}_s, \mathbf{E}_l$ encoded by the encoder g with parameters $\mathbf{\Theta}_g$.

3.2 Short-term Collaborative Relation Encoding

Firstly, we perform collaborative information learning for each short-term interaction graph. Specifically, we project each user u_i and item v_j in each time slot t, into a d-dimensional latent space with $\mathbf{e}^{(u)}_{t,i}, \mathbf{e}^{(v)}_{t,j} \in \mathbb{R}^d$. These embedding vectors compose embedding matrices $\mathbf{E}^{(u)}_t \in \mathbb{R}^{I\times d}$ and $\mathbf{E}^{(v)}_t \in \mathbb{R}^{J\times d}$ for the t-th period, respectively. Inspired by LightGCN [12], we employ the following simplified GCN for short-term graph modeling:

$$\mathbf{z}^{(u)}_{t,i} = \sigma(\mathcal{A}_{t\,i,*} \cdot \mathbf{E}^{(v)}_t), \quad \mathbf{z}^{(v)}_{t,j} = \sigma(\mathcal{A}_{t\,j,*} \cdot \mathbf{E}^{(u)}_t) \tag{2}$$

where $\mathbf{z}^{(u)}_{t,i}, \mathbf{z}^{(v)}_{t,j} \in \mathbb{R}^d$ denote the information aggregated from neighboring nodes to the centric node u_i and v_j and $\sigma(\cdot)$ denotes the LeakyReLU function. We adopt random drops to the edges of the graphs to mitigate the overfitting problem. We stack such GCN layers for high-order propagation.

In the l-th layer, the message-passing process is:

$$\mathbf{e}^{(u)}_{t,i,l} = \mathbf{z}^{(u)}_{t,i,l} + \mathbf{e}^{(u)}_{t,i,l-1}, \quad \mathbf{e}^{(v)}_{t,j,l} = \mathbf{z}^{(v)}_{t,j,l} + \mathbf{e}^{(v)}_{t,j,l-1} \tag{3}$$

where $\mathbf{e}^{(u)}_{t,i,l}, \mathbf{e}^{(v)}_{t,j,l} \in \mathbb{R}^d$ represent the embeddings of u_i and v_j at the l-th GNN layer for the t-th time period. Since different layers emphasize different propagation ranges, we concatenate them to obtain the final short-term embedding:

$$\mathbf{e}^{(u)}_{t,i} = \mathbf{e}^{(u)}_{t,i,1}||\cdots||\mathbf{e}^{(u)}_{t,i,L}, \quad \mathbf{e}^{(v)}_{t,j} = \mathbf{e}^{(v)}_{t,j,1}||\cdots||\mathbf{e}^{(v)}_{t,j,L} \tag{4}$$

3.3 Multi-level Long-term Sequence Modeling

In this section we aim to model the long-term relationship at two levels: i) Interval-Level Sequential Pattern Modeling, which integrates short-term features into long-term embeddings based on temporal attention, capturing the dynamic changes from period to period, and ii) Instance-Level Sequential Pattern Modeling, which learns pairwise relations between specific item instances directly.

3.3.1 **Interval-Level Sequential Pattern Modeling.** For every user and item, we construct a chronological embedding sequence based on their short-term embeddings. To inject the temporal information into our interval-level sequence, we utilize the Gated Recurrent Unit (GRU) network [1], instead of the position encoding used in Transformer because position embedding would be too simplistic to capture the temporal information when dealing with a small number of intervals in our work. The sequences for user u_i and item v_j are defined as follows:

$$\begin{aligned} S^{intv}_i &= (\mathbf{h}^{(u)}_{1,i}, \cdots, \mathbf{h}^{(u)}_{t,i}, \cdots, \mathbf{h}^{(u)}_{T,i}) \\ S^{intv}_j &= (\mathbf{h}^{(v)}_{1,j}, \cdots, \mathbf{h}^{(v)}_{t,j}, \cdots, \mathbf{h}^{(v)}_{T,j}) \\ \mathbf{h}^{(u)}_{t,i} = \text{GRU}(\mathbf{e}^{(u)}_{t,i}, \mathbf{h}^{(u)}_{t-1,i}), &\quad \mathbf{h}^{(v)}_{t,j} = \text{GRU}(\mathbf{e}^{(v)}_{t,j}, \mathbf{h}^{(v)}_{t-1,j}) \end{aligned} \tag{5}$$

where $\mathbf{h}^{(u)}_{t,i}, \mathbf{h}^{(v)}_{t,j} \in \mathbb{R}^d$ denote the hidden states of GRU at the t-th time slot. Then SelfGNN devices the multi-head dot-product attention, denoted as Self-Att$(\cdot)$, over the interval-level sequences S^{intv}_i and S^{intv}_j, respectively, to excavate the dynamic patterns:

$$\bar{\mathbf{H}}^{(u)}_i = \text{Self-Att}(S^{intv}_i), \quad \bar{\mathbf{H}}^{(v)}_j = \text{Self-Att}(S^{intv}_j) \tag{6}$$

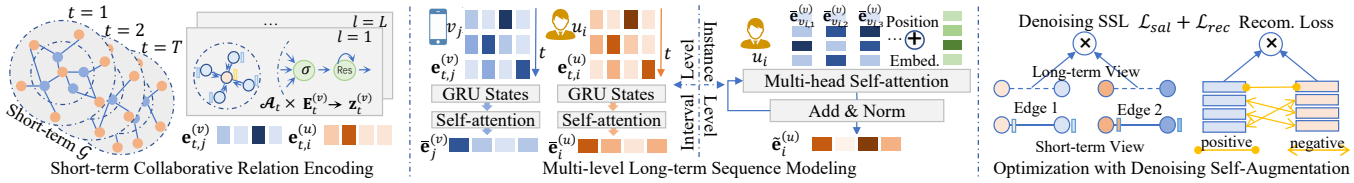


Figure 2: Overall framework of the proposed SelfGNN model.

Finally, we sum up the features to obtain the long-term features:

$$\bar{\mathbf{e}}_i^{(u)} = \sum_{t=1}^{T} \bar{\mathbf{H}}_{i,t}^{(u)}, \quad \bar{\mathbf{e}}_j^{(v)} = \sum_{t=1}^{T} \bar{\mathbf{H}}_{j,t}^{(v)} \tag{7}$$

where $\bar{\mathbf{e}}_i^{(u)}, \bar{\mathbf{e}}_j^{(v)} \in \mathbb{R}^d$ refer to the output embedding vectors for u_i and v_j, containing their long-term patterns. Compared to only modeling instance sequences, our interval-level modeling enriches the module with periodical collaborative signals.

