SLED: Structure Learning based Denoising for Recommendation

SHENGYU ZHANG* and TAN JIANG*, Zhejiang University, China
KUN KUANG†, Zhejiang University, China
FULI FENG†, University of Science and Technology of China, China
JIN YU, Alibaba Group, China
JIANXIN MA, Alibaba Group, China
ZHOU ZHAO, Zhejiang University, China
JIANKE ZHU, Zhejiang University, China
HONGXIA YANG, Alibaba Group, China
TAT-SENG CHUA, National University of Singapore, Singapore
FEI WU†, Zhejiang University, China

In recommender systems, click behaviors play a fundamental role in mining users' interests and training models (clicked items as positive samples). Such signals are *implicit* feedback and are arguably less representative of users' inherent interests. Most existing works denoise implicit feedback by introducing external signals, such as gaze, dwell time, and "like" behaviors. However, such *explicit* feedback is not always routinely available, or might be problematic to collect on a large scale. In this paper, we identify that an interaction's related structural patterns in its neighborhood graph are potentially correlated with some outcome of implicit feedback (*i.e.*, users' ratings after consuming items), analogous to findings in other domains such as social networks. Inspired by this finding, we propose a novel Structure LEarning based Denoising (SLED) framework for denoising recommendation without explicit signals, which consists of two phases: *center-aware graph structure learning* and *denoised recommendation*. Phase 1 pre-trains a structural encoder in a self-supervised manner and learns to capture an interaction's related structural patterns in its neighborhood graph. Phase 2 transfers the structure encoder to downstream recommendation datasets, which helps to down-weight the effect of noisy interactions on user interest modeling and loss calculation. We collect a relatively noisy industrial dataset across several days during a period of product promotion festival. Extensive experiments on this dataset and multiple public datasets demonstrate that the proposed SLED framework can significantly improve the recommendation quality over various base recommendation models.

Additional Key Words and Phrases: Recommender System, Implicit Feedback, User-item Graph, Structure Learning, Denoise

*Both authors contributed equally to this research.
†Corresponding Authors.

Authors' addresses: Shengyu Zhang, sy_zhang@zju.edu.cn; Tan Jiang, jiangtan@zju.edu.cn, Zhejiang University, Hangzhou, China, 43017-6221; Kun Kuang, Zhejiang University, Hangzhou, China, kunkuang@zju.edu.cn; Fuli Feng, University of Science and Technology of China, China, fulifeng93@gmail.com; Jin Yu, Alibaba Group, Hangzhou, China, yujin.yujin@alibaba-inc.com; Jianxin Ma, Alibaba Group, Hangzhou, China, jason.mjx@alibaba-inc.com; Zhou Zhao, Zhejiang University, Hangzhou, China, zhaozhou@zju.edu.cn; Jianke Zhu, Zhejiang University, Hangzhou, China, jkzhu@zju.edu.cn; Hongxia Yang, Alibaba Group, Hangzhou, China, yang.yhx@alibaba-inc.com; Tat-Seng Chua, National University of Singapore, Singapore, dcscts@nus.edu.sg; Fei Wu, Zhejiang University, Hangzhou, China, wufei@zju.edu.cn.

1046-8188/2023/8-ART $15.00
https://doi.org/10.1145/3611385

1 INTRODUCTION

Over the past few years, the quantity of searchable items has increased substantially, which intensifies the need for recommender systems to explore users' preferences more effectively [2, 6, 8–11] in various fields such as e-commerce, mass media, and video platforms. Recommendation models typically mine user preferences from the behaviors logged in the system. Among different kinds of behaviors, implicit feedback, such as clicks, play a fundamental role due to its prevalence and easy accessibility. However, industrial recommender systems contain inherent causes of noisy implicit feedback, including factors that may affect users' first impressions (*e.g.*, clickbait issue [51]) and external distractions (*e.g.*, sale promotions or suggestions from friends). Some works [22, 38, 51, 53] find that implicit feedback may not always reflect the inherent interests of users, and would result in inaccurate recommendation. Hence, denoising implicit feedback is of paramount importance for users' interest mining and model training in recommendation.

To the best of our knowledge, most existing works denoise implicit feedback by considering additional behavior signals [51, 90, 92]. However, some signals (*e.g.*, gaze) are not always routinely available, and others (*e.g.*, like and follow) can be hard to collect on a large scale since users may be unwilling to give such feedback. To pursue high usability, we set the target as denoising implicit feedback without additional behavior signals. Our key belief is that the local structural patterns in the user-item interaction graph[1] reveal the reliability of implicit feedback since the contagion is related to its neighborhood graph structure[2] [71]. Figure 1 provides empirical evidence where we depict the structural representation and outcome of implicit feedback, *i.e.*, users' ratings on items after consuming the items. Note that we obtain the structural representation through structure learning in a self-supervised manner (*c.f.* Section 3.1) without access to explicit feedback, such as ratings. We have mainly two observations in Figure 1:

- Interactions that share similar neighborhood structures (closer to each other) have similar ratings, reflecting the potential correlation between neighborhood structure and the reliability of implicit feedback.
- The inter-rating distances are larger than intra-rating distances between nodes, *i.e.*, interactions with different qualities being located relatively remotely, which means that interactions with different qualities are likely to be with different structures.

Therefore, we can take neighborhood structure as a proxy of explicit feedback to facilitate denoising implicit feedback.

In this light, we propose a novel Structure LEarning based Denoising (SLED) framework. As shown in Figure 2, SLED consists of two phases: center-aware structure learning and denoised recommendation. The core idea of structure learning is to learn a structure encoder that captures structural patterns given a neighborhood graph. However, not all structural patterns in the neighborhood graph are related to the *center interaction* from which the neighborhood graph is sampled. For example, as shown in Figure 2(a), not all loop structures in the neighborhood graph contain the center interaction (edge in bold). Loop structure containing or not containing the center interaction might be different structural patterns in determining the reliability of the center interaction. Furthermore, the neighborhood graphs of different interactions of a particular user have overlapped sub-graphs and structural patterns. As such, it is necessary to capture structural patterns related to each interaction such that we can distinguish the interactions of a particular user and determine their reliabilities. In this regard, the proposed **center-aware structure learning** progressively filters unrelated structural patterns *w.r.t.* the distance between structural patterns and the center interaction in the representation space. In **denoised recommendation**, the pre-trained center-aware structure encoder will be transferred to downstream recommendation datasets and yield structural representations for interactions by encoding their neighborhood graphs. We predict the reliability weights of interactions from their structural representations and down-weight the noisy interactions in

[1] In a user-item graph, users and items are represented as nodes, and user-item interactions are represented as edges.

[2] We use the neighborhood graph of an interaction denotes the r-ego [63] network of the corresponding edge.

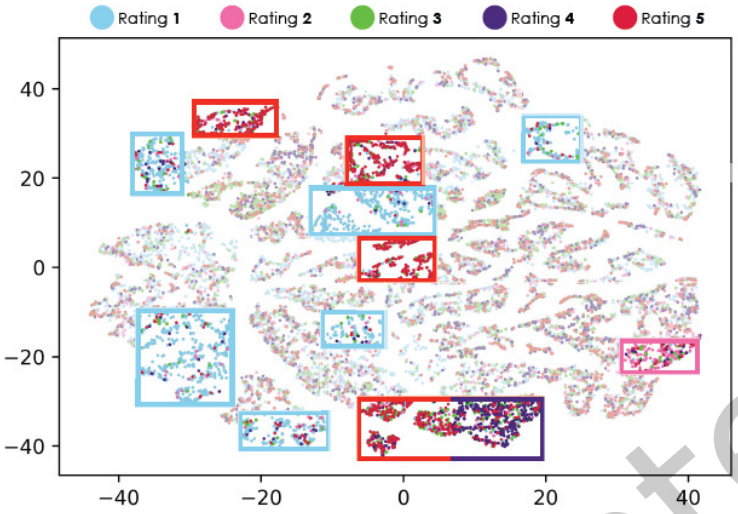

Fig. 1. Visualization of interactions' structural representations (t-SNE transformed) from Beauty dataset of Amazon Review. Each node denotes a user-item interaction where the position and the color are determined by the structural representation and the user's rating on the item, respectively. Interactions that have similar neighborhood structures (closer in space) are likely to have similar ratings (the same color).

recommendation model training. For training modern recommendation models (*e.g.*, graph-based and sequence-based models), there are mainly two components requiring denoising, *i.e.*, user representation extraction from historical interactions, and the ranking score prediction based on the user representation and the target interaction. Therefore, we denoise user representation learning by down-weighting the contribution of noisy interactions to the user representation, and down-weight ranking losses computed based on noisy target interactions. Note that we ***do not*** use explicit feedback, such as the rating values, in any part of the proposed framework, which is one of the critical merits.

In the experiments, we evaluate the effectiveness of SLED following the experimental setting of pretraining frameworks [14]. Concretely, we first perform center-aware structure learning on two large-scale recommendation datasets, *i.e.*, Taobao User Behavior dataset [103], and MIND [83] dataset. Then, we transfer the pre-trained structure encoder to an in-domain recommendation dataset and two out-of-domain recommendation datasets (with distribution shift from the pretraining dataset) for denoising. We conduct relatively extensive experiments, including ablation studies and case studies, validating the rationality of our analysis and the effectiveness of SLED. We show that SLED can improve various recommendation architectures, including two graph-based methods and two sequence-based methods, in a model-agnostic manner. In particular, we show that SLED can improve the performance of non-noisy testing recommendations, and can improve the base model by a large margin in a noisy industrial scenario where there are a lot of external distractions and noisy implicit feedback. The main contributions are summarized as follows:

- We study how to denoise the implicit feedback in recommender systems without additional behavior signals (*e.g.*, ratings, and like behavior). We identify that interaction-related structural patterns on the user-item graph are helpful for denoising.

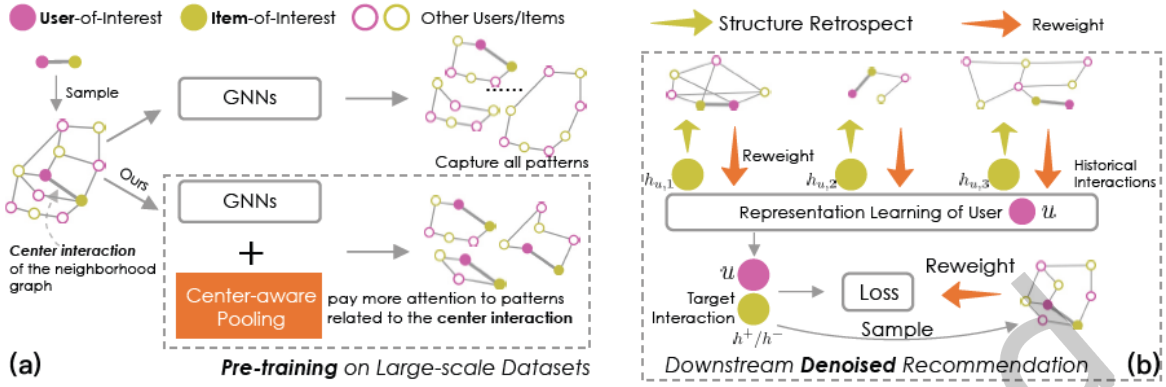

Fig. 2. An illustration of the Structure LEarning based Denoising (SLED) framework. We first pre-train a structure encoder on large-scale recommendation datasets, and learn to capture structural patterns related to the center interaction. Then we transfer the structure encoder to any downstream recommendation dataset to extract related structural patterns for each interaction, based on which we denoise user representation learning from historical interactions and ranking loss calculation for the target interaction.

- We propose the center-aware structure learning which captures related structural patterns for each interaction in its neighborhood graph. We devise the structure-based denoised recommendation, which distinguishes interactions based on structural patterns, and down-weights noisy interactions in user representation learning and ranking loss computation.
- We conduct extensive experiments to show that SLED can improve the recommendation performance by denoising, and discover structures to distinguish noisy/non-noisy interactions[3].

2 RELATED WORKS

2.1 Implicit Feedback

In AI-powered recommender systems [16, 20, 57, 95, 96, 105, 106], implicit feedback, such as click behaviors, is widely used as historical clues to mine users' interests or as positive target interactions to train a recommendation model [33, 78]. Many studies [22, 38, 51, 53, 77] have argued that these signals are noisy, and some of them may not reflect users' inherent interests due to ubiquitous external distractions in recommender systems. Application domains in this research field can vary from news recommendation [51, 53], music recommendation [82, 90], to micro-video recommendation [82]. Most existing works mitigate the negative impact of implicit feedback by introducing external signals, such as dwell time [38, 49, 92], gaze [97], skip [22, 82, 90], scroll intervals [51], like behavior [4], and item side information [51, 53]. In contrast to these methods, we propose to mitigate the negative effects without introducing external signals. Another line of works determines the relative importance of a user's interactions using the target item [21, 60, 89, 99, 100], *i.e.*, the target attention. However, these weights convey to what extent the historical interactions are similar to the target item rather than to what extent each historical interaction is noisy to the user. Moreover, the target item itself can be noisy. Liu *et al.* [50] use GRU to update each item embedding with the user embedding, and then predict the noisy-level weights based on the user-aware item embeddings, which however, still suffer from the noisy supervision signal. Recently, Yu *et*

[3]The code and the noisy e-commerce datasets will be made publicly available to promote future development.

al. [80] found that predictions on noisy items vary across different recommendation models and constructed two models for denoising. Gao *et al.* [25] argue that early-stage training is resistant to noises and mainly denoises the next noise-sensitive stage with memorized clean interactions from the early stage. Different from these works, through our research, we demonstrate the potential of utilizing center interaction-related structural patterns as a key factor in effectively mitigating noise and improving overall denoising outcomes.

