Graph-less Collaborative Filtering

Lianghao Xia
The University of Hong Kong
aka_xia@foxmail.com

Chao Huang*
The University of Hong Kong
chaohuang75@gmail.com

Jiao Shi
South China University of Technology
yjjiaoshi@scut.edu.cn

Yong Xu
South China University of Technology
yxu@scut.edu.cn

ABSTRACT

Graph neural networks (GNNs) have shown the power in representation learning over graph-structured user-item interaction data for collaborative filtering (CF) task. However, with their inherently recursive message propagation among neighboring nodes, existing GNN-based CF models may generate indistinguishable and inaccurate user (item) representations due to the over-smoothing and noise effect with low-pass Laplacian smoothing operators. In addition, the recursive information propagation with the stacked aggregators in the entire graph structures may result in poor scalability in practical applications. Motivated by these limitations, we propose a simple and effective collaborative filtering model (SimRec) that marries the power of knowledge distillation and contrastive learning. In SimRec, adaptive transferring knowledge is enabled between the teacher GNN model and a lightweight student network, to not only preserve the global collaborative signals, but also address the over-smoothing issue with representation recalibration. Empirical results on public datasets show that SimRec archives better efficiency while maintaining superior recommendation performance compared with various strong baselines. Our implementations are publicly available at: https://github.com/HKUDS/SimRec.

CCS CONCEPTS

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

KEYWORDS

Collaborative Filtering, Graph Neural Network, Contrastive Learning, Knowledge Distillation, Recommender Systems

ACM Reference Format:
Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. 2023. Graph-less Collaborative Filtering. In *Proceedings of the ACM Web Conference 2023 (WWW'23), April 30-May 4, 2023, Austin, TX, USA.* ACM, Austin, TX, USA, 11 pages. https://doi.org/10.1145/3543507.3583196

*Chao Huang is the corresponding author.

WWW'23, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583196

1 INTRODUCTION

Recent years have witnessed the great success of graph neural network (GNN) in learning latent representations for graph structured data [27, 31, 48]. Inspired by such development, many efforts have introduced GNN into Collaborative Filtering (CF) and shown its power in modeling high-order user-item relationships, such as NGCF [28], LightGCN [12], and GCCF [5]. At the core of GNN-based CF models is to utilize a neighborhood aggregation scheme to encode user (item) embeddings via recursively message passing.

Despite their achieved remarkable performance, we argue that two important limitations exist in current GNN-based CF methods.

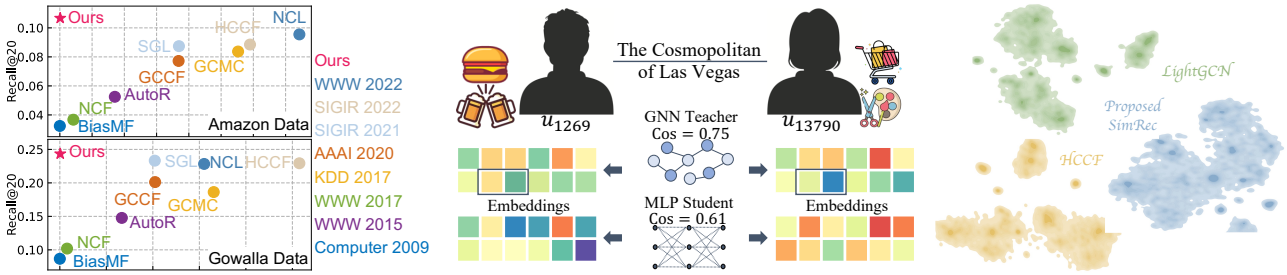
(i) **Over-Smoothing and Noise Issues**. The inherent design of GNN may lead to *over-smoothing* issue as the increase of stacked graph layers for embedding propagation [21, 47]. The neighborhood aggregator built upon low-pass Laplacian smoothing operation with graph-structured user-item connections, will unavoidably generate indistinguishable user and item representations as the number of layers increases. This results in suboptimal performance as the GNN-based recommenders may fail to capture diverse user preferences. Additionally, various biases of user behavior data widely exist in recommender systems [4, 45], such as misclick behaviors, popularity bias. The recursive aggregation schema used in GNN-based models is prone to fusing *noisy* signals, which can hinder the learning of genuine interaction patterns for recommendation.

(ii) **Scalability Limitation with Recursive Expansion**. Although GNN-based recommenders can capture high-order connectivity by stacking multiple propagation layers, the recursive neighbor information aggregation incurs expensive computation in model inference [9, 38, 46]. Therefore, GNN-based CF models with deeper graph neural layers require repeatedly propagate representations among many neighboring nodes, which show poor scalability in practical scenarios, especially for large-scale recommender systems. In light of this, the scalability limitation of current GNN-based methods brings an urge for designing an efficient and effective high-order relation learning paradigm in recommendation, which remains unexplored in existing CF recommendation models.

Having realized the importance of addressing the above challenges, however, it is non-trivial considering the following factors:

- How to well preserve global collaborative signals in an efficient manner for user-item interaction modeling, remains a challenge.
- How to encode informative representations which are robust to over-smoothing and noise issues while preserving high-order interaction patterns, requiring adaptive knowledge transfer.

As shown in Figure 1, we illustrate motivation examples for the model design in our SimRec recommender. To be specific, by



(a) Performance v.s. inference time (b) Advantage of our adaptive embedding recalibration (c) Distribution of learned user embeddings

Figure 1: Illustration of motivation and advantages of our SimRec model from different perspectives.

comparing our SimRec with various state-of-the-art GNN-based CF methods (*e.g.*, GCCF [5], SGL [32], HCCF [34]) in Figure 1(a), SimRec significantly improves model efficiency, meanwhile maintaining superior recommendation performance. In Figure 1(b) of two users with dissimilar interests, we show the advantage of our adaptive contrastive knowledge distillation to recalibrate the similar representations encoded by GNN teacher model into distinguishable embedding space. To reflect the better uniformity preserved in learned embeddings of SimRec, we visualize the distributions of projected embeddings learned by different methods in Figure 1(c).