3.3.2 **Instance-Level Sequential Pattern Modeling.** Motivated by the success of the self-attention mechanism [18, 31], we also augment our SelfGNN with the self-attention network directly over sequences containing users' interacted item instances. Denoting the m-th interacted item of user u_i as $v_{i,m}$, where $m \in \{1, 2, ..., M\}$ and M represents the maximum interaction length, we compose the following sequences for recording u_i's actions:

$$S_{i,0}^{inst} = (\bar{\mathbf{e}}_{v_{i,1}}^{(v)} + \mathbf{p}_1, \cdots, \bar{\mathbf{e}}_{v_{i,M}}^{(v)} + \mathbf{p}_M) \tag{8}$$

where $\bar{\mathbf{e}}_{v_{i,m}}^{(v)} \in \mathbb{R}^d$ refers to the embedding for item $v_{i,m}$, and $\mathbf{p}_m \in \mathbb{R}^d$ denotes the learnable positional embedding for the m-th position. We employ L_a layers of self-attention with residual connections to capture the sequential patterns:

$$S_{i,l}^{inst} = \sigma(\text{Self-Att}(S_{i,l-1}^{inst})) + S_{i,l-1}^{inst}, \quad \tilde{\mathbf{e}}_i^{(u)} = \sum S_{i,L_a}^{inst} \tag{9}$$

where $S_{i,l}^{inst}$ denotes sequence for u_i in the l-th self-attention iteration. Based on the processed sequences, we sum the element embeddings and use the above $\tilde{\mathbf{e}}_i^{(u)} \in \mathbb{R}^d$ to represent user u_i with instance-level sequential correlations.

3.4 Multi-view Aggregation and Prediction

Before prediction, we aggregate the multi-views user features derived from the instance-level and the interval-level approach, and make final predictions as follows:

$$\hat{\mathcal{A}}_{T+1,i,j} = \hat{\mathbf{e}}_i^{(u)\top} \cdot \bar{\mathbf{e}}_j^{(v)}, \quad \hat{\mathbf{e}}_i^{(u)} = \bar{\mathbf{e}}_i^{(u)} + \tilde{\mathbf{e}}_i^{(u)} \tag{10}$$

where $\hat{\mathcal{A}}_{T+1,i,j} \in \mathbb{R}$ denotes the prediction for u_i interacting with v_j, in the future $(T+1)$-th time slot by calculating the similarity between user and item. Nextly, positive samples consist of items that users have interacted with, while negative samples consist of items not interacted with. Imposing a restriction to prevent the predicted values from becoming arbitrarily large, we optimize the following loss function as follows:

$$\mathcal{L}_{rec} = \sum_{i=1}^{I} \sum_{k=1}^{N_{pr}} \max(0, 1 - \hat{\mathcal{A}}_{T+1,i,p_k} + \hat{\mathcal{A}}_{T+1,i,n_k}) \tag{11}$$

where N_{pr} is the number of samples, p_k, n_k represents the (k)-th positive and negative item index respectively.

3.5 Personalized Denoising Self-Augmentation

To alleviate the ubiquitous data sparsity and data noise problem in users' sequential behavior data, our SelfGNN is further enhanced by a personalized denoising self-supervised learning task. The term "noise" refers to temporary intents or misclicks, which cannot be considered as long-term user interests or new recent points of interest for predictions. Specifically, our SSL task focuses on filtering short-term non-inherent user preferences using long-term behavior patterns. This design is based on the observation that, users' behaviors may be motivated by short-term random interests, *e.g.*, a user who is not in favor of hiking may also buy hiking shoes and energy drinks because of a one-time activity. Such noisy behavior data may disturb the modeling of users' true interests in the long term. Furthermore, to accurately identify such noisy short-term behaviors, we personalize the denoising SSL task with different strengths for different users, which caters to users with different levels of interest variety, as shown in Figure 3.

In particular, for each training sample of our denoising SSL, we randomly sample two observed user-item edges (u_i, v_j) and $(u_{i'}, v_{j'})$ from the short-term graphs $\mathcal{A}_t$, and align the pairwise likelihood difference scores given. Taking (u_i, v_j) as an example, the formula is as follows:

$$s_{t,i,j} = \sum_{k=1}^{d} \sigma(e_{t,i,k}^{(u)} \cdot e_{t,j,k}^{(v)}), \quad \bar{s}_{i,j} = \sum_{k=1}^{d} \sigma(\bar{e}_{i,k}^{(u)} \cdot \bar{e}_{j,k}^{(v)}) \tag{12}$$

where $s_{t,i,j}, \bar{s}_{i,j} \in \mathbb{R}$ denote the likelihood of u_i interacting with v_j in the t-th period and in long term, respectively. These predictions are made using the learned short-term collaborative embeddings $\mathbf{e}_{t,i}^{(u)}, \mathbf{e}_{t,j}^{(v)}$, and the learned long-term sequential embeddings $\bar{\mathbf{e}}_i^{(u)}, \bar{\mathbf{e}}_j^{(v)}$. And $e_{t,i,k}^{(u)}, e_{t,j,k}^{(v)}, \bar{e}_{i,k}^{(u)}, \bar{e}_{j,k}^{(v)}$ denote the element value of the k-th embedding dimension. The likelihood $s_{t,i',j'}$ and $\bar{s}_{i',j'}$ of user $u_{i'}$ interacting with item $v_{j'}$ are calculated in a similar manner.