2.2 Denoising Models for Recommendation

Research on denoising recommender systems typically identifies malicious noise, often arising from deceptive user activities, and nonmalicious noise, often an upshot of unintentional human errors or random behaviors during item selection. While collaborative filtering recommender systems have seen an expansive body of research devoted to malicious noise detection, the natural noise, innately embedded in user behavior, has only begun to garner academic attention. An exemplary study by Li *et al.*[45] identifies genuine users who may provide some untruthful data by capturing and accumulating user's self-contradictions. Moreover, several investigations have looked into external user feedback [23, 26, 38, 97] and item characteristics [52] as predictors of user satisfaction. Notably, Negative feedback Re-weighting [82] utilized three item types in the training of their recommendation model, treating different items as negative samples with variant weights. While these methodologies offer promising results, they come with the caveat of requiring additional feedback and extensive manual label work, which could render them impractical in scenarios with evolving item pools, such as in news or movie recommendations. Another influential approach resides in selection-based methods [15, 24, 47, 81, 94] and reweighting-based methods [75, 80]. Pioneering efforts, such as WBPR [24] and IR [81], focus on the manipulation of interaction data, either through assigning higher selection probabilities to missing interactions of popular items or reevaluating and changing the labels of negative interactions. Contrastingly, Wang *et al.* [76, 77] find that noisy feedback has large losses in the early stage of training and propose two adaptive loss functions to down-weight noisy samples. We differ from these methods by capturing structural patterns related to the center interaction in the interaction's neighborhood user-item graph as clues for denoising.

2.3 Structure Learning via Pre-training

Traditional network embedding methods, such as LINE [67], DeepWalk [59], node2vec [28], learn node/sub-graph embeddings that are tied up with the given graph for training and cannot deal with the out-of-sample problem. Some works propose to pretrain graph neural networks (GNN) on labeled graphs (*e.g.*, molecular graphs [35]) and apply the GNN to encode unseen graphs. Recent works [87, 104] pay attention to capturing the generic structural representation on unlabeled graphs. Typically, Qiu *et al.*[62] propose to view the structures of sub-graphs as instances and leverage the contrastive learning framework to pretrain the GNN. Peng *et al.*[58] learn to predict the contextual position of nodes relative to others. The proposed center-aware structure learning method in this paper differs from the above methods in the following two aspects: 1) to denoise implicit feedback, we aim to learn structural representations for edges (interactions) rather than nodes, and 2) we are more interested in structural patterns to the center interaction from which the neighborhood structure is sampled and thus devising the multi-hierarchy center-aware pooling.

2.4 Graph Modeling for Recommendation

Recommendation data can be represented as a user-item graph. Recently, it is of increasing interest to incorporate graph neural networks to guide user-item representation learning for recommendation [12, 19, 54, 64, 69, 72, 85, 91, 93, 102], by leveraging high-hop neighbors. Typically, NGCF [79] directly injects the expressive modeling of high-order connectivity into the embedding process and propagates the collaborative signals. LightGCN [32] simplifies NGCF by only keeping the neighborhood aggregation module and linearly propagating user and item

embeddings on the user-item graph. Most graph modeling works in recommendation generally neglect the noisy interactions, which are edges on the user-item graph. Noisy edges would propagate noisy information to direct neighbors and remote neighbors with high-order connectivity. For example, if a user u accidentally clicked a dissatisfied item h, a graph recommendation model that failed to identify such noisy interaction $< u, h >$ on the user-item graph would propagate the interests of user u to represent another user u' that has interacted with item h and inherently likes item h. Obviously, there is an interest discrepancy between user u and user u', and the propagation needs denoising. We show that the proposed SLED could improve graph-based recommendation models by denoising.

3 METHOD

In essence, the ultimate goal of this paper is to leverage related structural patterns to help denoise implicit feedback of interactions for recommendation. An intuitive way is to equip a recommendation model with an additional graph neural network for structure encoding and train them jointly. However, we argue that end-to-end joint training has some drawbacks in real-world recommender systems. Firstly, recommendation models are trained daily or more frequently in industrial recommender systems. Training the structure encoder jointly brings accumulative additional training overload. Empirically, experimental results indicated that the joint training approach required 42.2% additional training time compared to structure encoder being fixed. Secondly, some recommendation datasets might have imbalanced structure distributions and limited data diversity for structure learning. Structure learning on such datasets might incur biased training and overfitting issues. This hypothesis aligns with the findings presented in Section 4.3.5 of our paper. Specifically, we demonstrated that using a more diverse set of structure learning datasets leads to improved performance in our framework.

To bridge the gap, we devise a two-stage framework where we pretrain the structure encoder on large-scale datasets, and transfer it to any downstream recommendation dataset:

- **Center-aware structure learning.** In structure learning, we *pretrain* a structure encoder on multiple large-scale recommendation datasets. Larger-scale pretraining drives the structure encoder to be generalizable across datasets and even domains, and suffers less from biased training and overfitting [18]. During structure learning, we expect that interactions sharing similar structural patterns should be closer in the representation space than the others. Recommendation objectives which aim at user-item matching do not necessarily meet this requirement. Therefore, we propose to leverage self-supervision signals where we define neighborhood graphs with similar structural patterns as positive pairs and regard the others as negatives. A *center-aware* structure encoder captures structural patterns related to the center interaction. To achieve this, we propose to construct multiple hierarchies where we progressively drop unrelated structural patterns from the last hierarchy, enabling more accurate relatedness measurement in the next hierarchy. Details are shown in Section 3.1.
- **Denoised recommendation.** We transfer the pretrained structure encoder to different downstream recommendation datasets for structural pattern extraction without re-training the structure encoder per dataset. Extracted structural patterns are used to determine the reliability of interactions for denoising. There are mainly two major components requiring denoising in modern deep recommendation models. The first component is user representation learning where users are represented from their historical interactions, and noisy interactions will hinder accurate user interest mining. As such, we down-weight noisy interactions in user representation learning *w.r.t.* interactions' reliability predicted from the structural patterns. The second component is loss calculation which is based on the ranking score prediction for a target interaction. Noisy target interactions might cause inaccurate training losses, hurting the optimization of recommendation models. As such, we propose to down-weight the losses of noisy target interactions. Details can be found in Section 3.2.

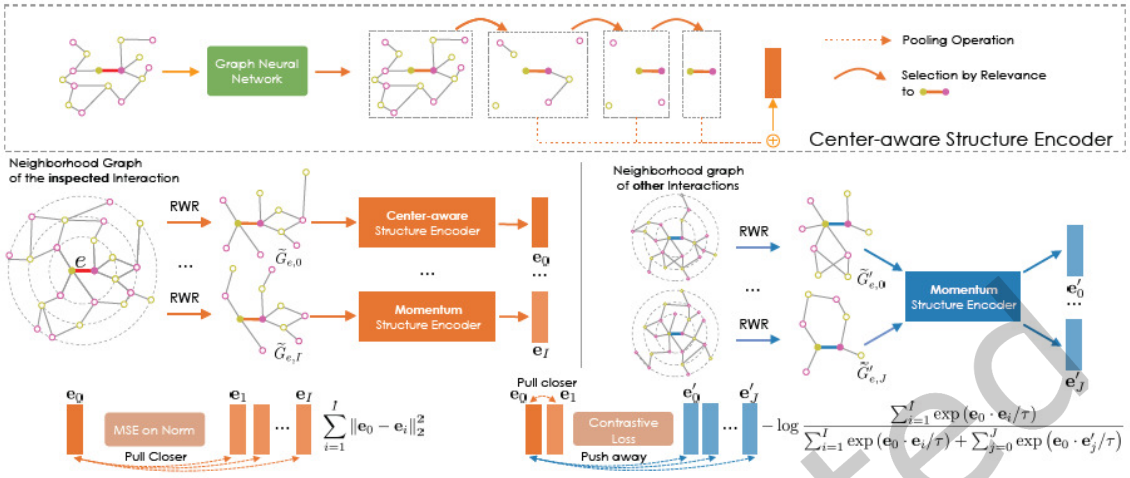

Fig. 3. Schematic of the proposed center-aware structure learning. To capture structural patterns related to the center interaction (edge in bold), we propose to progressively filter unrelated structural patterns from the neighborhood graph. Neighborhood graphs sampled from the same interaction with multi-turn random walk with restart (RWR) share similar structural patterns. We leverage self-supervised learning to pull the representation of similar structural patterns closer than the others.

3.1 Center-aware Structure Learning

In this paper, structure learning aims to learn a generalizable structure encoder, which can extract structural representations for each *interaction* (*i.e.*, edge) in a user-item graph. In the proposed center-aware structure learning, structural representations should encode structural patterns related to the center interaction in its neighborhood graph. Formally, given a sampled neighborhood graph $\widetilde{G}_e$ for interaction e, we aim to learn a graph neural network $f_\theta(\cdot)$ parameterized by θ followed by a non-parametric pooling function $g(\cdot)$ to jointly represent structural patterns related to the **center interaction** e in its neighborhood graph $\widetilde{G}_e$, *i.e.*,

$$\mathbf{e} = g \circ f_\theta(\widetilde{G}_e), \quad \widetilde{G}_e = \text{RWR}(e), \tag{1}$$

where RWR denotes random walk with restart operation for neighborhood graph sampling. The detailed neighborhood graph sampling process for interactions as well as the initial node representation in $\widetilde{G}_e$ can be found in Section 3.1.2.

In the following subsections, we separately introduce the details of the center-aware structure learning, including 1) how we capture the structural patterns that are related to the center interaction from the interaction's neighborhood graph, and 2) how we sample positive/negative neighborhood graphs for interactions and accordingly compute the losses for self-supervised structure learning. The schematic of the proposed method is depicted in Figure 3.

3.1.1 Center-aware Structure Encoding. We employ a graph neural network $f_\theta(\cdot)$ for structure information propagation on the neighborhood graph. In practice, we adopt the Graph Isomorphism Network (GIN) [88] for

its empirical effectiveness. Formally, GIN updates the representation of node v based on its neighbors as follows:

$$\mathbf{v}^{(k)} = \text{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot\mathbf{v}^{(k-1)} + \sum_{\bar{v}\in\mathcal{N}(v)}\bar{\mathbf{v}}^{(k-1)}\right), \tag{2}$$

where $\mathbf{v}^{(k)}$ denotes the representation of node v in the k-th layer, and $\mathcal{N}(v)$ denotes the neighbor nodes of v. We empirically set $\epsilon = 0$ for all layers due to the high performance, which is similar to the findings in [88]. $\text{MLP}^{(k)}$ denotes the multi-layer perceptron in the k-th layer.

Although it is common to incorporate GNNs as the encoder of graph structures, they are mainly designed to encode all the structural patterns within the graph. For example, in the neighborhood graph, *loop* structures containing or not containing the center interaction are considered without making a distinction (*c.f.* Figure 2). To capture structural patterns related to the center interaction, we explicitly measure the distance between structural patterns and the center interaction $< v_1, v_2 >$ in the representation space following Lee *et al.* [44]:

$$S = \{\sum_{\mathbf{v}\in\{\mathbf{v}_1,\mathbf{v}_2\}} \|\mathbf{v}-\tilde{\mathbf{v}}\|_1 \mid \tilde{\mathbf{v}} \in (\mathbf{D})^{-1/2}\,\mathbf{A}\,(\mathbf{D})^{-1/2}\,\mathbf{V}\}, \tag{3}$$

$$\widetilde{V} = \{v_i | i \in \text{bottom-rank}(S, K)\}, \tag{4}$$

where $\mathbf{V}$ is the vectorial representation of nodes V obtained by f_θ. $\mathbf{D}$ denotes the degree matrix, and $\mathbf{A}$ denotes the adjacency matrix. Function bottom-rank(S, K) returns the indices of K nodes with minimum values in S. Intuitively, we regard the updated node representations $(\mathbf{D})^{-1/2}\,\mathbf{A}\,(\mathbf{D})^{-1/2}\,\widetilde{\mathbf{V}}$ after information propagation as structural patterns, and explicitly measure the distance between them and the end nodes $\{\mathbf{v}_1, \mathbf{v}_2\}$ of the center interaction. The structural patterns are thus filtered according to the relevance to both v_1 and v_2. This design is to capture structural patterns related to the edge-of-interest $< v_1, v_2 >$, not particularly related to each connected node. We take K structural patterns $\widetilde{V}$ with the least distance to the center interaction in the representation space as related ones. The center-aware structural representation of the center interaction is obtained as follows:

$$\mathbf{e}_{pool} = \left[\text{avg-pool}(\widetilde{\mathbf{V}}), \text{max-pool}(\widetilde{\mathbf{V}})\right], \quad \mathbf{e} = \mathbf{e}_{pool} / \left\|\mathbf{e}_{pool}\right\|_2, \tag{5}$$

where avg-pool and max-pool denote the average pooling function and max pooling function, respectively. [,] denotes the concatenation of vectors. $\widetilde{\mathbf{V}}$ denotes the vectorial representation of nodes $\widetilde{V}$. $\mathbf{e}$ is the structural representation of the center interaction.