Inspired by the effectiveness of knowledge distillation (KD) in various domains (*e.g.*, computer vision [44], text mining [7], and graph mining [43]), KD has become an effective solution to transfer knowledge from a large model to a smaller one. In general, KD aims to reach the agreement between the prediction results of a well-trained teacher model and a student model by minimizing their distribution difference. However, collaborative filtering task usually involves highly sparse interaction data, which undermines the capability of knowledge distillation. Specifically, direct distillation from noisy and sparse graph structures, is difficult to advance the performance of original GNN model after being compressed. Fortunately, recent developments of contrastive learning bring new insights in alleviating data sparsity with auxiliary self-supervision signals, this paper explores the possibility of marrying the power of knowledge distillation and contrastive learning to pursue adaptive knowledge transfer with a robust and efficient CF model.

In this work, we propose a novel graph-less collaborative filtering framework, named SimRec, to improve both the effectiveness and efficiency of recommender without the sophisticated GNN structures. In particular, we propose a bi-level alignment framework to distill knowledge with both prediction-level and embedding-level signals. With such design, the distilled knowledge comes from not only the teacher model's predictions but also the latent high-order collaborative semantics preserved in embeddings. Furthermore, we propose to enhance our knowledge distillation paradigm against the perturbation of over-smoothing and noise effects in GNN teacher model. Towards this end, an adaptive knowledge transfer module is designed with contrastive regularization to capture the diversity of user preference, based on the derived consistency between the supervised CF objective and the augmented SSL task. In our proposed SimRec model, the latent knowledge of GNN-based teacher model will be distilled into a lightweight yet empowered feed-forward network that can jointly capture user-specific preference uniformity and cross-user global collaborative dependencies.

To summarize, our contributions are presented as follows:

- We propose contrastive knowledge distillation to compress GNN-based CF model into a simple recommender to improve both effectiveness and efficiency. In our adaptive distillation paradigm, an embedding calibration module is designed to enhance KD to preserve useful knowledge and discard the noisy information.
- Theoretical analysis is provided from two perspectives: i) the benefits of our distillation model in alleviating over-smoothing issue; ii) effectiveness of our distilled self-supervision signals for data augmentation in an adaptive manner.
- Extensive experiments on public datasets demonstrate that SimRec significantly improves the performance of CF tasks. Additionally, the empirical results show that SimRec gains more efficient embedding encoding over LightGCN on different datasets.

2 COLLABORATIVE FILTERING

In this section, we introduce important notations in collaborative filtering, and recap MLP-based Neural CF and GNN-based CF architectures. In a typical recommendation scenario, there are I users $\{u_1, u_2, ..., u_I\}$ and J items $\{v_1, v_2, ..., v_J\}$, indexed by u_i and v_j, respectively. An interaction matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ represents the observed interactions between users and items, in which an element $a_{i,j} = 1$ if user u_i has adopted item v_j, otherwise $a_{i,j} = 0$.

Based on the above definitions, a CF-based recommender can be formalized as an inference model that i) **Inputs** the user-item interaction data $\mathbf{A} \in \mathbb{R}^{I \times J}$, models users' interactive patterns based on the input; ii) **Outputs** interaction prediction results $y_{i,j}$ between the unobserved user-item pair (u_i, v_j). In general, a CF model can be summarized as the following two-stage schema:

$$y_{i,j} \leftarrow \textbf{Predict}(\mathbf{h}_i, \mathbf{h}_j), \quad \mathbf{h}_i, \mathbf{h}_j \leftarrow \textbf{Embed}(u_i, v_j; \mathbf{A}) \tag{1}$$

The first stage **Embed**$(\cdot)$ denotes the embedding process which projects user u_i and item v_j into a d-dimensional hidden space based on the observed historical interactions $\mathbf{A}$. The results of **Embed**$(\cdot)$ is vectorized representations $\mathbf{h}_i, \mathbf{h}_j \in \mathbb{R}^d$ for each user u_i and item v_j, to preserve user-item interactive patterns. The second stage **Predict**$(\cdot)$ aims to forecast user-item relations with the prediction score $y_{i,j} \in \mathbb{R}$ using the learned embeddings $\mathbf{h}_i, \mathbf{h}_j$. Based on the above two-stage schema, our proposed method SimRec aims to conduct knowledge distillation from both the embedding and prediction levels for effectively knowledge transferring.

MLP-based Collaborative Filtering. MLP-based neural CF methods [13, 37] are proposed to endow CF with non-linear relation modeling. Due to the simplicity in model architectures, MLP-based CF is highly-efficient and unlikely to learn over-smoothed emebddings like GNNs [3]. Inspired by the advantages, we adopt MLP as

the student model in our contrastive KD framework. In brief, the MLP in SimRec adheres to the two-stage paradigm as follows:

$$y_{i,j} = \mathbf{h}_i^\top \mathbf{h}_j, \quad \mathbf{h}_i = \textbf{M-Embed}(\bar{\mathbf{h}}_i), \quad \mathbf{h}_j = \textbf{M-Embed}(\bar{\mathbf{h}}_j) \tag{2}$$

where $\bar{\mathbf{h}}_i, \bar{\mathbf{h}}_j \in \mathbb{R}^d$ denote the initial embedding vectors for user u_i and item v_j, respectively. **M-Embed**$(\cdot)$ denotes the MLP-based embedding function. We adopt dot-product for **Predict**$(\cdot)$, which has been shown to be efficient and effective [23].