With the likelihood scores, SelfGNN aligns the score differences between the short and long-term views. Following the loss design of the main task, we adopt the SSL objective function:

$$\mathcal{L}_{sal} = \sum_{t=1}^{T} \sum_{(u_i,v_j),(u_{i'},v_{j'})} \max(0, 1 - d_1 \cdot d_2)$$
$$d_1 = w_{t,i}\bar{s}_{i,j} - w_{t,i'}\bar{s}_{i',j'}, \quad d_2 = s_{t,i,j} - s_{t,i',j'} \tag{13}$$

where d_1 represents the likelihood difference between edge (u_i, v_j) and $(u_{i'}, v_{j'})$ in the long-term view, and d_2 represents the difference score between edge (u_i, v_j) and $(u_{i'}, v_{j'})$ in the t-th period short-term view. Specially, SelfGNN applies learned weights $w_{t,i}$ and $w_{t,i'}$ for capturing the different degrees of preference consistency between short-term and long-term for users. These weights, which are formally calculated as follows, personalize the self-supervised

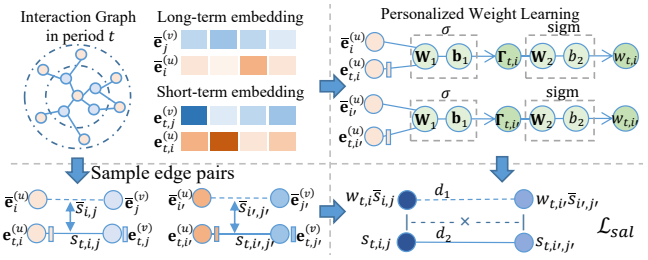


Figure 3: Workflow of the personalized SSL paradigm.

learning to avoid false cross-view preference alignment:

$$w_{t,i} = \text{sigm}(\mathbf{\Gamma}_{t,i} \cdot \mathbf{W}_2 + b_2),$$
$$\mathbf{\Gamma}_{t,i} = \sigma((\bar{\mathbf{e}}_i^{(u)} + \mathbf{e}_{t,i}^{(u)} + \bar{\mathbf{e}}_i^{(u)} \odot \mathbf{e}_{t,i}^{(u)})\mathbf{W}_1 + \mathbf{b}_1) \quad (14)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{sal}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{sal} \times 1}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{sal}}$, $b_2 \in \mathbb{R}$ are trainable transformation parameters, sigm is sigmoid function. $w_{t,i}$ determines the stability of the long-term score $\bar{s}_{i,j}$ as a reference for noise correction in the t-th period. We exclude the backpropagation of the long-term score $\bar{s}_{i,j}$ and $\bar{s}_{i',j'}$, to direct the optimization process towards correcting the short-term score and learning the user stability weight $w_{t,i}$ and $w_{t,i'}$. In the loss function Eq. (13), the role of d_1 is to guide the optimization of d_2 in terms of direction and magnitude. Specifically, if (u_i, v_j) exhibits a strong long-term relationship, while $(u_{i'}, v_{j'})$ has a weaker relationship (*i.e.*, $\bar{s}_{i,j} > \bar{s}_{i',j'}$), and the preference stability of u_i is higher than that of $u_{i'}$ (*i.e.*, $w_{t,i} > w_{t,i'}$), then $d_1 > 0$. This leads to the optimization of d_2 towards a larger value, indicating that the short-term relationship $s_{t,i,j}$ corresponding to (u_i, v_j) will be strengthened, while the short-term relationship $s_{t,i',j'}$ of $(u_{i'}, v_{j'})$ will be weakened. The larger the value of d_1, the greater the degree to which d_2 is optimized. A similar situation occurs when $d_1 < 0$, in which d_2 will be optimized for smaller negative values.

By combining the SSL task with the main recommendation task, we have the final loss function as follows:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_1 \cdot \mathcal{L}_{sal} + \lambda_2 \cdot \|\Theta\|_{\text{F}}^2 \quad (15)$$

where weights λ_1 and λ_2 are used to balance the importance of different loss terms. Additionally, we apply weight decay with l_2 regularization on the parameters Θ of our model.

We summarize the learning process of *SelfGNN* in Alg 1.

3.6 In-Depth Analysis of SelfGNN

3.6.1 **Theoretical Analysis.** Our personalized SSL method addresses short-term noises by generating adaptive gradients based on users who exhibit significant disparities between their long-term and short-term preferences. More specifically, for Eq. (13), we can quantify the impact of edges (u_i, v_j), $(u_{i'}, v_{j'})$ in the short-term and long-term view on the node embeddings by:

$$c'(w_{t,i'}) = \bar{s}_{i',j'} \cdot (s_{t,i,j} - s_{t,i',j'}) = \bar{s}_{i',j'} \cdot s_{t,i,j} - \bar{s}_{i',j'} \cdot s_{t,i',j'}$$
$$c'(w_{t,i}) = \bar{s}_{i,j} \cdot (s_{t,i',j'} - s_{t,i,j}) = \bar{s}_{i,j} \cdot s_{t,i',j'} - \bar{s}_{i,j} \cdot s_{t,i,j} \quad (16)$$

$$c'(s_{t,i',j'}) = w_{t,i} \cdot \bar{s}_{i,j} - w_{t,i'} \cdot \bar{s}_{i',j'}$$
$$c'(s_{t,i,j}) = w_{t,i'} \cdot \bar{s}_{i',j'} - w_{t,i} \cdot \bar{s}_{i,j} \quad (17)$$

Algorithm 1: Model Inference of the SelfGNN Framework.

Input: number of short-term time intervals T, partitioned graph-structured interactions $\{\mathcal{A}_t | 1 \le t \le T\}$, user behavior sequences $\{v_{i,1}, ..., v_{i,m}, ..., v_{i,M}\}$, sample number N, maximum epoch number E, weight for loss of self-augmented learning λ_1, regularization weight λ_2, learning rate ρ
Output: trained parameters in $\mathbf{\Theta}$