Multi-hierarchy Center-aware Structural Representation. Equation (4) introduces a hyperparameter K for related structural patterns selection. When K becomes smaller, the selected patterns will be more relevant but less comprehensive. As such, choosing one K faces a relevance-completeness trade-off. Moreover, unrelated structure information is also propagated by $(\mathbf{D})^{-1/2}\,\mathbf{A}\,(\mathbf{D})^{-1/2}\,\mathbf{V}$ in Equation (3), probably hindering accurate distance measurement. These challenges drive us to progressively filter unrelated structure information. Specifically, rather than choose one K, we set a sequence $\{K_t\}_{t=1,\ldots,T}$, where $K_{t+1} = \alpha K_t$ and $\alpha \in (0, 1)$. $t = 0$ corresponds to the original graph with no information filtered. $t(t > 0)$ indicates the t-th hierarchy where some unrelated structure information has been filtered from the $(t-1)$-th hierarchy, leading to more accurate distance computation in the t-th hierarchy. In addition, we aggregate the information of different hierarchies as the final center-aware structural representation, without facing the relevance-completeness trade-off in choosing one K. Formally, we have:

$$S_t = \{ \sum_{\mathbf{v} \in \{\mathbf{v}_1, \mathbf{v}_2\}} \|\mathbf{v} - \tilde{\mathbf{v}}\|_1 \mid \tilde{\mathbf{v}} \in (\mathbf{D}_t)^{-1/2} \mathbf{A}_t (\mathbf{D}_t)^{-1/2} \widetilde{\mathbf{V}}_t \}, \tag{6}$$

$$\widetilde{V}_{t+1} = \{v_i | i \in \text{bottom-rank}(S_t, K_{t+1})\}, \tag{7}$$

where $\widetilde{\mathbf{V}}_t$ is the vectorial representation of nodes $\widetilde{V}_t$, and t indicates the t-th hierarchy. $\widetilde{\mathbf{V}}_0$ are obtained by f_θ. The major difference between Equation (6) and Equation (3) lies in the information propagation $(\mathbf{D}_t)^{-1/2} \mathbf{A}_t (\mathbf{D}_t)^{-1/2} \widetilde{\mathbf{V}}_t$ where the information of some unrelated structures are filtered in $\mathbf{D}_t$, $\mathbf{A}_t$, and $\widetilde{\mathbf{V}}_t$ at the t-th hierarchy from $\mathbf{D}_{t-1}$, $\mathbf{A}_{t-1}$, and $\widetilde{\mathbf{V}}_{t-1}$ at the $(t-1)$-th hierarchy. As such, the computed distances S_t for constructing the $(t+1)$-th hierarchy will be more accurate. We combine the pooled representation of different hierarchies as the final center-aware structural representation:

$$\mathbf{e}_{pool} = \sum_{t=1}^{T} \left(\left[\text{avg-pool}(\widetilde{\mathbf{V}}_t), \text{max-pool}(\widetilde{\mathbf{V}}_t) \right] \right), \quad \mathbf{e} = \mathbf{e}_{pool} / \left\| \mathbf{e}_{pool} \right\|_2, \tag{8}$$

where $\widetilde{\mathbf{V}}_t$ is the vectorial representation of nodes $\widetilde{V}_t$ at the t-th hierarchy. $\mathbf{e}$ is the final structural representation of the center interaction with multi-hierarchy center-aware pooling.

Note that we assume node representations encode the structural patterns related to itself, which is a common practice in graph modeling [84, 86]. Specifically, Graph neural networks (GNNs) propagate structural information through graph edges to capture and encode patterns into node representations. These representations summarize the neighborhood information and structural patterns of each node, enabling the modeling of complex dependencies and interactions. In our proposed framework, we aim to capture structural patterns specific to each interaction. To achieve this, we leverage the joint representations of the end nodes involved in an interaction. These representations provide a compact summary of the local graph structure surrounding the interaction. In addition, taking node representations as structural patterns provides us with an efficient and scalable way of encoding the structural information of the interaction graph. Our experiments, including Figure 1, showed that using end node representations as structural patterns could help capture the reliability of interactions and denoise the implicit feedback in the recommendation systems.

3.1.2 Objetives. After training, we expect that neighborhood graphs with similar structural patterns related to the center interaction (positives) should be closer in the representation space than the others (negatives). In this light, we construct positive neighborhood graphs and negative neighborhood graphs for each interaction via neighborhood graph sampling. Then, we leverage self-supervised training where the contrastive learning objective and the mean squared error (MSE) objective together pull positive neighborhood graphs closer to each other in the representation space and push away negative neighborhood graphs.

Neighborhood Graph Sampling for Interactions. Given an edge (**interaction**) $e =< v_1, v_2 >$, we perform random walk with restart (RWR) on its end nodes v_1 and v_2 to have two neighborhood graph $\widetilde{G}_{v_1}$ and $\widetilde{G}_{v_2}$. Then, we combine these graphs as the neighborhood graph of interaction e:

$$\widetilde{G}_e = \text{RWR}(e) = \widetilde{G}_{v_1} \cup \widetilde{G}_{v_2} = \text{RWR}(v_1) \cup \text{RWR}(v_2). \tag{9}$$

Following [62], we initialize the representations of nodes in a given neighborhood graph $\widetilde{G}_e$ using generalized positional embedding. Concretely, with the adjacency matrix of $\widetilde{G}_e$ denoted as $\widetilde{\mathbf{A}}$ and degree matrix as $\widetilde{\mathbf{D}}$, we conduct eigen-decomposition on the normalized graph Laplacian:

$$\mathbf{I} - \widetilde{\mathbf{D}}^{-1/2} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-1/2} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top. \tag{10}$$

We take the top eigenvectors in $\mathbf{U}$ to construct the initial node embeddings, where the number of eigenvectors equals to the smaller of $|\widetilde{G}_e| - 2$ and a pre-defined hyper-parameter d_v. We zero-pad the node embedding matrix to d_v when the dimension is smaller than d_v. As a result, we have a node embedding matrix of dimension $|\widetilde{G}_e| \times d_v$. We further add one-hot encoding of vertex degrees and the binary indicator of the ego vertex according to [62].

Owing to the non-deterministic nature of random walk, we can sample multiple neighborhood graphs $\{\widetilde{G}_{e,i}\}_{i=0,\dots,I}$ for e with multi-turn RWR. These neighborhood graphs have similar structural patterns related to e, and are regarded as positive pairs. Intuitively, the neighborhood graphs of other interactions are more likely to have dissimilar structural patterns, and are potential negative samples. We use $\{\widetilde{G}'_{e,j}\}_{j=0,\dots,J}$ to denote these negative samples *w.r.t.* interaction e.

After the sampling of positive and negative neighborhood graph, we use the structure encoder $g \circ f_\theta(\cdot)$ to extract positive and negative structural representations, *i.e.*,

$$\mathbf{e}_i = g \circ f_\theta(\widetilde{G}_{e,i}), \quad i = 0, \dots, I, \quad \mathbf{e}'_j = g \circ f_\theta(\widetilde{G}'_{e,j}), \quad j = 0, \dots, J, \tag{11}$$

where $\mathbf{e}_i$ denotes the i-th positive structural representation for interaction e, and $\mathbf{e}'_j$ denotes the j-th negative structural representation for interaction e.

Contrastive learning objective. Contrastive learning based framework learns to distinguish positive pairs from negative samples. We adopt the InfoNCE [73] as the loss function:

$$\mathcal{L}_c = -\log \frac{\sum_{i=1}^{I} \exp\left(\mathbf{e}_0 \cdot \mathbf{e}_i / \tau\right)}{\sum_{i=1}^{I} \exp\left(\mathbf{e}_0 \cdot \mathbf{e}_i / \tau\right) + \sum_{j=0}^{J} \exp\left(\mathbf{e}_0 \cdot \mathbf{e}'_j / \tau\right)}, \tag{12}$$

where τ is the temperature hyper-parameter. We use the momentum-updated structure encoder [30] for encoding negative neighborhood graphs and positive neighborhood graphs except for $\widetilde{G}_{e,0}$, which permits efficient training with many samples.

MSE objective for positive samples. In addition to the contrastive learning objective, we also propose to directly minimize the Euclidean distance between multiple positive neighborhood graphs in the representation space following Grill *et al.* [27], *i.e.*,

$$\mathcal{L}_m = \sum_{i=1}^{I} \|\mathbf{e}_0 - \mathbf{e}_i\|_2^2 . \tag{13}$$

Therefore, the final objective can be:

$$\mathcal{L} = \mathcal{L}_c + \beta \mathcal{L}_m. \tag{14}$$

The pipeline of center-aware structure learning is summarized in Algorithm 1.

3.2 Denoised Recommendation

We transfer the learned center-aware structure encoder $g \circ f_\theta(\cdot)$ to any downstream recommendation dataset for denoising. We first employ the pre-trained structure encoder to extract structural representations of interactions, and devise a weight predictor to determine the reliability of interactions based on these structural representations. Then, we down-weight noisy interactions (with lower reliability) in recommendation models. In training recommendation models, interactions are leveraged in two representative manners, *i.e.*, serving as the raw features for user interest mining, and serving as the positive samples for ranking loss calculation, which both require denoising. As such, we devise information gates which control how each interaction affects user interest mining *w.r.t.* the predicted interaction reliability. Then, we devise the re-weighted recommendation objective function such that ranking losses computed from noisy interactions should have less effect on the model optimization.

Algorithm 1: Center-aware Structure Learning

Input: Large-scale recommendation dataset represented in a user-item graph $G = \{V, E\}$
Output: Parameters of the structure encoder θ
Randomly initialize θ
while *not converged* **do**
  Sample e randomly from E
  Sample positive neighborhood graphs $\{\widetilde{G}_{e,i}\}_{i=0}^{I}$ from e with Eq. (9) and multi-turn RWR
  Sample negative neighborhood graphs $\{\widetilde{G}'_{e,j}\}_{j=0}^{J}$ from other interactions with Eq. (9)
  Obtain positive and negative structural representations $\{\mathbf{e}_i\}_{i=0}^{I}$, $\{\mathbf{e}'_j\}_{j=0}^{J}$ with Eq. (11)
  Update θ by minimizing $\mathcal{L}_c + \beta\mathcal{L}_m$ computed with Eq. (12) and Eq. (13)
end

Let $\mathcal{D}$ denote a given recommendation dataset, which contains a set of users $\mathcal{U}$ and a set of items $\mathcal{H}$. Let $e_{u,h}$ denote an interaction between user u and h. We use bold letters to denote the dense representations (*e.g.*, $\mathbf{h}$ as the item embedding). With a slight abuse of notation, we will drop the sub-scripts occasionally and write e in place of $e_{u,h}$ for conciseness.

3.2.1 Interaction Reliability Prediction. For each interaction e, we sample the neighborhood graph $\widetilde{G}_e$ of interaction e (edge) in the user-item graph using RWR. We then extract structural representation $\mathbf{e}$ related to the center interaction e in its neighborhood graph $\widetilde{G}_e$ using the pretrained center-aware structure encoder $g \circ f_\theta(\cdot)$ in a pre-processed manner. Based on interactions' structural representations, we predict interactions' reliability weights using a fully-connected layer ϕ with sigmoid as activation function:

$$r = \phi \circ g \circ f_\theta(\widetilde{G}_e) = \phi(\mathbf{e}) = \sigma\left(\mathbf{W}_r\mathbf{e} + \mathbf{b}_r\right), \tag{15}$$

where $\mathbf{W}_r$ is the weight matrix of the fully-connected layer, and $\mathbf{b}_r$ is the bias term. $\mathbf{e}$ denotes the neighborhood structural representation, and $r \in (0, 1)$ denotes the interaction reliability weight for the interaction e. σ denotes the sigmoid function.