GNN-enhanced Collaborative Filtering. Most recent CF models apply graph neural information propagation on a bipartite interaction graph $\mathcal{G} = \{\mathcal{U}, \mathcal{V}, \mathcal{E}\}$, to encode users' high-order interactive relations into node embeddings. Here, $\mathcal{U} = \{u_i\}, \mathcal{V} = \{v_j\}$ denote user and item node sets, respectively. An edge $e_{i,j} \in \mathcal{E}$ exists if and only if $a_{i,j} = 1$. Typically, GNN-based CF can be abstracted as:

$$y_{i,j} = \mathbf{h}_i^\top \mathbf{h}_j, \ \mathbf{H} = \textbf{G-Embed}(\mathcal{G}, \bar{\mathbf{H}}) = \big(\textbf{Agg}(\textbf{Prop}(\mathcal{G}, \bar{\mathbf{H}}))\big)^L \tag{3}$$

where $\mathbf{H}, \bar{\mathbf{H}} \in \mathbb{R}^{(I+J)\times d}$ denote the embedding matrices whose rows are node embedding vectors. **G-Embed**$(\cdot)$ denotes the GNN-based embedding function which iteratively propagates (**Prop**$(\cdot)$) and aggregates (**Agg**$(\cdot)$) the embeddings $\bar{\mathbf{H}}$ along the interaction graph $\mathcal{G}$ for L times. Note that though it injects informative structural information, GNNs based on holistic graph modeling and high-order iterations also damage the model scalability and bring the risk of over-smoothing in the collaborative filtering task.

3 METHODOLOGY

In this section, we elaborate the technical details of our proposed SimRec framework, whose workflow is depicted in Figure 2.

3.1 Contrastive Knowledge Distillation

For the model design, we are motivated by the advantages of i) GNNs in learning structure-aware node embeddings, and ii) efficient MLPs in preventing over-smoothing issue. Towards this end, we propose to distill knowledge from a GNN-based teacher model to a MLP-based student model. Specifically, the teacher model is a lightweight Graph Convolutional Network (GCN) [2, 12, 32] whose embedding process is shown with the following propagation:

$$\mathbf{H}^{(t)} = \sum_{l=0}^{L} \mathbf{H}_l^{(t)}, \quad \mathbf{H}_{l+1}^{(t)} = \mathbf{D}^{-\frac{1}{2}}(\bar{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}} \cdot \mathbf{H}_l^{(t)} \tag{4}$$

where $\mathbf{H}^{(t)} \in \mathbb{R}^{(I+J)\times d}$ denotes the embedding matrix given by the teacher model. Index l indicates the number of graph neural iterations (totally L iterations). $\bar{\mathbf{A}} \in \mathbb{R}^{(I+J)\times(I+J)}$ denotes the symmetric adjacent matrix for graph $\mathcal{G}$ generated from the interaction matrix $\mathbf{A}$ [28]. $\mathbf{I}$ denotes the identity matrix, and $\mathbf{D}$ denotes the diagonal degree matrix of $\bar{\mathbf{A}}$. The iteration is initialized by $\mathbf{H}_0^{(t)} = \bar{\mathbf{H}}^{(t)}$. The student model uses a shared MLP network to extract features from the initial embeddings for both users and items. For user u_i, the embedding layer is formally presented as follows:

$$\mathbf{h}_i^{(s)} = \mathbf{FC}^{L'}(\bar{\mathbf{h}}_i^{(s)}), \quad \mathbf{FC}(\bar{\mathbf{h}}_i^{(s)}) = \delta(\mathbf{W}\bar{\mathbf{h}}_i^{(s)}) + \bar{\mathbf{h}}_i^{(s)} \tag{5}$$

where $\mathbf{h}_i^{(s)}, \bar{\mathbf{h}}_i^{(s)} \in \mathbb{R}^d$ denote embeddings for u_i given by the student. $\mathbf{FC}(\cdot)$ denotes the fully-connected layer. L' is the number of FC layers. An FC layer is configured with one transformation $\mathbf{W} \in \mathbb{R}^{d\times d}$. LeakyReLU activation $\delta(\cdot)$, and a residual connection [11] are applied. Item-side embedding layer is built analogously.

3.1.1 **Prediction-Level Distillation.** To distill knowledge from the teacher model to the student model, SimRec first follows the paradigm of KL-divergence-based KD [14] to align the predictive outputs between the teacher and student models. Inspired by the success of ranking-oriented BPR loss [22] in recommender systems, SimRec aligns the two models on the task of ranking user preference. Specifically, in each training step, we randomly sample a batch of triplets $\mathcal{T}_1 = \{(u_i, v_j, v_k)\}$, where u_i, v_j, v_k are individually sampled from the holistic user and item set with uniform probability. Then SimRec calculates the preference difference between (u_i, v_j) and (u_i, v_k) for both models, as follows:

$$z_{i,j,k} = y_{i,j} - y_{i,k} = \mathbf{h}_i^\top \mathbf{h}_j - \mathbf{h}_i^\top \mathbf{h}_k \tag{6}$$

where $z_{i,j,k} \in \mathbb{R}$ denotes the difference scores of user preferences for triplet (u_i, v_j, v_k). We denote the score given by the student model as $z_{i,j,k}^{(s)}$ and denote the score given by the teacher model as $z_{i,j,k}^{(t)}$. Then, the prediction-oriented distillation is conducted by minimizing the following loss function:

$$\mathcal{L}_1 = \sum_{(u_i,v_j,v_k)\in\mathcal{T}_1} -\Big(\bar{z}_{i,j,k}^{(t)} \cdot \log \bar{z}_{i,j,k}^{(s)} + (1 - \bar{z}_{i,j,k}^{(t)}) \cdot \log(1 - \bar{z}_{i,j,k}^{(s)})\Big)$$
$$\bar{z}_{i,j,k}^{(t)} = \text{sigm}(z_{i,j,k}^{(t)}/\tau_1), \quad \bar{z}_{i,j,k}^{(s)} = \text{sigm}(z_{i,j,k}^{(s)}/\tau_1) \tag{7}$$

where $\bar{z}_{i,j,k}^{(t)}, \bar{z}_{i,j,k}^{(s)}$ are preference differences processed by sigmoid function $\text{sigm}(\cdot)$ with temperature factor τ_1. Here, $z_{i,j,k}^{(t)}$ is given by well-trained teacher model and does not back-propagate gradients. With the help of prediction-oriented distillation $\mathcal{L}_1$, simple MLPs learn to mimic the predictions of advanced GNN models and thus directly generate recommendation results. Through this end-to-end supervision, the parameters of student model are optimized to preserve the knowledge distilled from the teacher model.

It is worth noting that, our prediction-level KD differs from vanilla KD in its training sample enrichment for deep *dark knowledge* learning [8, 24] in CF. Specifically, vanilla KD for multi-class classification [14] mines dark knowledge from not only the class with highest score, but also from the ranks for all classes. However, treating CF as multi-classification is problematic, as there are too many classes (items), such that the soft labels easily approach zero and become hard to rank. To solve it, our prediction-level KD adopts the pair-wise ranking task instead, and excavates the dark knowledge by distilling from *enriched samples*. Unlike BPR-based model training which pairs each positive item with one negative item, our KD scheme learns from the teacher's predictions on v_j, v_k individually sampled from the holistic item set. Here v_j, v_k are not fixed to be positive or negative. This greatly enriches the training set for our KD and facilitate deeper dark knowledge distillation.

3.1.2 **Embedding-Level Distillation.** Despite the efficacy, the above prediction-level distillation only supervises the model outputs, but ignores the potential difference of embedding distributions between the student and teacher. As both models follow the embedding and prediction schema in Eq 1, we extend the KD paradigm in SimRec with an embedding-level knowledge transferring based

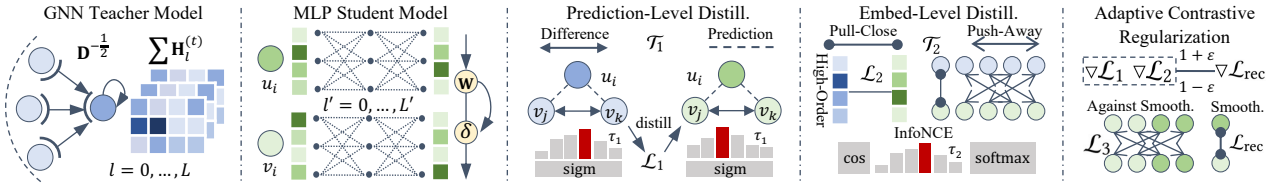


Figure 2: Model architecture of the proposed SimRec framework.

on contrastive learning. Specifically, we sample a batch of users and items $\mathcal{T}_2 = \{u_i, v_j\}$ from the observed interactions in each training step. Then, we apply the following contrastive loss on the corresponding user/item embeddings:

$$\mathcal{L}_2 = \sum_{u_i \in \mathcal{T}_2} -\log \frac{\exp\left(\cos(\mathbf{h}_i^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{i,l}^{(t)})/\tau_2\right)}{\sum_{u_{i'} \in \mathcal{U}} \exp\left(\cos(\mathbf{h}_{i'}^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{i,l}^{(t)})/\tau_2\right)} + \sum_{v_j \in \mathcal{T}_2} -\log \frac{\exp\left(\cos(\mathbf{h}_j^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{j,l}^{(t)})/\tau_2\right)}{\sum_{v_{j'} \in \mathcal{V}} \exp\left(\cos(\mathbf{h}_{j'}^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{j,l}^{(t)})/\tau_2\right)} \tag{8}$$

where $\cos(\cdot)$ denotes the cosine similarity function. τ_2 represents the temperature hyperparameter. To force the student model to learn more from the high-order patterns which MLP-based CF lacks, here we only use the high-order node embeddings from the teacher. Embeddings of the teacher are well-trained and fixed in parameter optimization. Through directly regularizing the hidden embeddings with this embedding-oriented distillation, SimRec not only further improves the performance of student model, but also greatly accelerates the cross-model distillation, which has been validated in our empirical evaluations.

3.2 Adaptive Contrastive Regularization

To prevent transferring over-smoothed signals from the GNN-based teacher to the student model, SimRec proposes to regularize the embedding learning of the student by universally minimizing the node-wise similarity. Specially, SimRec adaptively locates which nodes are more likely being over-smoothed by comparing the gradients of distillation tasks with the main task gradients. In particular, we reuse the sampled users and items $\mathcal{T}_2$ from the embedding-level distillation, and apply the following adaptive contrastive regularization for node embeddings of the student model:

$$\mathcal{L}_3 = \sum_{u_i \in \mathcal{T}_2} \varphi(u_i, \mathcal{U}, \omega_i) + \varphi(u_i, \mathcal{V}, \omega_i) + \sum_{v_j \in \mathcal{T}_2} \varphi(v_j, \mathcal{V}, \omega_j)$$
$$\varphi(u_i, \mathcal{U}, \omega_i) = \omega_i \cdot \log \sum_{u_{i'} \in \mathcal{U}} \exp(\mathbf{h}_i^{(s)\top} \mathbf{h}_{i'}^{(s)} / \tau_3) \tag{9}$$