Initialize all parameters in $\mathbf{\Theta}$
for $e = 1$ *to* E **do**
  for $t = 1$ *to* T **do**
    Propagate multi-hop collaborative information through the LightGCN paradigm
    Obtain user short-term embeddings $\mathbf{e}_{t,i}^{(u)}$ and item short-term embeddings $\mathbf{e}_{t,j}^{(v)}$ at period t
  end
  Construct interval-level user feature change sequences S_i^{intv} and item feature change sequences S_j^{intv}
  Generate interval-level long-term user embedding $\bar{\mathbf{e}}_i^{(u)}$ and item embedding $\bar{\mathbf{e}}_j^{(v)}$
  Capture instance-level sequential long-term user feature $\tilde{\mathbf{e}}_i^{(u)}$
  Draw a mini-batch $\mathbf{U}$ from all users, each with N_{pr} positive-negative samples
  $\mathcal{L} = \lambda_2 \cdot \|\mathbf{\Theta}\|_{\text{F}}^2$
  for *each* $u_i \in \boldsymbol{U}$ **do**
    Compute predictions $\hat{\mathcal{A}}_{T+1,i,p_k}, \hat{\mathcal{A}}_{T+1,i,n_k}$
    $\mathcal{L} += \sum_{k=1}^{N_{pr}} \max(0, 1 - \hat{\mathcal{A}}_{T+1,i,p_k} + \hat{\mathcal{A}}_{T+1,i,n_k})$
  end
  for $t = 1$ *to* T **do**
    Sample N_{sal} pairs of user-item edge pairs (u_i, v_j) and $(u_{i'}, v_{j'})$, denoted as $\mathbf{P}$
    for *each* $(u_i, v_j), (u_{i'}, v_{j'}) \in \boldsymbol{P}$ **do**
      Compute the likelihood score of user interacting with item in the t-th period, *i.e.*, $s_{t,i,j}$ for (u_i, v_j) and $s_{t,i',j'}$ for $(u_{i'}, v_{j'})$
      Compute the likelihood score of user interacting with item in long term, *i.e.*, $\bar{s}_{i,j}$ for (u_i, v_j) and $\bar{s}_{i',j'}$ for $(u_{i'}, v_{j'})$
      Compute user personalized weight $w_{t,i}$
      Compute long-term difference d_1 and short-term difference d_2
      $\mathcal{L} += \lambda_1 \cdot \max(0, 1 - d_1 \cdot d_2)$
    end
  end
  for *each parameter* $\theta \in \mathbf{\Theta}$ **do**
    $\theta = \theta - \rho \cdot \partial\mathcal{L}/\partial\theta$
  end
end
return all parameters Θ

where $c'(x)$ represents the gradient with respect to x. In Eq. (16), if $s_{t,i,j} - s_{t,i',j'} > 0$, the gradient $c'(w_{t,i'})$ will monotonically increase with the variation of $\bar{s}_{i',j'}$. Moreover, when the difference between the long-term value and short-term edge value is larger, *i.e.*, when $\bar{s}_{i',j'} s_{t,i',j'}$ is smaller, the learning gradient of $w_{t,i'}$ will be larger. The same applies to the effect of $\bar{s}_{i,j}$ and $s_{t,i,j}$ on the gradient of $w_{t,i}$. In Eq. (17), if $w_{t,i'} \bar{s}_{i',j'}$ is small, indicating that user i' exhibits unstable short-term behavior and has a weak long-term relationship with item j', the gradient of $s_{t,i',j'}$ will increase. Overall, self-augmented learning emphasizes gradients from samples with instability between long-term and short-term user-item relationships, thereby enhancing the model training process. Additionally, short-term graphs offer a novel level of detail for understanding user behavior, thereby enhancing model performance.

3.6.2 **Model Complexity Analyses.** The short-term GNN encoding takes $\sum_{t=1}^{T} O(L \times |\mathcal{A}_t| \times d) = O(L \times |\mathcal{A}| \times d)$ time complexity,

Table 1: Statistics of the experimental datasets.

Dataset	User #	Item #	interaction #	Density
Amazon-book	11199	30821	375916	1.1e-3
Gowalla	48653	52621	1807125	7.1e-4
Movielens	24312	8688	1758929	8.3e-3
Yelp	19751	38391	1467157	1.9e-3

where $|\mathcal{A}|$ is the number of all interaction edges, which means the time cost is the same as that of a complete long-term graph. The instance-level sequence learning takes $O((T\times d^2+T^2\times d)\times(I+J))$, and interval-level sequence modeling with attention mechanism takes $O((M\times d^2+M^2\times d)\times B)$, where B is the batch size. In the SAL paradigm, the cost is $O(B\times N_{sal}\times d)$. we only need to do a vector dot product of randomly selected edges, requiring less time than commonly used SSL methods such as contrastive learning [35].

4 EVALUATION

Our experiments are designed to answer the following questions.
RQ1: How does SelfGNN perform *w.r.t* top-k recommendation as compared with the state-of-the-art models?
RQ2: What are the benefits of the components proposed?
RQ3: How does SelfGNN perform in the data noise issues?
RQ4: How do the key hyperparameters influence SelfGNN?
RQ5: In real cases, how can the designed self-augmented learning in SelfGNN provide useful interpretations?

4.1 Experimental Settings

4.1.1 **Experimental Datasets.** The experiments are conducted on four open-source datasets shown in Table 1. **Amazon-book** [11]: it records users' ratings of Amazon books in 2014. **Gowalla** [6]: it is the user geolocation check-in dataset from Gowalla in 2010. **Movielens** [10]: it contains users' ratings for movies from 2002 to 2009. **Yelp**: this is a dataset of venue reviews, and we sample data from 2009 to 2019. We apply the 5-core setting to ensures that each user and item has at least five interactions.

4.1.2 **Evaluation Protocols.** We follow the evaluation protocol of recent works on sequential recommendation [5, 23]. The data is split into three sets: the most recent interaction of each user is used as the test set, the penultimate one is used as the validation set, and the remaining interactions in users' sequences are used as the training data. We randomly sample 10,000 users as test users. For each test user, we sample negative samples by randomly selecting 999 items that the user does not interact with. For evaluation, both positive and negative items are ranked together for each user and we employ two metrics, *i.e.*, *Hit Rate (HR)@N* and *Normalized Discounted Cumulative Gain (NDCG)@N* [26, 36].

4.1.3 **Compared Baseline Methods.** We compare our SelfGNN with state-of-the-art baselines from different research lines.

Conventional Factorization-based Technique.

- **BiasMF** [19]: It augments matrix factorization to incorporate implicit feedback, temporal effects, and confidence levels.

Neural Factorization Method.

- **NCF** [14]: This method designs a neural network framework for collaborative filtering based on neural networks.

Sequential Recommendation.

- **GRU4Rec** [15]: It is a session-based recommendation with RNN.
- **SASRec** [18]: The model employs self-attention mechanisms to capture the sequential patterns within recommender systems.
- **TiSASRec** [20]: This method proposes to view the user's interactions as a sequence with different time intervals, modeling them as relations between any two interactions.
- **Bert4Rec** [31]: This approach models user behavior sequences with the bidirectional self-attention network based on Transformer architecture through the Cloze task.

Traditional Graph Neural Networks Method.