3.2.2 Information Gates. We use the predicted interactions' reliability weights to denoise user representation learning by down-weighting noisy historical interactions of users. The cutting-edge techniques for historical interaction modeling are graph-based [32, 79] and sequence-based [5, 7, 46, 48, 70, 101] which aggregate the information of interacted items to obtain the user representation with consideration on higher-order connection and chronological dependency, respectively. The predicted interaction weights can be used in information gates to control the contribution of interacted items to the user representation. We then illustrate how these interaction weights are leveraged in sequence-based and graph-based methods, respectively.

Sequence-based methods. For a particular user u, we have a sequence of historically interacted items $H_u = \{h_{u,t}\}_{t=1,\ldots,N_u}$ in the ascending order by the time of clicking. Each item associates a trainable embedding $\mathbf{h}_{u,t}$. The sequence modeling component can be formulated as:

$$\mathbf{u} = \psi(\mathbf{H}_u) = \psi(\mathbf{h}_{u,1}, \mathbf{h}_{u,2}, \ldots, \mathbf{h}_{u,N_u}), \tag{16}$$

where $\mathbf{u}$ denotes the user interest representation. Noisy interactions are arguably less representative of users' inherent interests due to the click-bait issue, sale promotions, and suggestions from friends, etc. To pursue denoised user interest representation, we adopt the predicted interaction weights as information gates in historical

interaction modeling. Formally,

$$\widetilde{\mathbf{u}} = \psi(r_{u,h_{u,1}} * \mathbf{h}_{u,1}, r_{u,h_{u,2}} * \mathbf{h}_{u,2}, \ldots, r_{u,h_{u,N_u}} * \mathbf{h}_{u,N_u}), \tag{17}$$

where $r_{u,h_{u,t}}$ is the reliability weight of interaction between u and $h_{u,t}$, and controls the contribution of interactions to the output, similar to the output gate in LSTM [34]. $\widetilde{\mathbf{u}}$ is the denoised interest representation of user u. As for ψ, we choose two sequence-based models, including ComiRec-SA [46] using self-attention mechanisms [17], and Mind [5] using the capsule network in the experiments.

Graph-based methods. Graph-based recommendation models typically represent the recommendation data as a user-item graph. Many of them adopt Graph Neural Networks (GNN) to permit message passing and message aggregation along the edges of the user-item graph. Generally, the k-th layer of GNN for $item \rightarrow user$ and $user \rightarrow item$ propagation can be formulated as follows:

$$\mathbf{a}_u^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_{u,i}^{(k-1)} : h_{u,i} \in \mathcal{N}(u)\right\}\right), \quad \mathbf{u}^{(k)} = \text{COMBINE}^{(k)}\left(\mathbf{u}^{(k-1)}, \mathbf{a}_u^{(k)}\right), \tag{18}$$

$$\mathbf{a}_h^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{u}_{h,j}^{(k-1)} : u_{h,j} \in \mathcal{N}(h)\right\}\right), \quad \mathbf{h}^{(k)} = \text{COMBINE}^{(k)}\left(\mathbf{h}^{(k-1)}, \mathbf{a}_h^{(k)}\right), \tag{19}$$

where operation AGGREGATE aggregates the information of historically interacted items $\mathcal{N}(u)$ for user u or historically interacted users $\mathcal{N}(h)$ for item h. The operation COMBINE updates the user representation $\mathbf{u}$ or the item representation $\mathbf{h}$. Noisy interactions will propagate false information to both one-order neighbors and higher-order neighbors in multi-layer GNNs, which require denoising. Towards this end, we leverage interaction reliability weights as information gates to control the information propagation process on the user-item graph:

$$\widetilde{\mathbf{a}}_u^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_{u,i}^{(k-1)} * r_{u,h_{u,i}} : h_{u,i} \in \mathcal{N}(u)\right\}\right), \quad \widetilde{\mathbf{u}}^{(k)} = \text{COMBINE}^{(k)}\left(\widetilde{\mathbf{u}}^{(k-1)}, \widetilde{\mathbf{a}}_u^{(k)}\right), \tag{20}$$

$$\widetilde{\mathbf{a}}_h^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{u}_{h,j}^{(k-1)} * r_{u_{h,j},h} : u_{h,j} \in \mathcal{N}(h)\right\}\right), \quad \widetilde{\mathbf{h}}^{(k)} = \text{COMBINE}^{(k)}\left(\widetilde{\mathbf{h}}^{(k-1)}, \widetilde{\mathbf{a}}_h^{(k)}\right), \tag{21}$$

where $r_{u,h_{u,i}}$ is the reliability weight of interaction between user u and item $h_{u,i}$. $r_{u_{h,j},h}$ is the reliability weight of interaction between user $u_{h,j}$ and item h.

Discussion. In the generic graph research domain, there are some works that predict the edge weights. For example, Graph Attention Network (GAT) [74] employs attention mechanism to estimate edge weights, which represent the relative importance of direct neighbors and the summation of neighbor edge weights is one. Differently, the interaction weights in our work indicate the absolute importance, and the interactions of a particular node can be all noisy (summation near zero). In addition, GAT predicts the edge weights based on the matching degree of adjacent nodes while we predict the interaction weights based on related structural patterns. To the best of our knowledge, we are among the initiatives to investigate denoising interactions in the graph-based recommendation domain. In the experiments, we successfully enhance two representative graph-based recommendation models, *i.e.*, NGCG [79], and LightGCN [32].

3.2.3 Re-weighted Loss Function. The ranking loss of existing recommendation models is calculated based on the user representation and a target interaction, aiming to maximize the user-item matching score of the target interaction. Maximizing the user-item matching of noisy target interactions might falsely guide the training of recommendation models. In this regard, we propose to leverage interaction weights to re-weight the loss function for denoising. Generally, the re-weighted loss function for recommendation can be defined as:

$$\mathcal{L}_{rec}^{*} = \sum_{(u,h^+)\in\mathcal{D}} \mathcal{L}_{rec}^{*}(u, h^+, \mathcal{H}^-) = \sum_{(u,h^+)\in\mathcal{D}} r_{u,h^+} \mathcal{L}_{rec}(u, h^+, \mathcal{H}^-), \tag{22}$$

where h^+ denotes an item historically interacted by user u, *i.e.*, a positive item, and $\mathcal{H}^-$ denotes the set of (sampled) negative items that the user has not interacted with. $\mathcal{L}$ denotes an original recommendation loss

function. We take the sampled softmax loss (SSM) [36], as an example, which has been used in training many deep recommendation models [5, 46]. It can be formulated as:

$$\mathcal{L}_{SSM} = \sum_{(u,h^+)\in\mathcal{D}} -\log p'(h^+ \mid u), \tag{23}$$

$$p'(h^+ \mid u) = \frac{\exp\left(\varphi\left(u, h^+\right)\right)}{\exp\left(\varphi\left(u, h^+\right)\right) + \sum_{h^-\in\mathcal{H}^-} \exp\left(\varphi\left(u, h^-\right)\right)}, \tag{24}$$

where $\varphi(u, h)$ computes the matching score for user u and item h. We down-weight the training loss of noisy target interactions as follows:

$$\mathcal{L}^*_{SSM} = \sum_{(u,h^+)\in\mathcal{D}} -r_{u,h^+} \log p'(h^+ \mid u). \tag{25}$$

This re-weighting strategy can be easily adapted to other losses, such as BPR loss [65]. In summary, the probability of a interaction being identified as non-noisy in the interaction reliability prediction component determines the contribution of this interaction in the recommendation component; while the re-weighted prediction error outputted in the recommendation component acts as an important signal in the interaction reliability prediction component. These two components can mutually enhance each other for better reliability and recommendation prediction.

3.2.4 Training. For a downstream recommendation dataset $\mathcal{D}$, we first extract structural representations of each interaction using Equation (1) in a pre-processed manner. Given user u and item h^+ sampled from the recommendation dataset $\mathcal{D}$, we predict reliability weights of historical interactions of user u (and also historical interactions of item h^+ for graph-based models) using Equation (15) based on the pre-extracted structural representations. Then, we use the interaction reliability weights to denoise user representation extraction for sequence-based models using Equation (17), and user-item representation extraction for graph-based models using Equation (20)-(21). We leverage the denoised representations to compute the ranking prediction loss $\mathcal{L}_{rec}$. Then, we use the reliability weight of interaction between user u and item h^+ to re-weight the loss and obtain denoised loss $\mathcal{L}^*_{rec}$ with Equation (22). The training process is summarized in Algorithm 2.

3.2.5 Time Complexity. In denoised recommendation, the pre-trained center-aware structure encoder will not be fine-tuned. The additional time complexity introduced by our approach comes from the pre-extraction of structural representations for interactions and the computation of interaction weights using one fully-connected layer in Equation (15).

- *Structural Representation Extraction.* For the user-item graph of a downstream recommendation dataset, let $|E|$ denote the number of edges/interactions, and d denote the hidden size of the structural representation. The graph sampling process yields time complexity $O(M|E|)$ where M is the number of random walk steps in sampling neighborhood graphs. The pretrained center-aware encoder extracts the structural representation per sampled neighborhood graph. As such, during representation extraction, the graph neural network $f_\theta(\cdot)$ has time complexity $O(L|E|Md^2)$, where L denotes the number of graph convolution layers. The multi-hierarchy center-aware pooling $g(\cdot)$ yields $O(L_c(|E|Md + |E|))$ time complexity where L_c is the number of pooling layers.
- *Interaction Reliability Weight Prediction.* The interaction weight prediction layer yields $O(s|E|d)$ time complexity where s denotes the number of training epochs.

Therefore, the structure learning yields $O(L|E|Md^2 + L_c|E|Md + s|E|d)$ additional time complexity (we omit some components since $Md >> 1$ and $Ld >> 1$). A typical graph-based recommender yields $O(L_r s|E|d^2)$ according to NGCF [79], where L_r is the number of graph convolution layers. As such, the time complexity of the proposed

Algorithm 2: Denoised Recommendation

Input: Downstream recommendation dataset $\mathcal{D}$, which can be represented as a user-item graph $G_d = \{V_d, E_d\}$; pretrained structure encoder $g \circ f_\theta(\cdot)$
Output: Parameters of the recommendation model Ω; the interaction weight predictor ϕ
// Structural Representation Extraction
for $e \in E_d$ **do**
  Sample neighborhood graph $\widetilde{G}_e$ for e with Eq. (9)
  Obtain structural representation $\mathbf{e} = g \circ f_\theta(\widetilde{G}_e)$, and cache $\mathbf{e}$
end
// Denoised Recommendation
Randomly initialize Ω
while *not converged* **do**
  Sample user u, positive item h^+, and negative items $\mathcal{H}^-$ from $\mathcal{D}$
  if *sequence-based recommendation model* **then**
    Extract denoised user representation with Eq. (17)
  else if *graph-based recommendation model* **then**
    Extract denoised user-item representation with Eq. (20)-(21)
  Update Ω and ϕ by minimizing $\mathcal{L}^*_{rec}$ computed by Eq. (22)
end

Table 1. Statistics of datasets. $\mathcal{U}$, $\mathcal{I}$, and $\mathcal{E}$ denote the users, items, and interactions of a dataset.

(a) Structure learning datasets and public recommendation datasets.

	Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{E}\rvert$	Density
Structure Learning	User Behaviour	987,994	4,162,024	100,150,807	0.002%
	MIND	161,013	317,080	24,155,470	0.047%
Public Rec. Benchmark	User Behaviour (Small)	4,062	4,777	29,377	0.15%
	Movielens-1M	6,040	3,706	1,000,209	4.47%
	Amazon-Beauty	40,226	54,542	353,962	0.02%

(b) A noisy industrial dataset collected from Taobao in consecutive 5 days within a product promotion festival.

		Train	Test			
$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{E}_{day1}\rvert$	$\lvert\mathcal{E}_{day2}\rvert$	$\lvert\mathcal{E}_{day3}\rvert$	$\lvert\mathcal{E}_{day4}\rvert$	$\lvert\mathcal{E}_{day5}\rvert$
1144	6839	11310	7817	7554	6912	6058

denoised recommendation is comparable to the time complexity of a graph-based recommendation model and affordable in practice. Empirically, the major part of structure-based denoised recommendation, *i.e.*, pre-extraction of structural representations, takes 1.5 minutes and 80 minutes for all interactions on the Taobao User Behavior dataset and MovieLens-1M dataset, respectively, while the training of NGCF takes 3.7 minutes and 169 minutes on these two datasets.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Structure Learning. To obtain a generalizable structure encoder, we conduct structure learning on two large-scale recommendation datasets, *i.e.*, Taobao User Behaviour [103], and MIND [83]. The basic statistics of structure learning datasets can be found in Table 1a.

- **User Behavior [103].** This is a large-scale recommendation dataset with logged data from Taobao recommender systems. We download this dataset from their official website[4], and take the click behavior data for pretraining.
- **MIND [83].** This is a large-scale news recommendation dataset collected from MSN News logs in one month. This dataset contains sampled 1 million users who had at least 5 news click records during 6 weeks from October 12 to November 22, 2019. We download the training set from their official website[5] for pretraining.