where the loss $\mathcal{L}_3$ is composed of three terms ($\varphi(\cdot)$) that pushes away the user-user distance, the user-item distance, and the item-item distance, respectively. The first term $\varphi(u_i, \mathcal{U}, \omega_i)$ minimizes the dot-product similarity between the embedding of u_i and the embedding of each user $u_{i'}$ in $\mathcal{U}$, with a weighting factor ω_i. Here, the similarity score is adjusted with the temperature hyperparameter τ_3. The $\phi(\cdot)$ functions for user-item relations and item-item relations work analogously. The weighting factor ω_i, ω_j correspond to u_i, v_j respectively, and the weight is calculated as follows:

$$\omega_i = \begin{cases} 1-\varepsilon & \text{if } \nabla_i^{1,2\top} \nabla_i^{\text{rec}} > \nabla_i^{1\top} \nabla_i^2 \\ 1+\varepsilon & \text{otherwise} \end{cases} \qquad \nabla_i^{\text{rec}} = \frac{\partial \mathcal{L}_{\text{rec}}}{\partial \mathbf{h}_i^{(s)}}$$
$$\nabla_i^{1,2} = \frac{\partial(\mathcal{L}_1 + \mathcal{L}_2)}{\partial \mathbf{h}_i^{(s)}}, \qquad \nabla_i^{1} = \frac{\partial \mathcal{L}_1}{\partial \mathbf{h}_i^{(s)}}, \qquad \nabla_i^{2} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{h}_i^{(s)}} \tag{10}$$

where $\omega_i \in \mathbb{R}$ adjusts the weight of contrastive regularization for user u_i. In brief, ω_i has the larger value (*i.e.*, $1+\epsilon$) when the gradients given by distillation tasks (which may over-smooth) contradict to the gradients generated by the main task (which hardly over-smooth). Here, $0 < \epsilon < 1$ is a hyperparameter. $\nabla_i \in \mathbb{R}^d$ denotes the gradients for the embedding vector $\mathbf{h}_i$ *w.r.t*, different optimization tasks. For example, $\nabla_i^{1,2}$ denotes the compound gradients of the two distillation task objectives $\mathcal{L}_1$ and $\mathcal{L}_2$. ∇_i^{rec} denotes gradient of the recommendation task, which is independent to the GNN-based teacher and thus has no risk of over-smoothing. The task $\mathcal{L}_{\text{rec}}$ will be elaborated later. The similarity between the gradients is estimated using dot-product. When the similarity between the distillation tasks and the recommendation task, is larger than the similarity between two distillation tasks, we can assume that the difference in optimization between the distillation and the recommendation is small enough to weaken the regularization.

3.3 Parameter Learning of SimRec

Following the training paradigm of knowledge distillation, our SimRec first trains the GNN-based teacher model until convergence. In each step, SimRec samples a batch of triplets $\mathcal{T}_{\text{bpr}} = \{(u_i, v_j, v_k) | a_{i,j} = 1, a_{i,k} = 0\}$ where u_i denotes anchor user. v_j and v_k denotes positive item and negative item, respectively. The BPR loss function [22] is applied on the sampled data as follows:

$$\mathcal{L}^{(t)} = -\sum_{(u_i, v_j, v_k) \in \mathcal{T}_{\text{bpr}}} \log \text{sigm}(y_{i,j}^{(t)} - y_{i,k}^{(t)}) + \lambda^{(t)} \|\bar{\mathbf{H}}^{(t)}\|_{\text{F}}^2 \tag{11}$$

where the last term denotes the weight-decay regularization with weight $\lambda^{(t)}$ for preventing over-fitting.

Then, SimRec conducts joint training to optimize the parameters of the MLP-based student, during which the structure-aware node representations are distilled from advanced GNNs to over-smoothness-resistant MLPs. The training process is elaborated in A.1. Strengthened by the two distillation tasks and the regularization terms, the overall optimization objective is presented:

$$\mathcal{L}^{(s)} = \mathcal{L}_{\text{rec}} + \lambda_1 \cdot \mathcal{L}_1 + \lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3 + \lambda_4 \cdot \mathcal{L}_4$$
$$\mathcal{L}_{\text{rec}} = -\sum\nolimits_{(u_i, v_j) \in \mathcal{T}_2} y_{i,j}, \qquad \mathcal{L}_4 = \|\bar{\mathbf{H}}^{(s)}\|_{\text{F}}^2 \tag{12}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights for different optimization terms. $\mathcal{T}_2$ denotes the aforementioned set containing user-item pairs sampled

from the observed interactions $\mathcal{E}$. As the contrastive regularization $\mathcal{L}_3$ minimizes the similarity between negative user-item pairs, the recommendation objective $\mathcal{L}_{\text{rec}}$ only maximizes the similarity between positive user-item pairs. $\mathcal{L}_4$ denotes the weight-decay regularization for the MLP neural network.