- **NGCF** [36]: This method crafts a graph neural network to enhance high-order collaborative filtering in recommendation.
- **LightGCN** [12]: It simplifies the GCNs in recommendation.
- **SRGNN** [40]: It model separated session sequences into graph structure to capture complex item transitions.
- **GCE-GNN** [37]: It introduces the global-level and the session-level pairwise item-transition graph for sequence recommender.

Dynamic or Temporal Graph Neural Networks Method.

- **SURGE** [3]: It proposes to aggregate implicit signals into explicit ones from user behaviour sequences by designing GNN models.
- **DGCF** [21]: It utilizes dynamic graph structures to effectively encapsulate the collaborative and sequential dynamics.
- **TGSREC** [8]: It proposes a Temporal Collaborative Transformer layer that integrates collaborative attention to capture both user-item interactions and temporal dynamics over a temporal graph.

Recommenders enhanced by Self-Supervised Learning.

- **SGL** [39]: This model employs random techniques for structural and feature augmentation, creating diverse data perspectives that enrich the self-supervised learning process.
- **ICLRec** [5]: The model discerns user intents by clustering across all user behavior sequences and refines these intents through contrastive learning to ensure alignment with user expectations.
- **CoSeRec** [23]: It introduces new sequential data augmentation strategies to enhance contrastive learning for recommendation.
- **CoTRec** [42]: It proposes a self-supervised method to exploit the session graph to two views and two distinct graph encoders.
- **CLSR** [48]: It is a contrastive learning method to capture long and short-term interests by comparing with proxy representations.

4.1.4 **Implementation Details.** We implement SelfGNN with TensorFlow and use Adam optimizer with the $1e^{-3}$ learning rate and 0.96 epoch decay ratio. The embedding dimension size is 64. The number of graph neural layers is selected from {1,2,3}, the training batch size is selected from {128, 256, 512}, the attention layer for instance-level sequence is selected from {2,3,4}, the short-term graph number T is selected from $\{t|2\leq t\leq 14\}$, the dimension of personalized weight is selected from {16,32,48}, the weight λ_1 for SSL loss is tuned from $\{1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}\}$, the weight λ_2 for regularization loss is $1e^{-2}$, and dropout rate is 0.5.

4.2 Overall Performance Comparison (RQ1)

The results are presented in Table 2. SelfGNN and the best-performed baseline are retrained 5 times for p-values.

Table 2: Performance comparison on Amazon, Gowalla, MovieLens and Yelp datasets in terms of *HR* and *NDCG*.

Data	Amazon				Gowalla				Movielens				Yelp			
Metric	Top 10		Top 20		Top 10		Top 20		Top 10		Top 20		Top 10		Top 20	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
BiasMF	0.314	0.170	0.457	0.206	0.543	0.362	0.669	0.393	0.208	0.115	0.311	0.141	0.281	0.151	0.411	0.184
NCF	0.306	0.165	0.431	0.197	0.606	0.418	0.716	0.446	0.198	0.107	0.299	0.133	0.313	0.169	0.462	0.206
GRU4Rec	0.148	0.077	0.232	0.098	0.393	0.253	0.515	0.284	0.127	0.076	0.238	0.099	0.087	0.043	0.143	0.057
SASRec	0.268	0.171	0.324	0.185	0.562	0.360	0.688	0.392	0.121	0.072	0.163	0.082	0.106	0.057	0.151	0.057
TiSASRec	0.270	0.171	0.326	0.185	0.573	0.376	0.690	0.405	0.120	0.071	0.165	0.082	0.092	0.051	0.133	0.061
Bert4Rec	0.312	0.173	0.445	0.207	0.544	0.359	0.676	0.393	0.175	0.087	0.299	0.118	0.290	0.158	0.428	0.193
NGCF	0.277	0.147	0.307	0.180	0.550	0.347	0.692	0.383	0.147	0.076	0.238	0.099	0.325	0.174	0.475	0.212
LightGCN	0.244	0.127	0.271	0.159	0.456	0.286	0.592	0.318	0.211	0.115	0.319	0.142	0.346	0.189	0.495	0.226
SRGNN	0.199	0.109	0.281	0.129	0.575	0.371	0.687	0.399	0.122	0.069	0.166	0.080	0.097	0.054	0.137	0.064
GCE-GNN	0.279	0.145	0.401	0.180	0.578	0.373	0.709	0.406	0.169	0.088	0.264	0.112	0.297	0.157	0.440	0.194
SURGE	0.362	0.215	0.468	0.242	0.479	0.259	0.654	0.304	0.237	0.124	0.337	0.149	0.278	0.144	0.428	0.182
SGL	0.336	0.184	0.465	0.217	0.413	0.259	0.495	0.280	0.223	0.120	0.282	0.135	0.320	0.170	0.461	0.206
DGCF	0.307	0.163	0.357	0.201	0.524	0.339	0.629	0.388	0.216	0.116	0.309	0.136	0.318	0.168	0.452	0.202
TGSRec	0.228	0.172	0.426	0.289	0.424	0.347	0.695	0.557	0.145	0.125	0.229	0.177	0.217	0.143	0.458	0.301
ICLRec	0.285	0.226	0.343	0.240	0.197	0.122	0.265	0.193	0.190	0.126	0.248	0.140	0.195	0.113	0.274	0.133
CoSeRec	0.368	0.230	0.465	0.254	0.556	0.362	0.669	0.390	0.166	0.112	0.221	0.125	0.195	0.110	0.270	0.129
CoTRec	0.338	0.213	0.421	0.233	0.531	0.400	0.608	0.419	0.216	0.117	0.311	0.141	0.283	0.152	0.416	0.185
CLSR	0.332	0.189	0.441	0.217	0.529	0.296	0.699	0.339	0.061	0.036	0.076	0.040	0.261	0.134	0.400	0.169
SelfGNN	**0.391**	**0.240**	**0.487**	**0.264**	**0.637**	**0.437**	**0.745**	**0.465**	**0.239**	**0.128**	**0.341**	**0.154**	**0.365**	**0.201**	**0.509**	**0.237**
p-val.	$2.9e^{-4}$	$2.7e^{-3}$	$8.4e^{-5}$	$3.1e^{-3}$	$5.6e^{-6}$	$9.5e^{-5}$	$2.0e^{-8}$	$2.0e^{-8}$	$3.1e^{-4}$	$1.7e^{-3}$	$8.7e^{-4}$	$5.5e^{-4}$	$1.8e^{-4}$	$2.2e^{-4}$	$2.3e^{-5}$	$7.6e^{-5}$