We train for 468, 750 steps with mini-batch size of 32 and use Adam [39] for optimization with learning rate of 0.005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, weight decay of $1e-4$, learning rate warm-up over the first 7, 500 steps, and linear decay of the learning rate. The pre-defined size of initial node embeddings is set to $d_v = 32$. Gradient norm clipping is applied with range $[-1, 1]$. The negative queue size for MoCo-style contrastive learning is 16,384. The temperature τ is set as 0.07, and the momentum m is set as 0.999. We adopt GIN [88] as the backbone of the center-aware encoder with 5 graph convolution layers with 64 hidden units per layer and 5 pooling layers. α in multi-hierarchy center-aware structural representations is tuned in the range of $\{0.3, 0.4, 0.5, 0.6, 0.7\}$.The parameters of the structure encoder will be frozen during the training of recommendation models.

4.1.2 Recommendation. We evaluate the proposed SLED on three real-world recommendation datasets, *i.e.*, Taobao User Behavior dataset, MovieLens [29], and Amazon Beauty [31, 55]. These datasets differ in domains and densities. Recommendation dataset statistics are listed in Table 1a.

- **User Behaviour [103].** We use a sampled version of the original User Behaviour dataset to speed up the evaluation.
- **MovieLens [29].** MovieLens is a widely used benchmark for evaluating recommendation models. We adopt a well-established version, *i.e.*, MovieLens-1M[6], to evaluate SLED on datasets with high density (4.79%).
- **Amazon Beauty.** We take the *Beauty* category of the Amazon[7] dataset [31, 55] for evaluation, which has a relatively low density (0.02%).

Evaluation Protocol and Metrics. We mainly follow the *leave-one-out* evaluation protocol, which has been widely used in [33, 37, 66, 68, 98]. Concretely, we chronologically sort the historical interactions of each user and take the last interaction for testing. We take the interaction just before the last interaction for validation, and view the remaining interactions for training. Importantly, different from [66, 98] that pair the testing item with a sampled set of random items to speed up the metric computation, which is found to be less effective for evaluation by [41], we choose to rank the testing item and the full item set for each user. Following existing works [32, 79], we adopt *Recall* and *Normalized Discounted Cumulative Gain* (NDCG) as the evaluation metrics. Metrics are computed based on the top 20/60/100 recommended candidates. For brevity, we occasionally denote Recall@20 and NDCG@20 as R@20 and N@20, respectively. Higher scores demonstrate better recommendation performance for all metrics.

[4]https://tianchi.aliyun.com/dataset/dataDetail?dataId=46
[5]https://msnews.github.io/#getting-start
[6]https://grouplens.org/datasets/movielens/1m/
[7]http://jmcauley.ucsd.edu/data/amazon/

Table 2. Model performance on three public datasets. We use SLED to enhance two graph-based models (NGCF, LightGCN). The baseline method N2W predicts interaction weights from the end node embeddings of each edge, similar to GAT [74]. SLED predicts the interaction weights based on interactions' structural patterns in the neighborhood graph. All values below are percentages with '%' omitted. We run SLED and the base model independently for five times and * indicates the performance improvement over the base model is statistically significant with $p < 0.01$ under t-tests.

Datasets	Methods	NGCF	+ N2W	+ SLED	LightGCN	+ N2W	+ SLED
Taobao	R@20	3.99	3.67	5.54*	5.44	5.76	6.02*
	R@60	8.03	8.59	11.87*	10.46	11.82	11.48*
	R@100	11.37	11.96	16.12*	14.45	16.16	15.64*
	N@20	1.67	1.51	2.29*	2.46	2.67	2.79*
	N@100	2.97	2.98	4.17*	4.06	4.52	4.50*
MovieLens	R@20	11.19	11.11	11.37	13.96	13.46	14.35*
	R@60	23.76	24.59	24.92*	29.98	28.81	30.43*
	R@100	33.79	34.39	34.55*	40.10	38.96	40.55*
	N@20	4.41	4.41	4.46	5.49	5.23	5.59*
	N@100	8.41	8.55	8.57*	10.19	9.79	10.27*
Amazon	R@20	5.08	5.75	5.90*	4.88	4.88	5.14*
	R@60	9.33	10.41	10.55*	8.78	8.76	9.16*
	R@100	11.95	13.21	13.39*	11.33	11.28	11.64*
	N@20	2.12	2.47	2.55*	2.07	2.08	2.21*
	N@100	3.36	3.81	3.90*	3.22	3.22	3.38*

Base Models. Since the proposed SLED is model-agnostic, we reveal its effectiveness on two kinds of state-of-the-art base recommendation methods, including two graph-based methods (NGCF [79], LightGCN [32]), and two sequence-based methods (Mind [46], ComiRec [5]).

- **NGCF [79]**. A graph-based method that incorporates the graph convolutional network [40] to generate the user/item embedding from the user-item graph. We use the official implementation[8] and use the default hyper-parameters.
- **LightGCN [32]**. LightGCN simplifies and improves the NGCF by linearly propagating user/item embeddings on the user-item interaction graph. We use the official implementation[9] and use the default hyper-parameters.
- **Mind [46]**. Mind transforms the historically interacted user sequence to multiple vectors to encode the different aspects of the user's interests. We use this implementation[10] which can be directly comparable to ComiRec and use the default hyper-parameters.
- **ComiRec-SA [5]**. ComiRec-SA employs the self-attention mechanism to generate multiple interest vectors and proposes a controllable factor to balance the recommendation accuracy and diversity. We download their official codebase[10] and use their default hyper-parameters.

4.2 Overall Performance Analysis

4.2.1 ***Can SLED enhance different recommendation architectures?*** To answer this question, we apply SLED to multiple state-of-the-art architectures. As for the baseline method, we borrow the idea from Graph

[8]https://github.com/huangtinglin/NGCF-PyTorch
[9]https://github.com/gusye1234/LightGCN-PyTorch
[10]https://github.com/THUDM/ComiRec

Table 3. Model performance on three public datasets. We use SLED to enhance two sequence-based base models (Mind, ComiRec-SA). The baseline method N2W predicts interaction weights from the item embedding with sequence context. SLED predicts the interaction weights based on interactions' structural patterns in the neighborhood graph. All values below are percentages with '%' omitted. We run SLED and the base model independently for five times and * indicates the performance improvement over the base model is statistically significant with $p < 0.01$ under t-tests.

Datasets	Methods	Mind	+ N2W	+ SLED	ComiRec	+ N2W	+ SLED
Taobao	R@20	1.84	1.92	2.24*	2.24	2.16	2.52*
	R@60	7.60	7.55	7.7	7.85	7.17	8.34*
	R@100	12.68	12.61	13.12*	12.95	11.82	13.04
	N@20	0.57	0.60	0.68*	0.72	0.68	0.82*
	N@100	2.45	2.46	2.55*	2.58	2.36	2.66
MovieLens	R@20	8.56	8.39	8.86	10.36	10.10	10.07*
	R@60	23.76	23.56	26.49*	28.73	28.33	29.74*
	R@100	38.49	38.31	44.32*	46.71	45.86	49.65*
	N@20	2.94	2.86	3.16*	3.57	3.45	3.62
	N@100	8.14	8.05	9.31*	9.86	9.66	10.47*
Amazon	R@20	1.88	1.71	1.99*	2.12	2.14	2.25*
	R@60	5.47	5.10	5.58*	6.10	6.31	6.50*
	R@100	8.81	8.29	9.15*	10.00	10.13	10.52*
	N@20	0.65	0.59	0.69	0.71	0.72	0.76*
	N@100	1.86	1.74	1.93*	2.08	2.11	2.20*

Attention Network (**GAT**) [74], which predicts edge weight from the end node embeddings of each edge, for graph-based methods. For sequence-based methods, the baseline predicts interaction weights from the item embeddings in the sequence. For brevity, we denote this baseline as Node-to-Weight (N2W). Experimental results on graph-based (NGCF, LightGCN), and sequence-based (Mind, ComiRec) are listed in Table 2 and Table 3, respectively. According to the results, we have the following key observations:

- In a nutshell, we can observe a consistent improvement of SLED-enhanced models over both the graph-based base models and sequence-based base models across the datasets with various domains and densities, which indicates that the SLED can help to denoise implicit feedback and improve recommendation quality.
- In many cases, the strategy of N2W leads to a performance drop or minor improvement, especially from 0.4010 to 0.3896 for Recall@100 of LightGCN, from 0.4671 to 0.4586 for Recall@100 of ComiRec-SA on the MovieLens dataset, and 0.0881 to 0.0829 for Recall@100 of Mind on the Amazon Beauty datasets. This is probably because, during recommendation training, node representation contains complex semantics, such as recommendation-specific collaborative filtering signals. Directly using heterogeneous semantics for interaction reliability weight prediction might overfit the given dataset and fail to determine the reliability of interactions. In contrast, we conduct structure learning as large-scale pretraining, where we disregard recommendation-specific supervision signals and content features, and leverage self-supervision to learn pure structural representations. Such structure disentanglement contributes to the effectiveness of SLED.
- For graph-based methods, the improvements brought by SLED are more significant when applying it to datasets with low density (*e.g.*, User Behaviour, and Amazon Beauty) than the dataset with high density (*i.e.*, MovieLens). In datasets with low density, the interactions of a specific user are fewer, and users might be more sensitive to the noise of interactions. Moreover, due to the higher-order information propagation

[10]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

Table 4. Comparison between SLED and other structure-based baseline methods, including Line2Vec (L2V) [3] and AdaBipartite (AdaB), which is an extension of the original Adamic/Adar method [1].

Methods	R@20	R@60	R@100	N@20	N@100
NGCF	3.99	8.03	11.37	1.67	2.97
NGCF (AdaB)	3.90	8.76	12.15	1.59	3.06
NGCF(L2V)	3.94	8.48	12.02	1.61	3.04
NGCF(SLED)	**5.54**	**11.87**	**16.12**	**2.29**	**4.17**
LightGCN	5.44	10.46	14.45	2.46	4.06
LightGCN (AdaB)	5.32	10.81	14.87	2.36	4.06
LightGCN (L2V)	5.22	11.06	15.09	2.25	4.01
LightGCN (SLED)	**6.02**	**11.48**	**15.64**	**2.79**	**4.50**
Mind	1.84	7.60	12.68	0.57	2.45
Mind(AdaB)	2.11	**7.79**	12.93	0.66	2.53
Mind(L2V)	2.07	7.28	12.89	0.62	2.48
Mind(SLED)	**2.24**	7.75	**13.12**	**0.68**	**2.55**
ComiRec-SA	2.24	7.85	12.95	0.72	2.58
ComiRec-SA(AdaB)	2.07	7.55	12.25	0.66	2.43
ComiRec-SA(L2V)	2.09	7.79	12.44	0.68	2.49
ComiRec-SA(SLED)	**2.52**	**8.34**	**13.04**	**0.82**	**2.66**

in graph-based models, information passing through noisy interactions will affect a broader range of remote nodes, making the problem more severe. SLED successfully denoises implicit feedback and the information propagation along noisy interactions, thus achieving more performance gains. The results of SLED-enhanced graph models demonstrate the effectiveness of the proposed framework in denoising implicit feedback.

- An interesting finding is that, when the base models themselves already obtain high recommendation performance, the improvements brought by SLED become more significant, such as from 0.0399 to 0.0554 *w.r.t.* Recall@20 and base model NGCF on the User Behavior dataset, from 0.4671 to 0.4965 *w.r.t.* Recall@100 and base model ComiRec-SA on the MovieLens dataset, and 0.1195 to 0.1339 *w.r.t.* Recall@100 and base model NGCF on the Amazon Beauty dataset. These results demonstrate that the proposed SLED can better boost the performance of well-trained/well-fitted methods. This is probably because, in such methods, the performance bottleneck lies in the noises of recommendation datasets rather than the model's fitting capability. As such, denoising with SLED achieves more performance gains. Further boosting a high-capacity recommendation model could be a practical merit.

4.2.2 ***How does SLED perform compared to other structure-based baselines?*** we take the following two structure-based baselines for comparison:

AdaBipartite (AdaB). We design the AdaB method which extends the Adamic/Adar [1]. Adamic/Adar is originally not applicable to the user-item graph. AdaBipartite can be formulated as,

$$A(u^*, N(h^*)) = \frac{1}{\|N(h^*)\|} \sum_{u \in N(h^*)} \sum_{h \in N(u) \cap N(u^*)} \frac{1}{\log \|N(h)\|}, \tag{26}$$

Table 5. Comparison between SLED and a state-of-the-art denoising baseline, Wang *et al.* [77], which down-weight the target interactions with large loss values.