3.4 Further Discussion of SimRec

3.4.1 **Adaptive High-Order Smoothing via KD.** An important strength of GNN-based CF lies in its ability to smooth user/item embeddings using their high-order neighbors. Through derivation, we show our method is able to perform the high-order smoothing in an adaptive manner. Detailed derivations are presented in A.6.2. In brief, for our light-weight GCN teacher, the embedding parameters $\bar{\mathbf{h}}_i^{(t)}, \bar{\mathbf{h}}_j^{(t)}$ of two nodes n_i, n_j (either user or item nodes) are smoothed using each other, when minimizing the following terms from the BPR loss $\mathcal{L}^{(t)}$ in Eq 11:

$$\frac{\partial \mathcal{L}_{i,j}^{(t)}}{\partial \bar{\mathbf{h}}_i^{(t)}} = \sum_{v_k} -\sigma \cdot \Big(\sum_{\mathcal{P}_{i,j}^{2L}} \prod_{(n_a,n_b)\in\mathcal{P}_{i,j}^{2L}} \frac{1}{\sqrt{d_a d_b}} \Big) \cdot \frac{\partial \bar{\mathbf{h}}_i^{(t)\top} \bar{\mathbf{h}}_j^{(t)}}{\partial \bar{\mathbf{h}}_i^{(t)}} \tag{13}$$

where $\mathcal{L}_{i,j}^{(t)}$ denotes the terms that pull close the embeddings of n_i and n_j in loss $\mathcal{L}^{(t)}$. $\sigma \in (0,1)$ is a BPR-relevant factor. $\mathcal{P}_{i,j}^{2L}$ represents a possible path between n_i and n_j with maximum length $2L$. d_a, d_b denotes the node degrees of n_a and n_b, respectively. Eq 13 reveals that GCNs smooth embeddings for high-order nodes with weighted gradients. The weights (*i.e.*, the bracketed part) encode how closely nodes are connected via multi-hop graph walks. Similarly, we analyze the gradients from our prediction-level KD $\mathcal{L}_1$ over embedding parameters $\mathbf{h}_i^{(s)}$, as follows:

$$\frac{\partial \mathcal{L}_{i,j}^{(1)}}{\partial \mathbf{h}_i^{(s)}} = \sum_{v_k} -\frac{1}{\tau_1} \cdot (\bar{z}_{i,j,k}^{(t)} - \bar{z}_{i,j,k}^{(s)}) \cdot \frac{\partial \mathbf{h}_i^{(s)\top} \mathbf{h}_j^{(s)}}{\partial \mathbf{h}_i^{(s)}} \tag{14}$$

where $\mathcal{L}_{i,j}^{(1)}$ denotes the part from $\mathcal{L}_1$ that maximizes the similarity between the embeddings of n_i and n_j. Eq 14 shows that, by utilizing the prediction-level KD, our MLP-based student can also be supercharged with high-order embedding smoothing without the cumbersome holistic-graph information propagation. Furthermore, the weights for different node pairs (*i.e.*, the bracketed part) are derived from a well-trained GCN model, instead of depending on handcrafted heuristic manners as in Eq 13. This makes our KD framework robust to the noise of observed graph structures.

3.4.2 **Enriched Supervision Augmentation via KD.** Recent works [20, 32, 34] propose to address the noise and the sparsity problems of CF by providing self-supervision signals using contrastive learning (CL) techniques. We show that our KD approach can provide even more additional supervisions. Specifically, we list the pull-close gradients from both InfoNCE-based CL loss, and our KD loss $\mathcal{L}_1$, *w.r.t*, a single node embedding $\mathbf{h}_i$, as follows:

$$\frac{\partial \mathcal{L}_{\text{CL}}}{\partial \mathbf{h}_i} = -\frac{1}{\tau} \cdot \frac{\partial \mathbf{h}_i'^{\top} \mathbf{h}_i'' / (\|\mathbf{h}_i'\|_2 \|\mathbf{h}_i''\|_2)}{\partial \mathbf{h}_i}$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{h}_i^{(s)}} = \sum_{v_j} \frac{\partial \mathcal{L}_{i,j}^{(1)}}{\partial \mathbf{h}_i^{(s)}} = \sum_{v_j, v_k} -w_{i,j,k} \cdot \frac{\partial \mathbf{h}_i^{(s)\top} \mathbf{h}_j^{(s)}}{\partial \mathbf{h}_i} \tag{15}$$

Table 1: Statistics of the experimental datasets.

Dataset	# Users	# Items	# Interactions	Interaction Density
Gowalla	25,557	19,747	294,983	5.85×10^{-4}
Yelp	42,712	26,822	182,357	1.59×10^{-4}
Amazon	76,469	83,761	966,680	1.51×10^{-4}

where $w_{i,j,k}$ represents the factors for simplicity. Shown by the second equation, our KD generates $|\{v_j, v_k | (u_i, v_j, v_k) \in \mathcal{T}_2\}|$ pull-close optimization terms for each node u_i, while CL method only generates one training sample. This evidently shows that our KD-based scheme can enrich the supplementary supervision signals, even without the data augmentation in CL [32].

3.4.3 **Complexity Analysis.** We analyze the complexity of SimRec to answer the following questions: i) How do GCNs compared to MLPs in efficiency? ii) How is the efficiency of our KD paradigm compared to state-of-the-art methods? Detailed analysis is presented in A.6.1. In concise, the computational complexity of the MLP network in SimRec is $O(|\mathcal{T}_2| \times L' \times d^2)$, and the complexity of the GCN teacher is $O(|\mathcal{E}| \times L \times d)$. The MLP student is more efficient to the GNN teacher. For the second question, the supplementary losses in SimRec takes $O(|\mathcal{T}_2| \times (I+J) \times d)$ complexity, which is comparable to existing SSL collaborative filtering methods.

4 EVALUATION

We conduct experiments from different aspects to validate the efficacy of the propose SimRec framework. The implementation details for our SimRec and the baseline methods are presented in A.2. Our experiments aim to answer the following research questions:

- **RQ1**: How does the proposed SimRec perform on different experimental datasets in comparison to state-of-the-art baselines?
- **RQ2**: How does different sub-modules of the proposed SimRec framework contribute to the overall performance?
- **RQ3**: How scalabile is SimRec in handling large-scale data?
- **RQ4**: How does the model performance vary when tuning important hyperparameters of the proposed SimRec model?
- **RQ5**: How can our SimRec model address the over-smoothing issue compared with GNN-based recommendation methods?