- **Overall Performance Validation.** Our proposed SelfGNN consistently outperforms all other SOTAs across various evaluation metrics. This observation substantiates the superiority of SelfGNN, attributed to the following factors. i) Integration of short-term graphs and long-term sequence information: Our model leverages both short-term collaborative information and long-term holistic dependencies to capture user interest changes. By combining these two types of information, our model gains a comprehensive understanding of user preferences and achieves improved performance. ii) Personalized self-augmented learning schema: Our model benefits from a personalized self-augmented learning framework, which enables it to handle the noise present in sparse short-term user interactions. Through cross-view supervision signals that bridge the gap between short-term and long-term representations, our model can effectively correct for noise and enhance the quality of recommendations. In addition, note that the Gowalla dataset is a locally-punched dataset, with many users repeatedly going to certain locations and checking in, so the simple NCF model performed best in these baselines.
- **Superiority of Sequential Recommendation with Graph.** Referring to state-of-the-art baselines, sequential approaches based on dynamic graphs such as DGCF, TGSRec, and SURGE outperform most conventional Attention-based and RNN-based models (*e.g.*, SASRec, TiSASRec, Bert4Rec, and GRU4Rec), as graphs can extract collaborative signals instead of solely modeling the sequence-specific context. Meanwhile, our SelfGNN transforms long-term information into multiple short-period graphs, which aggregate time and collaborative signals at different stages, reflecting the challenging long-range dependence of users through stage preferences. Consequently, SelfGNN exhibits superior performance compared to traditional GNN-based recommendations and sequential recommendations.
- **Effectiveness of Self-Supervised Learning.** From the evaluation results, it is evident that self-supervised learning significantly enhances the performance of all approaches (*e.g.*, SGL, ICLRec, CoSeRec, CoTRec, and CLSR), regardless of whether they are GNN-based or sequence-based. TGSRec and our SelfGNN similarly combine collaborative filtering signals with attention-based sequential learning. However, SelfGNN significantly outperforms TGSRec, thanks to our self-supervised learning paradigm, which TGSRec does not incorporate for model enhancement. Self-supervised learning leverages different views of the data itself to augment supervision signals, thereby addressing the crucial limitation of supervision insufficiency in recommendation. Specifically, SGL employs stochastic graph data augmentation. ICLRec and CoSeRec generate self-augmented sequences through insertion and substitution operations. CoTRec formulates contrastive learning as maximizing agreement between the representation of the last clicked item and the predicted item. CLSR adopts a contrastive task to supervise the similarity between long and short-term interest and their corresponding interest proxies. In comparison to the aforementioned methods, the self-augmentation in our SelfGNN offers two main advantages: i) Instead of relying on random masking or permutation, which may inadvertently introduce noises into the sequence and collaboration modeling, our approach is based on users' short-term and long-term preferences using the original data. ii) We propose an innovative denoising SSL approach that explicitly uses stable long-term patterns to filter users' random and noisy short-term behaviors. This allows for the removal of noise caused by abnormal short-term interactions of stable users.

4.3 Ablation Study of SelfGNN (RQ2)

We individually remove the applied techniques from three major parts for ablation study, and Table 3 shows the results.

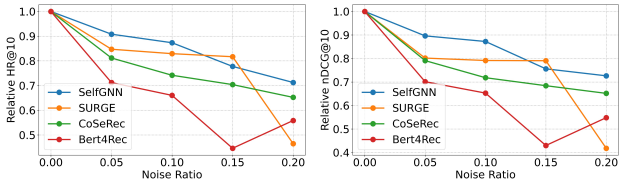
Table 3: Module ablation study on SelfGNN *w.r.t* top 10.

Category	Data	Amazon		Gowalla		Movielens		Yelp	
	Variants	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
SAL	*-SAL*	0.364	0.218	0.631	0.428	0.211	0.110	0.328	0.173
	-UW	0.355	0.219	0.635	0.430	0.198	0.104	0.268	0.136
Short-term	*-STG*	0.347	0.206	0.525	0.353	0.165	0.088	0.313	0.171
Long-term	*-ATL*	0.337	0.200	0.564	0.363	0.211	0.115	0.324	0.175
	-GAT	0.277	0.164	0.541	0.339	0.185	0.097	0.248	0.134
	-CF	0.247	0.156	0.320	0.209	0.170	0.089	0.246	0.133
SelfGNN		**0.387**	**0.239**	**0.637**	**0.437**	**0.239**	**0.128**	**0.365**	**0.201**

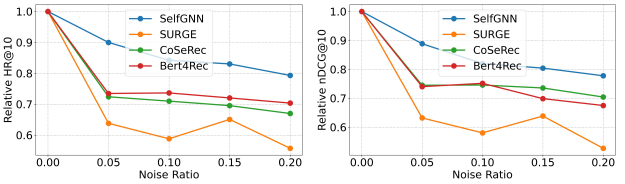
- **Effect of Short-term Graph Structure Learning.** We replace the several short-term graphs with a global user-item graph, denoted as *-STG*. From the table, it is evident that the *-STG* model significantly reduces the accuracy of the recommendation task. This is attributed to that our SelfGNN captures CF relationships and their changes at different periods and emphasizes the more important stages through temporal attention. In contrast, the static graph used in *-STG* fails to capture these periodic aspects.
- **Effect of Long-term Learning.** In the long-term feature representation, we construct a variant called *-GAT* by replacing the GRU-attention mechanism for interval-level learning with a feature fusion method based on summation and another variant called *-ATL* by removing the attention-based instance-level sequential pattern. It is evident from the results that SelfGNN consistently outperforms the variants. Assigning different importance to user preferences at different stages is crucial, as the GRU effectively captures the position information and dynamic interest changes. Additionally, attention modeling for long-term behavior at the instance level represents users from a comprehensive sequential perspective, allowing for fusion and mutual supervision with interval-level features from short-term graphs. Regarding the effectiveness of collaborative relation encoding, we conducted ablation experiments that remove the collaborative filtering paradigm, as indicated by *-CF*. It demonstrate that constructing the user-item collaborative graph is crucial for the sequence recommendation. The collaborative knowledge it acquires enhances the model's ability to represent users.
- **Effect of Long and Short-term Personalized Self-Augmented Learning.** We also conduct ablations on the components of personalized self-augmented learning by removing the personalized user weight (*-UW*) and the entire self-augmented learning schema (*-SAL*). The experimental results prove that supervising short-term graph noise through long-term features significantly improves the recommendation effectiveness. Comparing the results of *-UW* and SelfGNN, it can be observed that weighting for different users is crucial for the Amazon, Yelp and Movielens datasets, in which many users may have variable behavioral interests so that personalized weights are needed for discrimination of noise and selective correction of short-term behavior encoding.