Models	R@20	R@60	R@100	N@20	N@100
NGCF	3.99	8.03	11.37	1.67	2.97
NGCF (Re-BPR, $\beta = 0.25$)	3.61	7.91	11.05	1.64	2.46
NGCF (Re-BPR, $\beta = 0.5$)	4.19	8.85	12.86	1.85	3.37
NGCF (Re-BPR, $\beta = 1.0$)	4.19	9.49	13.71	1.82	3.50
NGCF (LCD)	4.95	11.13	14.72	1.87	3.61
NGCF (SLED)	**5.54**	**11.87**	**16.12**	**2.29**	**4.17**

$$A(N(u^*), h^*) = \frac{1}{\|N(u^*)\|} \sum_{h \in N(u^*)} \sum_{h \in N(h) \cap N(h^*)} \frac{1}{\log \|N(u)\|}, \tag{27}$$

$$A(u^*, h^*) = \left(A(u^*, N(h^*)) + A(N(u^*), h^*)\right)/2, \tag{28}$$

where $N(h^*)$ denotes the direct neighbors of node h^* and $A(N(h^*), u^*)$ denotes the proximity of node u^* and $N(h^*)$. For each neighbor node $u \in N(h^*)$ (other than u^*), based on the common neighbors of node u and u^*, *i.e.*, $N(u) \cap N(u^*)$, we compute the accumulation of $\frac{1}{\log \|N(h)\|}$ for each common neighbor h. Intuitively, two nodes are of high proximity when they have more common neighbors, and when each common neighbor has fewer other neighbors. Similarly, the interaction between user u^* and item h^* can be more reliable when the user u^* has more commonly clicked items with other users clicking item h^*, and when each commonly clicked item h is clicked by fewer other users. We take the proximity $A(u^*, h^*)$ as the interaction reliability weight.

Line2Vec [3] (L2V). Line2Vec is a task-independent unsupervised edge embedding framework that learns continuous representations for edges. They argue that simply aggregating the embeddings of the end nodes associated with the edge as edge embeddings is suboptimal and propose the collective homophily to embed a line graph based on pairwise homophily [56]. We leverage the edge embeddings generated by Line2Vec in the same way as our structural representations. We download their official codebase[11] and use their default hyper-parameters. The comparison results on the User Behavior dataset are listed in Table 4. We can see that:

- In many cases, structure-based methods (AdaB, L2V, SLED) can boost the performance of various base models (NGCF, LightGCN, Mind, and ComiRec-SA). Such performance gains demonstrate that structure patterns can help distinguish interactions into noisy and non-noisy ones, and better represent users' interests by denoising.
- The proposed center-aware structure learning still yields a large-margin improvement over AdaB/L2V. AdaB only captures simple patterns/dependencies, such as common neighbor counting normalized by the number of neighbors for each common neighbor. The main objective of Line2Vec is to minimize the distances between one edge and its neighbors to enforce collective homophily. These methods might not capture useful structural patterns for interaction reliability weight prediction.
- On datasets with high density (*i.e.*, MovieLens), Line2Vec quickly runs out of memory on a computing server with 330 GB RAM, suggesting that it may not be applicable to heterogeneous real-world recommendation scenarios.

4.2.3 *How do SLED perform compared to the state-of-the-art denoising implicit feedback methods.*

To answer this question, we choose to compare the proposed SLED with three state-of-the-art baselines, *i.e.*, Re-BPR [77], and LCD [13] [61]. As for Re-BPR, we follow the Reweighted loss as proposed in [77], which

[11] https://github.com/anirban-code-to-live/line2vec

Table 6. Ablation study by progressively adding the proposed components to the NGCF base model.

Models	R@20	R@60	R@100	N@20	N@100
NGCF	3.99	8.03	11.37	1.67	2.97
+BStruct	3.89	8.62	12.11	1.56	3.02
+Ca	4.16	8.74	12.75	1.66	3.17
+ReL	**5.54**	**11.87**	**16.12**	**2.29**	**4.17**

down-weights the positive interactions with large loss values, to construct the baseline method. Specifically, Reweighted BPR (Re-BPR) loss can be formulated as follows:

$$\omega_{u,h^+} = \sigma(\hat{y}_{u,h^+})^{\beta}, \quad \omega_{u,h^-} = (1 - \sigma(\hat{y}_{u,h^-}))^{\beta}, \tag{29}$$

$$\mathcal{L}_{BPR} = \sum_{(u,h^+,h^-)\in\mathcal{D}} - \ln \sigma \left(\omega_{u,h^+} * \hat{y}_{u,h^+} - \omega_{u,h^-} * \hat{y}_{u,h^-}\right). \tag{30}$$

To be consistent with Wang *et al.* [77] who require that the prediction should be within $(0, 1)$, we add the sigmoid function σ to the prediction $\hat{y}_{u,h}$ when computing ω. For hyper-parameter β, we follow their work to choose $\beta \in \{0.25, 0.5, 1.0\}$, and report all results. As for LCD, we strictly follow the steps outlined in the original paper. The results on the User Behavior dataset with NGCF as the base model are listed in Table 5.

- We observe performance gains of Re-BPR and LCD over the base method. These results indicate that the absolute loss magnitude and the batch-level relative loss magnitude might be correlated to the noisy level of the corresponding sample to some extent, which is a key assumption of Re-BPR and LCD, respectively.
- Both Re-BPR, LCD and SLED improve the base method, which shows that loss correction (*e.g.*, re-weighting) with effective weights can help mitigate the negative effects of noisy interactions.
- When β is smaller (such as 0.25), which means the penalty is heavier, Re-BPR may hurt the effectiveness of the base recommendation model. These results probably indicate that penalizing interactions with large losses may be at risk of penalizing hard positive samples such as fresh items for a particular user.
- The large-margin improvement of SLED over Re-BPR and LCD demonstrates the effectiveness of the proposed structure learning based denoising.

4.3 Model Analysis

4.3.1 ***Does all proposed components/strategies improve the overall effectiveness (Ablation Study)?*** To answer this question, we surgically and progressively add the proposed modules to the base model and test the performance of different architectures. According to the results listed in Table 6, we can observe that removing any proposed component will lead to a performance drop. To be specific, we have the following key observations:

- **+BStruct**, which means adding the base structure learning method without center-aware pooling for structural pattern extraction. We leverage the structural representations to predict interaction weights as in Section 2.2.1, and use these weights as the information gates to denoise user-item representation learning in NGCF as in Section 2.2.2. The performance improvement *w.r.t.* R@20 and R@100 indicates that structure learning is helpful for interaction reliability determination.
- **+Ca**, which means adding the multi-hierarchy center-aware pooling to +BStruct (*c.f.* Section 3.1.1). The improvement demonstrates the merits of center-awareness and rationality of our analysis. Recall that the neighborhood graphs of interactions of a particular user have overlapped structural patterns. Capturing structural patterns related to the center interaction is the key to distinguishing different interactions of a particular user for denoising.

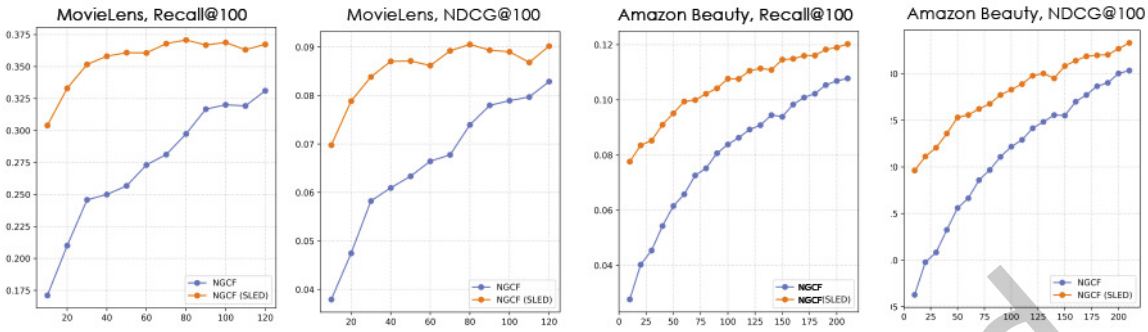

Fig. 4. Recommendation performance on ***non-noisy*** testing samples (with rating > three) across epochs.

- **+ReL**, which means further using the interaction reliability weights to denoise the recommendation objective function (*c.f.* Section 3.2.3) besides denoising the user-item representation learning. Loss denoising down-weights the effect of false-positive target interaction on the training of the recommendation model. We observe a large-margin improvement, which further reveals the effectiveness of the structure-learning based denoising.

4.3.2 ***Performance on Non-noisy/High-rating Recommendations***. We refer to recommendations where users will click the corresponding items and further give high ratings to these items after consumption as non-noisy recommendations. Denoised models are expected to be less affected by noisy interactions in the historical data, and make more non-noisy recommendations to users in the future. In this regard, we are interested in whether SLED can improve the performance of non-noisy recommendations. In this regard, we take an in-depth look at the evaluation results of testing samples with ***rating over three*** on the MovieLens and Amazon Beauty datasets, and plot the testing performance of the NGCF base model and the SLED-enhanced version across different training epochs in Figure 4. We can find that:

- SLED consistently improves the Base model in different training epochs by a large margin. For instance, at epoch 10, the SLED-enhanced model improves the base model from 0.171 → 0.304 on the MovieLens dataset and 0.028 → 0.078 on the Amazon Beauty dataset, *w.r.t.* Recall@100. These results indicate that SLED can help models to identify interactions that better reflect users' inherent interests, leading to more effective users' interest representations and more non-noisy recommendations. They demonstrate the effectiveness of SLED in denoising implicit feedback in recommender systems and improving the recommendation quality.
- We notice that base model takes nearly 100 epochs to have comparable performance to the performance of the SLED-enhanced model at epoch 10. In other words, the SLED-enhanced model achieves **fast convergence**, which is a critical merit in industrial recommender systems. We attribute the fast convergence to that SLED helps the base model to quickly identify noisy interactions based on structural patterns at the early stage of training. Then, by leveraging non-noisy interaction to represent users and items, and to train the entire recommendation model, the base model could quickly improve its recommendation performance without being affected by noisy data. Still, at convergence, the SLED-enhanced models achieve higher performance than the Base model.

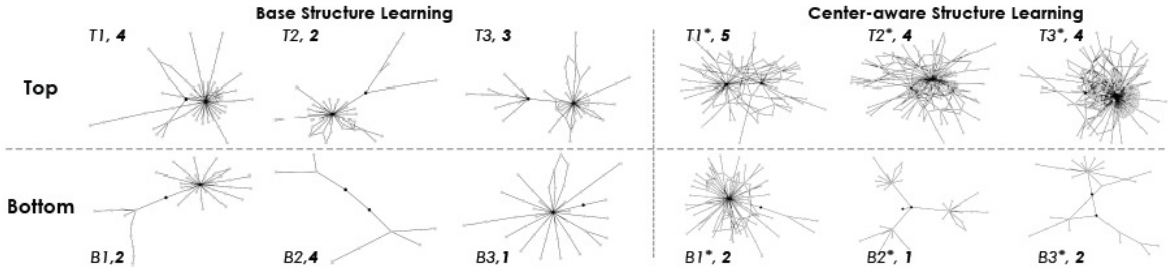

Fig. 5. Case study on the neighborhood graphs of interactions with highest reliability weights (Top) and lowest reliability weights (Bottom) predicted by the base structure learning method without center-aware pooling, and center-aware structure learning in SLED. We also report the ground-truth rating of the interaction-of-interest in bold. SLED determines the reliability of interactions based on structural patterns related to the center interaction rather than all structural patterns in the neighborhood graph. As such, although there are many connected components in B1* analogous to T1*/T2*/T3*, the center interaction in B1* is regarded as noisy.

4.3.3 ***Case Studies.*** We are interested in whether center-aware structure learning and the interaction reliability weight prediction can jointly discover useful structural patterns for denoising. Towards this end, we propose to visualize the top three neighborhood graphs with the highest reliability weights and the bottom three with the lowest reliability weights. As a comparison, we also visualize the results of the base structure learning method without center-aware pooling. We mark the end nodes of the center interaction from which the neighborhood graph is sampled in bold style. Based on the results depicted in Figure 5, we can see that:

- We find that when the neighborhood graph has more connected components, the center interaction is more likely to be determined as a non-noisy one, which is related to the findings in social networks [71]. In connected components, many users click the same item and many items are clicked by the same user, indicating that users and items share similarities. The center interaction might be less noisy due to such similarities in the neighborhood graph.
- An interesting finding is that the SLED with center-awareness regards the center interaction of neighborhood graph $B1*$ as a noisy interaction. The neighborhood graph $B1*$ also has many connected components similar to top-weighted neighborhood graphs $T1*$, $T2*$, and $T3*$. However, different from the top-weighted neighborhood graphs, the connected components in $B1*$ hardly include the center interaction as a member. The center-aware structure encoder in SLED regards such connected components as structural patterns unrelated to the center interaction and disregards them in determining the reliability of the center interaction. In contrast, the base structure learning model without center-aware awareness assigns high reliability weights to T2 and T3, which contain many connected components that do not include the center interaction as a member. These results and the ablation study in Table 6 jointly demonstrate that SLED successfully captures structural patterns related to the center interaction with multi-hierarchy center-aware pooling, and eventually improves the recommendation performance by center-aware structure-based denoising.
- To further support our findings, we also report the ground-truth ratings of the interaction-of-interest. The observations from these ratings are consistent to the above analysis and overall, SLED better captures the reliability of interactions than the base structure learning model. These results further demonstrate the rationality of our design.