4.1 Experimental Settings

4.1.1 **Experimental Datasets.** Three benchmark datasets collected from real-world online services are used to evaluate the performance of SimRec. Data statistics are shown in Table 1. We split the interaction data into training set, validation set and test set with 70%:5%:25%. Details of the experimental datasets are:

- **Gowalla**: This dataset is collected from Gowalla, including user check-in records at geographical locations, from Jan to Jun, 2010.
- **Yelp**: This dataset contains users' ratings on venues, collected from Yelp platform. The time range is from Jan to Jun, 2018.
- **Amazon**: This dataset is composed of users' rating behaviors over books collected from Amazon platform, during 2013.

4.1.2 **Evaluation Protocols.** Following previous works on CF recommenders [28, 35], we conduct all-rank evaluation, in which positive items from test set are ranked with all un-interacted items for each user. The widely-used *Recall@N* and *NDCG@N* metrics [16, 32] are used adopted for evaluation, where $N = 20$ by default.

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

Data	Metric	BiasMF	NCF	AutoR	PinSage	STGCN	GCMC	NGCF	GCCF	LightGCN	DGCF	SLRec	NCL	SGL	HCCF	*SimRec*	p-val.
Gowalla	Recall@20	0.0867	0.1019	0.1477	0.1235	0.1574	0.1863	0.1757	0.2012	0.2230	0.2055	0.2001	0.2283	0.2332	0.2293	**0.2434**	$2.1e^{-8}$
	NDCG@20	0.0579	0.0674	0.0690	0.0809	0.1042	0.1151	0.1135	0.1282	0.1433	0.1312	0.1298	0.1478	0.1509	0.1482	**0.1592**	$1.2e^{-9}$
	Recall@40	0.1269	0.1563	0.2511	0.1882	0.2318	0.2627	0.2586	0.2903	0.3181	0.2929	0.2863	0.3232	0.3251	0.3258	**0.3399**	$2.4e^{-8}$
	NDCG@40	0.0695	0.0833	0.0985	0.0994	0.1252	0.1390	0.1367	0.1532	0.1670	0.1555	0.1540	0.1745	0.1780	0.1751	**0.1865**	$1.7e^{-9}$
Yelp	Recall@20	0.0198	0.0304	0.0491	0.0510	0.0562	0.0584	0.0681	0.0742	0.0761	0.0700	0.0665	0.0806	0.0803	0.0789	**0.0823**	$3.7e^{-4}$
	NDCG@20	0.0094	0.0143	0.0222	0.0245	0.0282	0.0280	0.0336	0.0365	0.0373	0.0347	0.0327	0.0402	0.0398	0.0391	**0.0414**	$3.8e^{-5}$
	Recall@40	0.0307	0.0487	0.0692	0.0743	0.0856	0.0891	0.1019	0.1151	0.1175	0.1072	0.1032	0.1230	0.1226	0.1210	**0.1251**	$4.8e^{-3}$
	NDCG@40	0.0120	0.0187	0.0268	0.0315	0.0355	0.0360	0.0419	0.0466	0.0474	0.0437	0.0418	0.0505	0.0502	0.0492	**0.0519**	$2.4e^{-4}$
Amazon	Recall@20	0.0324	0.0367	0.0525	0.0486	0.0583	0.0837	0.0551	0.0772	0.0868	0.0617	0.0742	0.0955	0.0874	0.0885	**0.1067**	$1.1e^{-10}$
	NDCG@20	0.0211	0.0234	0.0318	0.0317	0.0377	0.0579	0.0353	0.0501	0.0571	0.0372	0.0480	0.0623	0.5690	0.0578	**0.0734**	$7.0e^{-12}$
	Recall@40	0.0578	0.0600	0.0826	0.0773	0.0908	0.1196	0.0876	0.1175	0.1285	0.0912	0.1123	0.1409	0.1312	0.1335	**0.1535**	$6.6e^{-10}$
	NDCG@40	0.0293	0.0306	0.0415	0.0402	0.0478	0.0692	0.0454	0.0625	0.0697	0.0468	0.0598	0.0764	0.0704	0.0716	**0.0879**	$2.0e^{-12}$

4.1.3 **Baseline Models.** We compare SimRec with the following 14 baselines from 4 research lines for comprehensive validation.

Traditional Collaborative Filtering Technique:

- **BiasMF** [18]: It is a classic matrix factorization approach that combines user/item biases with learnable embedding vectors.

Non-GNN Neural Collaborative Filtering:

- **NCF** [13]: It is an early study of deep learning CF model that enhances the user-item interaction modeling with MLP networks.
- **AutoR** [25]: This method applies a three-layer autoencoder with fully-connected layers to encode user interaction vectors.

Graph Neural Architectures for Collaborative Filtering:

- **PinSage** [40]: This method combines random walk with graph convolutions for web-scale graph in recommendation.
- **STGCN** [42]: This method augments GCN with autoencoding sub-networks on hidden features for better inductive inference.
- **GCMC** [1]: This is a representative work to introduce graph convolutional operations into the matrix completion task.
- **NGCF** [28]: It is a GNN-based CF method which conducts graph convolutions on the user-item interaction graph for embeddings.
- **GCCF** [5] and **LightGCN**[12]: These two methods propose to simplify conventional GCN structures by removing transformations and activations for improving performance.

Disentangled GNN-based Collaborative Filtering:

- **DGCF**[29]: This method disentangles user-item interactions into multiple hidden factors in the graph message passing process.