4.4 Model Robustness Test (RQ3)

4.4.1 **Performance *w.r.t* Data Noise Degree.** To evaluate the robustness of SelfGNN against noise issues, we conduct experiments by randomly replacing different percentages of real interaction items with randomly-generated fake items for all users and retraining the model using the corrupted sequences as input. The noise ratios considered are 5%, 10%, 15%, and 20%, respectively. The target models for evaluation are SURGE, CoSeRec, and Bert4Rec, which are well-performing sequential recommendation models. Figure 4 illustrates the performance degradation in different noise scenarios, showcasing the potential of our SelfGNN in addressing data noise. In the sparser Amazon dataset, while SURGE initially performs well with small amounts of noise, its performance rapidly declines when faced with 20% noise. On the other hand, our SelfGNN is less affected by noise compared to the other models. Even with 20% noise, it achieves a *HR@10* value of 72% and an *NDCG@10* value of 73% in the absence of noise. In the relatively denser Movielens dataset, our model attains a relative *HR@10* value of 79% and an *NDCG@10* value of 78% in the case of 20% noise. We attribute this superiority to SelfGNN's ability to mitigate noise in the short-term graph through long-term features in personalized self-augmented learning. Furthermore, our long-term representation is obtained by combining short-term features through interval-level GRU attention, allowing the final long-term user interest representation to benefit from the denoising effect when short-term noise is reduced.

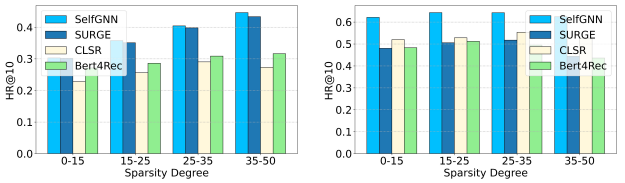


(a) Amazon data



(b) Movielens data

Figure 4: Relative performance degradation *w.r.t* noise ratio.



(a) Amazon data (b) Gowalla data

Figure 5: Performance *w.r.t* interaction degrees.

4.4.2 **Performance *w.r.t* Data Sparsity.** To assess the impact of data sparsity on model performance, we categorize users into different groups based on their interaction numbers (*e.g.*, 0-15, 15-25) and evaluate our SelfGNN alongside the top-performing models (*i.e.*, SURGE, CLSR, and Bert4Rec). As depicted in Figure 5, the models generally exhibit lower predictive accuracy for sparser users. However, our SelfGNN consistently demonstrates greater stability than the other models. This can be attributed to our approach, which combines collaborative graph modeling and attention-based

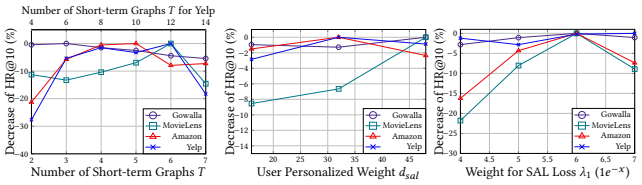


Figure 6: Hyperparameter study of the SelfGNN.

sequence modeling, compensating for missing information and leveraging additional self-supervised signals. In contrast, SURGE relies solely on item-item graphs for modeling, which may not capture sufficient information in sparse scenarios. This observation is further corroborated by the experimental results from the Gowalla dataset. Similarly, the sequential models (*i.e.*, CLSR and Bert4Rec) also exhibit poorer performance when the sequences are very short.

4.5 Hyperparameter Analysis (RQ4)

The decrease of the experimental results with the changing of two key hyperparameters (*i.e.*, T, d_{sal} and λ_1) is plotted in Figure 6. Firstly, when the dimension d_{sal} for user personalization weight is set to 48 on the Gowalla and Movielens and is set to 32 on Amazon and Yelp, SelfGNN achieves optimal performance. Secondly, the performance initially improves and then declines as the number of short-term graph partitions T increases because a higher number of partitions allows the model to capture more fine-grained information and if T becomes too large, the graphs become excessively sparse. The results shows the optimal number of splits varies across different datasets due to their duration and densities. For instance, the optimal number of short-term partitions for the Yelp dataset is 12, while for Gowalla, only $T = 3$ are needed. Thirdly, the best value for the self-augmented loss weight λ_1 is $1e^{-7}$ for the Yelp dataset and $1e^{-6}$ for the rest of the dataset.

4.6 Case Study (RQ5)

In this section, we utilize concrete data examples to investigate the effect of self-augmented learning denoising. As shown in Figure 7, we randomly selected a user (6128) and part of the behavior sequence of that user, and the other user (824) which has at least 20 interaction items in common with the user (6128). We calculate and normalize the final user-item similarity scores $\hat{\mathcal{A}}_{T+1,i,j}$ in the case of both without (wo-score) and with (w-score) self-augmented learning. We display the item's score, title, and category in Figure 7. Upon examination, it can be observed that the score for item (6282) decreases from 0.8239 to 0.3686 with the inclusion of self-augmented learning. This indicates that the model identifies it as a noisy interaction that needs to be weakened. We prove that this behavior may be noise behavior from two aspects. Firstly, the category of item (6282) is "Mystery" which differs from the category of other items ("Action & Adventure") that the user follows. Additionally, as a user who shares numerous similar interests with user (6128), user (824) does not follow item (6282). Moreover, from the encoding heatmap of items, it is evident that item (6282) exhibits distinctive features across multiple dimensions compared to other items. Furthermore, the features of items interacted with by the same user in a continuous manner exhibit greater dissimilarity compared to