Table 7. Analysis of fine-tuning SLED on downstream datasets.

Datasets	Methods	R@20	R@60	R@100	N@20	N@100
MovieLens	Base model	11.19	23.76	33.79	4.41	8.41
	SLED	11.37	24.92	34.55	4.46	8.57
	SLED (fine-tuned)	11.38	24.87	34.95	4.52	8.71
Amazon	Base model	5.08	9.33	11.95	2.12	3.36
	SLED	5.90	10.55	13.39	2.55	3.90
	SLED (fine-tuned)	5.86	10.39	13.16	2.55	3.86

Table 8. Analysis of the choices of structure learning datasets.

Pretraining Datasets	R@20	R@60	R@100	N@20	N@100
User Behavior + MIND	5.54	11.87	16.12	2.29	4.17
User Behavior	5.40	11.37	15.45	2.14	3.96
MIND	5.08	10.97	15.13	2.01	3.86

4.3.4 ***Structure fine-tuning on downstream datasets.*** We are interested in how fine-tuning SELD on downstream recommendation datasets affects the model performance. According to the results listed in Table 7, we observe that there is no significant performance change after fine-tuning where the performance is consistently better than the base model. These results indicate the rationality of performing structure learning on large-scale datasets, capturing basic and transferrable structural patterns for denosing recommendation.

4.3.5 ***Analysis of the structure learning datasets.*** We are interested in how using different structure learning datasets affect thes the model performance. In this regard, we conduct an additional experiment to investigate the impact of using just one dataset for structure learning as opposed to the two large-scale recommendation datasets initially used in our study (*cf.* Table 8). Our findings from this new experiment are twofold: Firstly, we observed that using both datasets for structure learning yielded consistently superior performance compared to using only one. This could potentially be attributed to the enriched diversity in structures from different domains that are introduced when multiple datasets are employed, leading to a greater generalization capacity for the structure learning method. Secondly, we found that when a single dataset was used, the performance was generally better when we used the larger dataset (User Behavior dataset) as compared to the smaller one (MIND dataset). This suggests that the pretraining dataset size might contribute to the effectiveness of the structure learning process.

4.3.6 ***Analysis of choosing K in multi-hierarchy center-aware structural representation.*** We note that, in SLED, multi-hierarchy center-aware structural representation is a critical component that captures structural patterns particularly important to the edge of interest (*c.f.* Section 3.1). We are interested in how choosing a fixed number and a dynamic number of structural patterns differ in performances. In this light, we vary K in the range of $\{2, 4, 8, 16\}$ and test the denoised recommendation performances on the Taobao user behavior dataset. According to the results listed in Table 9, we observe that choosing a dynamic number of structural patterns consistently outperforms choosing a fixed one with different K values. These results further demonstrate the rationality of the SLED design in capturing multi-hierarchy structural patterns for center-aware structure learning.

4.3.7 ***Evaluation on In-domain Downstream Datasets.*** The efficacy of SLED, when applied to datasets featuring identical and independently distributed structures analogous to the pretraining datasets, is presented in

Table 9. Comparison between fixed K and dynamic K in multi-hierarchy center-aware structural representations. D stands for choosing dynamic K as illustrated in Section 3.1.1.

K	R@20	R@60	R@100	N@20	N@100
D	5.54	11.87	16.12	2.29	4.17
2	5.02	10.66	15.24	2.10	3.91
4	5.32	11.15	15.31	2.08	3.85
8	5.12	11.45	15.24	2.03	3.85
16	5.27	11.50	15.56	2.10	3.93

Table 10. Evaluation on In-domain Downstream Datasets.

Downstream Datasets	Models	R@20	R@60	R@100	N@20	N@100
User Behavior	Base Model	3.99	8.03	11.37	1.67	2.97
	SLED	5.54	11.87	16.12	2.29	4.17
MIND	Base Model	7.51	15.38	2.10	3.04	5.44
	SLED	7.70	16.10	2.19	3.10	5.62

Table 10. In this study, we elected to jointly pretrain SLED using the User Behavior and MIND datasets, subsequently evaluating its capability to denoise recommendations within these individual datasets. Our observations reveal that SLED's performance excelled particularly in the User Behavior dataset, which contained a significantly larger volume of interaction data during the pretraining phase, as compared to its performance on the MIND dataset. This outcome corroborates our findings presented in Table 8, where excluding the User Behavior dataset from the pretraining process resulted in a pronounced decline in performance metrics. A plausible explanation for this observable trend could be that product click behavior in the e-commerce landscape may be inherently more susceptible to noise in comparison to the consumption behavior on news platforms. As such, the process of denoising could potentially yield substantial performance improvements. Despite these variations, SLED consistently demonstrated a capacity to augment performance across both datasets, further evidencing its robust effectiveness.

4.4 Performance on Highly Noisy Industrial Datasets

We introduce a noisy e-commerce dataset collected from Mobile Taobao from June 18 to June 23. Such a period is one of the biggest annual product promotion festivals in China. The data statistics are listed in Table 1b. This dataset can be noisy since the promotion strategies of the platform and sales strategies of shop owners are external distractions for users, resulting in noisy interactions beyond users' inherent interests. Moreover, such strategies can vary significantly from day to day where overfitting to noisy interactions will incur low-quality recommendations in the following days. We are interested in whether SLED can successfully improve the recommendation quality by denoising in such real-world industrial scenarios.

In this light, we train the base model (NGCF) and the SLED-enhanced model on the interaction data of the first day, and test their performance on the interaction data of the remaining four days respectively. We use five commonly used recommendation measurements which may help make a comprehensive analysis. Each metric is computed on the top 20 items recommended by models, *i.e.*, @20. The testing results are listed in Table 11. We can find that:

- Overall, the consistent performance improvement over the base model indicates that SLED can denoise implicit feedback and be helpful for real-world industrial scenarios.

Table 11. Results on a noisy industrial dataset collected during the period of a product promotion festival. By denoising implicit feedback data in Day1, SLED captures users' inherent interests that can generalize across days. Performance variance across days is an indicator of model's stability against noisy interactions. Rel. Diff. denotes the relative difference between the metric values of SLED and the base models. Larger values are better for all metrics except for Variance.

Model	Testing Set	Day2	Day3	Day4	Day5	Variance
Base model	Precision	0.0071	0.0061	0.0061	0.0038	1.9
	Recall	0.0161	0.0146	0.0151	0.0124	2.4
	NDCG	0.0303	0.0254	0.0286	0.0215	14.9
	HitRate	0.0901	0.0754	0.0812	0.0626	132.9
	AUC	0.5743	0.6085	0.6103	0.5999	274.9
+ SLED	Precision	0.0089	0.0088	0.0080	0.0064	1.3
	Recall	0.0220	0.0211	0.0202	0.0191	1.6
	NDCG	0.0385	0.0415	0.0384	0.0326	13.9
	HitRate	0.1303	0.1304	0.1192	0.1040	155.2
	AUC	0.6515	0.6819	0.6767	0.6824	214.6
%Rel. Diff.		+29.4%	+47.3%	+31.4%	+50.8%	-15.4%

- Interestingly, the relative improvements over the base method in the distant future (*i.e.*, 47.3%/31.4%/50.8% improvement in day3/4/5) are overall larger than the improvements in the near future (*i.e.*, 29.4% improvement in day2). Intuitively, the promotion strategies are more likely to be similar on adjacent days (*e.g.*, day1 and day2) than non-adjacent days (*e.g.*, day1 and day5). These results provide evidence that the base model without denoising overfits to noisy interactions, hurting the generalization to remote days. In contrast, by denoising, SLED captures inherent user interests that can generalize across days and yields less performance drop than the base model in the distant future.
- We take the performance variance across days as an indicator of model stability [42] or robustness against noisy interactions. In summary, SLED reduces the performance variance by 15.4%, which further demonstrates that SLED achieves successful denoising and improves the generalization capability of models.

5 CONCLUSION AND FUTURE WORK

In this paper, we study how to denoise implicit feedback in recommender systems without external signals. Analogous to findings in the social network, we reveal that the structural patterns are correlated with some explicit feedback (*i.e.*, users' ratings on items). To capture such structural patterns, we propose the center-aware structure learning on multiple large-scale recommendation datasets. For a given recommendation dataset, we determine the reliability of interactions based on their related structural patterns. We leverage the predicted reliability weights to denoise the user-item representation learning and the ranking objective function. We conduct experiments on three real-world datasets and a noisy industrial dataset, validating the effectiveness of SLED in denoising implicit feedback and improving the recommendation quality.

We believe the insights of SLED will inspire further research on improving the stability and robustness of recommendation models. Capturing users' inherent interests that could generalize across time and domains is essential for recommender systems. There are more to explore in this direction. We plan to investigate the intersection of generic stable prediction techniques [42, 43] and the proposed SLED framework in the future. Another future work is to simplify the modeling of neighborhood graphs. For example, we will study whether fast solutions exist for neighborhood graph sampling and center-aware structure encoding for edges. Lastly, although we mainly evaluate the effectiveness of SLED in the matching stage of recommender systems, SLED is

potentially applicable for the deep ranking stage with complex model architectures and various content features. We will further explore it in the future.

6 ACKNOWLEDGMENTS

This work was supported in part by the Key R & D Projects of the Ministry of Science and Technology (2020YFC0832500), National Natural Science Foundation of China (62006207, U20A20387, 62037001),Young Elite Scientists Sponsorship Program by CAST (2021QNRC001), Zhejiang Province Natural Science Foundation (LR19F020006, LQ21F020020), Project by Shanghai AI Laboratory (P22KS00111), Program of Zhejiang Province Science and Technology (2022C01044), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010).

REFERENCES

[1] Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the Web. *Soc. Networks* 25, 3 (2003), 211–230. https://doi.org/10.1016/S0378-8733(03)00009-1

[2] Gediminas Adomavicius, Jesse Bockstedt, Shawn Curley, and Jingjing Zhang. 03/2021. Effects of Personalized and Aggregate Top-N Recommendation Lists on User Preference Ratings. *ACM Transactions on Information Systems* 39, 2 (03/2021), 1–38. https://doi.org/10.1145/3430028

[3] Sambaran Bandyopadhyay, Anirban Biswas, M. Narasimha Murty, and Ramasuri Narayanam. 2019. Beyond Node Embedding: A Direct Unsupervised Edge Representation Framework for Homogeneous Networks. *CoRR* abs/1912.05140 (2019). http://arxiv.org/abs/1912.05140

[4] Zhi Bian, Shaojun Zhou, Hao Fu, Qihong Yang, Zhenqi Sun, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. Denoising User-aware Memory Network for Recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 400–410. https://doi.org/10.1145/3460231.3474237

[5] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951. https://doi.org/10.1145/3394486.3403344

[6] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems* 38, 2 (2020), 1–28. https://doi.org/10.1145/3373807

[7] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. 2022. Denoising Self-Attentive Sequential Recommendation. In *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*. ACM, 92–101.

[8] Jiawei Chen, Chengquan Jiang, Can Wang, Sheng Zhou, Yan Feng, Chun Chen, Martin Ester, and Xiangnan He. 2021. CoSam: An Efficient Collaborative Adaptive Sampler for Recommendation. *ACM Transactions on Information Systems* 39, 3 (2021), 1–24. https://doi.org/10.1145/3450289

[9] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten De Rijke. 2019. Joint Neural Collaborative Filtering for Recommender Systems. , 30 pages. https://doi.org/10.1145/3343117

[10] Yifan Chen, Yang Wang, Xiang Zhao, Hongzhi Yin, Ilya Markov, and MAARTEN De Rijke. 2020. Local Variational Feature-Based Similarity Models for Recommending Top- N New Items. *ACM Transactions on Information Systems* 38, 2 (2020), 1–33. https://doi.org/10.1145/3372154

[11] Yifan Chen, Yang Wang, Xiang Zhao, Jie Zou, and Maarten de Rijke. 2020. Block-Aware Item Similarity Models for Top-*N* Recommendation. *ACM Transactions on Information Systems* 38, 4 (2020), 42:1–42:26. https://doi.org/10.1145/3411754

[12] Zhengyu Chen, Teng Xiao, and Kun Kuang. 2022. BA-GNN: On Learning Bias-Aware Graph Neural Network. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3012–3024.

[13] Quanyu Dai, Yalei Lv, Jieming Zhu, Junjie Ye, Zhenhua Dong, Rui Zhang, Shu-Tao Xia, and Ruiming Tang. 2022. LCD: Adaptive Label Correction for Denoising Music Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*. ACM, 3903–3907.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[15] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2021. Sampler Design for Bayesian Personalized Ranking by Leveraging View Data. *IEEE Trans. Knowl. Data Eng.* 33, 2 (2021), 667–681.