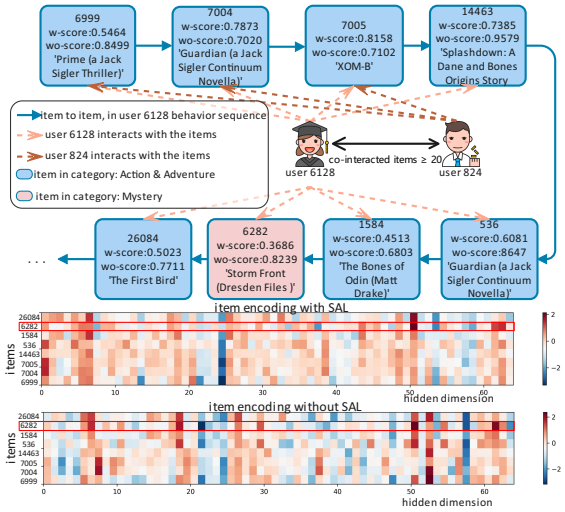


Figure 7: User-item likelihood scores in a partial sequence of user (6128) and heatmap of items' embedding in SelfGNN.

cases where SAL is not employed, which proves the SAL paradigm lightens the smoothing problem caused by GNNs.

To validate whether other noise items similar to item (6282) in all user sequences exhibit significant feature differences compared to the remaining normal items in their respective sequences, we conducted a statistical experiment. We computed the average cosine similarity of the feature embeddings between item (6282) and other items in the behavior sequence of user (6128) under both with and without self-augmented paradigms. Then, we similarly calculated the average cosine similarity between items that satisfy the noise conditions in other user sequences and the remaining items in the sequence where this item is located. The statistical results are presented in Table 4, which show that in the self-augmented learning model (w-SAL), the average cosine similarity between noisy items and other items is significantly lower than the corresponding value in the model without self-augmented learning (w/o-SAL). In summary, our model demonstrates the ability to accurately identify and weaken interaction behaviors that are likely to be noise.

Table 4: Similarity statistics of noise items with other items.

User	User 6218		Other Users	
Model	w-SAL	w/o-SAL	w-SAL	w/o-SAL
Average Cosine Similarity	0.605	0.715	0.620	0.711

5 CONCLUSION

In this paper, we explore sequential recommender systems with graph neural networks and introduce a novel personalized self-augmented learning paradigm to boost recommendation robustness. Our SelfGNN framework refines interest representation by integrating collaborative patterns and user behavior sequences, employing self-augmented learning to adaptively reduce short-term noise based on user-specific stability traits. Extensive experiments confirm the model's enhanced performance and denoising ability over existing baselines. For future work, we aim to investigate adaptive dynamic short-term graph partitioning techniques to more accurately capture short-term characteristics across various datasets, further enhancing the recommendation performance.

REFERENCES

[1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural News Recommendation with Long- and Short-term User Representations. In *ACL*. 336–345.

[2] Esther Rodrigo Bonet, Duc Minh Nguyen, and Nikos Deligiannis. 2021. Temporal Collaborative Filtering with Graph Convolutional Neural Networks. In *ICPR*. IEEE, 4736–4742.

[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. , 378–387 pages.

[4] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*, Vol. 34. 27–34.

[5] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *WWW*. 2172–2182.

[6] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*. 1082–1090.

[7] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *CIKM*. 433–442.

[8] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *CIKM*. 433–442.

[9] Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. 2020. Understanding echo chambers in e-commerce recommender systems. In *SIGIR*. 2261–2270.

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* (2015).

[11] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*.

[13] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30 (2018).

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[15] BalÃ¡zs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[16] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive graph contrastive learning for recommendation. In *KDD*. 4252–4261.

[17] Mengyuan Jing, Yanmin Zhu, Tianzi Zang, and Ke Wang. 2023. Contrastive self-supervised learning in recommender systems: A survey. *ACM Transactions on Information Systems (TOIS)* 42, 2 (2023), 1–39.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. IEEE, 197–206.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[20] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[21] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic Graph Collaborative Filtering. In *ICDM*. IEEE, 322–331.

[22] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2023. Self-Supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 35, 1 (2023), 857–876.

[23] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive Self-supervised Sequential Recommendation with Robust Augmentation.

[24] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*. 259–270.

[25] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. *NeurIPS* 28 (2015).

[26] 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. In *SIGIR*. 89–98.

[27] Xubin Ren, Lianghao Xia, Yuhao Yang, Wei Wei, Tianle Wang Wang, Xuheng Cai, and Chao Huang. 2024. SSLRec: A Self-Supervised Learning Framework for Recommendation. In *WSDM*. 567–575.

[28] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. , 501–509 pages.

[29] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NeurIPS*. 1257–1264.

[30] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31 (2019).

[31] 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 *CIKM*. 1441–1450.

[32] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S Yu Philip. 2022. Graph structure learning with variational information bottleneck. In *AAAI*, Vol. 36. 4165–4174.

[33] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. HiGPT: Heterogeneous Graph Language Model. *arXiv preprint arXiv:2402.16024* (2024).

[34] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. arXiv:1706.02263 [stat.ML]

[35] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 [cs.LG]

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[37] 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 *SIGIR*. 169–178.

[38] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-modal self-supervised learning for recommendation. In *WWW*. 790–800.

[39] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.

[40] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.

[41] Lianghao Xia, Chao Huang, Chunzhen Huang, Kangyi Lin, Tao Yu, and Ben Kao. 2023. Automated self-supervised learning for recommendation. In *WWW*. 992–1002.

[42] Xin Xia, Hongzhi Yin, Junliang Yu, Yingxia Shao, and Lizhen Cui. 2021. Self-Supervised Graph Co-Training for Session-based Recommendation. In *CIKM*. 2180–2190.

[43] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. 2022. Multi-Behavior Hypergraph-Enhanced Transformer for Sequential Recommendation. In *KDD*. 2263–2274.

[44] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2024. Self-Supervised Learning for Recommender Systems: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 36, 1 (2024), 335–355.

[45] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. 2021. Mining latent structures for multimedia recommendation. In *MM*. 3872–3880.

[46] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2023. Dynamic Graph Neural Networks for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 35, 5 (2023), 4741–4753.

[47] Zhou Zhao, Hanqing Lu, Deng Cai, Xiaofei He, and Yueting Zhuang. 2016. User Preference Learning for Online Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 28, 9 (2016), 2522–2534.

[48] Yu Zheng, Chen Gao, Jianxin Chang, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2022. Disentangling Long and Short-Term Interests for Recommendation. In *WWW*. 2256–2267.