[16] Jingtao Ding, Guanghui Yu, Yong Li, Xiangnan He, and Depeng Jin. 2020. Improving Implicit Recommender Systems with Auxiliary Data. *ACM Transactions on Information Systems* 38, 1 (2020), 1–27. https://doi.org/10.1145/3372338

[17] Xin-yu Duan, Si-liang Tang, Sheng-yu Zhang, Yin Zhang, Zhou Zhao, Jian-ru Xue, Yue-ting Zhuang, and Fei Wu. 2018. Temporality-enhanced knowledgememory network for factoid question answering. *Frontiers of Information Technology & Electronic Engineering* 19, 1 (2018), 104–115.

[18] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.* 11 (2010), 625–660.

[19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference on - WWW '19*. 417–426. https://doi.org/10.1145/3308558.3313488

[20] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2021. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Transactions on Information Systems* 39, 1 (2021), 1–42. https://doi.org/10.1145/3426723

[21] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 8/2019. Deep Session Interest Network for Click-Through Rate Prediction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2301–2307. https://doi.org/10.24963/ijcai.2019/319

[22] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 04/2005. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems* 23, 2 (04/2005), 147–168. https://doi.org/10.1145/1059981.1059982

[23] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan T. Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.* 23, 2 (2005), 147–168.

[24] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Personalized Ranking for Non-Uniformly Sampled Items. In *Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011 (JMLR Proceedings, Vol. 18)*. JMLR.org, 231–247.

[25] Yunjun Gao, Yuntao Du, Yujia Hu, Lu Chen, Xinjun Zhu, Ziquan Fang, and Baihua Zheng. 2022. Self-Guided Learning to Denoise for Robust Recommendation. *CoRR* abs/2204.06832 (2022). https://doi.org/10.48550/arXiv.2204.06832

[26] Yingqiang Ge, Mostafa Rahmani, Athirai A. Irissappane, Jose Sepulveda, Fei Wang, James Caverlee, and Yongfeng Zhang. 2023. Automated Data Denoising for Recommendation. *CoRR* abs/2305.07070 (2023).

[27] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

[28] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864. https://doi.org/10.1145/2939672.2939754

[29] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 1–19. https://doi.org/10.1145/2827872

[30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 6/2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975

[31] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web*. 507–517. https://doi.org/10.1145/2872427.2883037

[32] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648. https://doi.org/10.1145/3397271.3401063

[33] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. ACM, 173–182. https://doi.org/10.1145/3038912.3052569

[34] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[35] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020*.

[36] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1–10. https://doi.org/10.3115/v1/P15-1001

[37] Wang-Cheng Kang and Julian McAuley. 11/2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. https://doi.org/10.1109/ICDM.2018.00035

[38] Youngho Kim, Ahmed Hassan, Ryen W. White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 193–202. https://doi.org/10.1145/2556195.2556220

[39] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.
[40] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net.
[41] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757. https://doi.org/10.1145/3394486.3403226
[42] Kun Kuang, Peng Cui, Susan Athey, Ruoxuan Xiong, and Bo Li. 2018. Stable Prediction across Unknown Environments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1617–1626. https://doi.org/10.1145/3219819.3220082
[43] Kun Kuang, Ruoxuan Xiong, Peng Cui, Susan Athey, and Bo Li. 2020. Stable Prediction with Model Misspecification and Agnostic Distribution Shift. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (2020), 4485–4492. https://doi.org/10.1609/aaai.v34i04.5876
[44] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-Attention Graph Pooling. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019 (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 3734–3743.
[45] Bin Li, Ling Chen, Xingquan Zhu, and Chengqi Zhang. 2013. Noisy but non-malicious user detection in social recommender systems. *World Wide Web* 16, 5-6 (2013), 677–699.
[46] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623. https://doi.org/10.1145/3357384.3357814
[47] Weilin Lin, Xiangyu Zhao, Yejing Wang, Yuanshao Zhu, and Wanyu Wang. 2023. AutoDenoise: Automatic Data Instance Denoising for Recommendations. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. ACM, 1003–1011.
[48] Yujie Lin, Chenyang Wang, Zhumin Chen, Zhaochun Ren, Xin Xin, Qiang Yan, Maarten de Rijke, Xiuzhen Cheng, and Pengjie Ren. 2023. A Self-Correcting Sequential Recommender. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. ACM, 1283–1293.
[49] Chao Liu, Ryen W. White, and Susan Dumais. 2010. Understanding web browsing behaviors through Weibull analysis of dwell time. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*. 379. https://doi.org/10.1145/1835449.1835513
[50] Yiyu Liu, Qian Liu, Yu Tian, Changping Wang, Yanan Niu, Yang Song, and Chenliang Li. 2021. Concept-Aware Denoising Graph Neural Network for Micro-Video Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1099–1108. https://doi.org/10.1145/3459637.3482417
[51] Hongyu Lu, Min Zhang, and Shaoping Ma. 2018. Between Clicks and Satisfaction: Study on Multi-Phase User Preferences and Satisfaction for Online News Reading. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 435–444. https://doi.org/10.1145/3209978.3210007
[52] Hongyu Lu, Min Zhang, and Shaoping Ma. 2018. Between Clicks and Satisfaction: Study on Multi-Phase User Preferences and Satisfaction for Online News Reading. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. ACM, 435–444.
[53] Hongyu Lu, Min Zhang, Weizhi Ma, Ce Wang, Feng xia, Yiqun Liu, Leyu Lin, and Shaoping Ma. 2019. Effects of User Negative Experience in Mobile News Streaming. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 705–714. https://doi.org/10.1145/3331184.3331247
[54] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 4212–4221.
[55] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52. https://doi.org/10.1145/2766462.2767755
[56] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 08/2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* 27, 1 (08/2001), 415–444. https://doi.org/10.1146/annurev.soc.27.1.415
[57] Yunhe Pan. 2020. Multiple knowledge representation of artificial intelligence. *Engineering* 6, 3 (2020), 216–217.
[58] Zhen Peng, Yixiang Dong, Minnan Luo, Xiao-Ming Wu, and Qinghua Zheng. 2020. Self-Supervised Graph Representation Learning via Global Context Prediction. *CoRR* abs/2003.01604 (2020). https://arxiv.org/abs/2003.01604
[59] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710. https://doi.org/10.1145/2623330.2623732
[60] Yuqi Qin, Pengfei Wang, and Chenliang Li. 2021. The World is Binary: Contrastive Learning for Denoising Next Basket Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 859–868. https:

//doi.org/10.1145/3404835.3462836
[61] Yuqi Qin, Pengfei Wang, and Chenliang Li. 2021. The World is Binary: Contrastive Learning for Denoising Next Basket Recommendation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. ACM, 859–868.
[62] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160. https://doi.org/10.1145/3394486.3403168
[63] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. ACM, 2110–2119. https://doi.org/10.1145/3219819.3220077
[64] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting Cross-session Information for Session-based Recommendation with Graph Neural Networks. *ACM Transactions on Information Systems* 38, 3 (2020), 1–23. https://doi.org/10.1145/3382764
[65] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[66] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*. ACM, 1441–1450. https://doi.org/10.1145/3357384.3357895
[67] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*. 1067–1077. https://doi.org/10.1145/2736277.2741093
[68] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573. https://doi.org/10.1145/3159652.3159656
[69] Zhiqiang Tian, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, and Mingyue Zhu. 2022. Exploiting Group Information for Personalized Recommendation with Graph Neural Networks. *ACM Transactions on Information Systems* 40, 2 (2022), 1–23. https://doi.org/10.1145/3464764
[70] Xiaohai Tong, Pengfei Wang, Chenliang Li, Long Xia, and Shaozhang Niu. 2021. Pattern-enhanced Contrastive Policy Learning Network for Sequential Recommendation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. ijcai.org, 1593–1599.
[71] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. 2012. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* 109, 16 (2012), 5962–5966. https://doi.org/10.1073/pnas.1116502109
[72] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* abs/1706.02263 (2017). http://arxiv.org/abs/1706.02263
[73] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). http://arxiv.org/abs/1807.03748
[74] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018*.
[75] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising Implicit Feedback for Recommendation. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*. ACM, 373–381.
[76] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Learning Robust Recommender from Noisy Implicit Feedback. *CoRR* abs/2112.01160 (2021). https://arxiv.org/abs/2112.01160
[77] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising Implicit Feedback for Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 373–381. https://doi.org/10.1145/3437963.3441800
[78] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. 950–958. https://doi.org/10.1145/3292500.3330989
[79] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174. https://doi.org/10.1145/3331184.3331267
[80] Yu Wang, Xin Xin, Zaiqiao Meng, Joemon M. Jose, Fuli Feng, and Xiangnan He. 2022. Learning Robust Recommenders through Cross-Model Agreement. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2015–2025.
[81] Zitai Wang, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Implicit Feedbacks are Not Always Favorable: Iterative Relabeled One-Class Collaborative Filtering against Noisy Interactions. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*. ACM, 3070–3078.
[82] Hongyi Wen, Longqi Yang, and Deborah Estrin. 2019. Leveraging post-click feedback for content recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 278–286. https://doi.org/10.1145/3298689.3347037

[83] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A Large-scale Dataset for News Recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*. Association for Computational Linguistics, 3597–3606. https://doi.org/10.18653/v1/2020.acl-main.331

[84] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao (Eds.). 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Nature Singapore. https://doi.org/10.1007/978-981-16-6054-2

[85] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. AAAI Press, 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[86] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.

[87] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. 2022. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *Proceedings of the ACM Web Conference 2022*. 1070–1079. https://doi.org/10.1145/3485447.3512156

[88] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019*.

[89] Weinan Xu, Hengxu He, Minshi Tan, Yunming Li, Jun Lang, and Dongbai Guo. 2020. Deep Interest with Hierarchical Attention Network for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1905–1908. https://doi.org/10.1145/3397271.3401310

[90] Byoungju Yang, Sangkeun Lee, Sungchan Park, and Sang-goo Lee. 2012. Exploiting various implicit feedback for collaborative filtering. In *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*. 639. https://doi.org/10.1145/2187980.2188166

[91] Jun Yang, Weizhi Ma, Min Zhang, Xin Zhou, Yiqun Liu, and Shaoping Ma. 2022. LegalGNN: Legal Information Enhanced Graph Neural Network for Recommendation. *ACM Transactions on Information Systems* 40, 2 (2022), 1–29. https://doi.org/10.1145/3469887

[92] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14*. 113–120. https://doi.org/10.1145/2645710.2645724

[93] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. ACM, 974–983. https://doi.org/10.1145/3219819.3219890

[94] Wenhui Yu and Zheng Qin. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 861–870.

[95] Dong Zhang, Shu Zhao, Zhen Duan, Jie Chen, Yanping Zhang, and Jie Tang. 2020. A Multi-Label Classification Method Using a Hierarchical and Transparent Representation for Paper-Reviewer Recommendation. *ACM Transactions on Information Systems* 38, 1 (2020), 1–20. https://doi.org/10.1145/3361719

[96] Yuan Zhang, Fei Sun, Xiaoyong Yang, Chen Xu, Wenwu Ou, and Yan Zhang. 2021. Graph-based Regularization on Embedding Layers for Recommendation. *ACM Transactions on Information Systems* 39, 1 (2021), 1–27. https://doi.org/10.1145/3414067

[97] Qian Zhao, Shuo Chang, F. Maxwell Harper, and Joseph A. Konstan. 2016. Gaze Prediction for Recommender Systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 131–138. https://doi.org/10.1145/2959100.2959150

[98] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3985–3995. https://doi.org/10.1145/3447548.3467102

[99] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. 5941–5948. https://doi.org/10.1609/aaai.v33i01.33015941

[100] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068. https://doi.org/10.1145/3219819.3219823

[101] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2388–2399.

[102] Dingyuan Zhu, Daixin Wang, Zhiqiang Zhang, Kun Kuang, Yan Zhang, Yulin Kang, and Jun Zhou. 2023. Graph Neural Network with Two Uplift Estimators for Label-Scarcity Individual Uplift Modeling. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. ACM, 395–405.

[103] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1079–1088. https://doi.org/10.1145/3219819.3219826

[104] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080. https://doi.org/10.1145/3442381.3449802

[105] Yueting Zhuang, Ming Cai, Xuelong Li, Xiangang Luo, Qiang Yang, and Fei Wu. 2020. The next breakthroughs of artificial intelligence: The interdisciplinary nature of AI. *Engineering* 6, 3 (2020), 245–247.

[106] Yue-ting Zhuang, Fei Wu, Chun Chen, and Yun-he Pan. 2017. Challenges and opportunities: from big data to knowledge in AI 2.0. *Frontiers of Information Technology & Electronic Engineering* 18 (2017), 3–14.