Self-Supervised Learning Approaches for Recommendation:

- **SLRec** [39]: This method applies contrastive learning to recommendation models with feature-level data augmentations.
- **NCL** [20]: This approach enhances self-supervised graph CF models with enriched neighbor-wise contrastive learning.
- **SGL** [32]: It conducts various types of graph augmentations and feature augmentations with graph contrastive learning for CF.
- **HCCF** [34]: This method augments GNN-based CF with a global hypergraph GNN and conducts cross-view contrastive learning.

4.2 Overall Performance Comparison (RQ1)

The overall performance of SimRec and the baselines are shown in Table 2. From the results we have the following observations:

- Our SimRec consistently achieves best performance compared to baselines methods. Also, we re-train SimRec and the best-performed baselines (*i.e.*, SGL and NCL) for 5 times to calculate p-values. The experimental results validate the significance of the improvement by SimRec. Compared to the state-of-the-art GNN methods, the MLP-based inference model of our graph-less SimRec generates more accurate recommendation results, due to its adaptive contrastive knowledge distillation. Specifically, the dual-level KD in SimRec enables enriched and adaptive high-order smoothing, which not only distills the accurate dark knowledge in the well-trained GNN teacher, but also avoids being affected by the over-smoothing signals. Furthermore, the adaptive contrastive regularization automatically alleviates the over-smoothing effects, which further boosts the performance.
- While the self-supervised learning schema greatly improves the performance of GNN-based CF, our graph-less SimRec model still significantly outperforms the SSL-enhanced graph models. We attribute the performance deficiency to the inherent incapability of existing SSL frameworks in filtering over-smoothing signals. For example, SGL augments model training by introducing random noises, which may even aggravate the inaccuracy in node embeddings when the noises are magnified through high-order graph propagation. As for NCL and HCCF, they seek to connect nodes based on global semantic relatedness, which may even over-smooth nodes distant from each other in the original graph. In comparison, our graph-less SimRec model abandons GNN architectures in the inference model, which fundamentally minimizes the possibility of over-smoothed node embeddings. Furthermore, our KD paradigm avoids distilling over-smoothed embeddings via the adaptive contrastive regularization.
- We observe that non-GNN CF models (*i.e.*, NCF and AutoR) present very bad performance, event though they have similar MLP-based network architectures as the inference model in SimRec. This sheds light on the deficiency of MLPs in modeling high-order graph connectivity into user/item embeddings. While sharing similar MLP structures, our SimRec is additionally supervised by knowledge distilled from advanced GNN models. This not only improves the optimization for MLP networks, but also makes it possible to adaptively filter the over-smoothing signals in parameter learning. The huge performance gap between NCF/AutoR and our SimRec strongly shows the effectiveness of our contrastive knowledge distillation.

4.3 Model Ablation Study (RQ2)

We validate the effectiveness of the applied sub-modules in SimRec by ablating each module separately. The evaluated performance is shown in Table 3. We also show the performance change *w.r.t.* training epochs in Figure 3. We have the following observations:

- **Effect of Prediction-Level Distillation**: Our prediction-level distillation (*i.e.*, $\mathcal{L}_1$) excavates deep dark knowledge in the teacher

Table 3: Ablation study on key components of SimRec.

Data		Gowalla		Yelp		Amazon	
Variant		Recall	NDCG	Recall	NDCG	Recall	NDCG
-$\mathcal{L}_1$		0.2180	0.1415	0.0756	0.0377	0.1012	0.0692
-$\mathcal{L}_2$	User	0.2292	0.1493	0.0806	0.0405	0.0998	0.0667
	Item	0.2266	0.1477	0.0808	0.0406	0.0974	0.0649
	Both	0.2222	0.1451	0.0787	0.0399	0.0938	0.0626
-$\mathcal{L}_3$	U-I	0.2330	0.1496	0.0814	0.0410	0.0939	0.0607
	U-U	0.2349	0.1512	0.0811	0.0407	0.0965	0.0634
	I-I	0.2331	0.1514	0.0813	0.0409	0.1009	0.0674
	All	0.2282	0.1480	0.0810	0.0407	0.0933	0.0605
SimRec		**0.2434**	**0.1592**	**0.0823**	**0.0414**	**0.1067**	**0.0734**

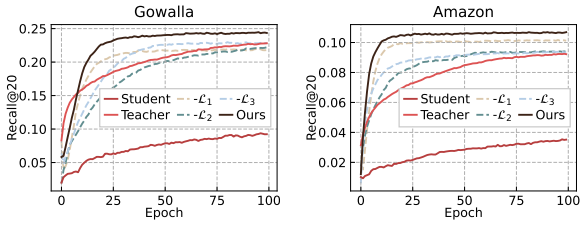


Figure 3: Test performance in each epoch for ablated models.

Table 4: Model performance and per-epoch model inference time of representative methods on large-scale Tmall dataset.

Metric	# Edges	DGCF	SGL	HCCF	NCL	*SimRec*
R@20	1.6M	0.0221	0.0258	0.0272	0.0286	**0.0308**
	2.9M	0.0253	0.0278	0.0283	0.0294	
N@20	1.6M	0.0258	0.0296	0.0309	0.0337	**0.0366**
	2.9M	0.0279	0.0311	0.0319	0.0334	
Time	1.6M	7190.2s	1331.8s	1342.5s	1392.2s	**785.1s**
	2.9M	11431.8s	1456.3s	1530.8s	1693.8s	

using the pair-wise ranking task with enriched KD samples. The variant -$\mathcal{L}_1$ removes this module, which leads to performance degradation on Gowalla and Yelp data. The results validate the effectiveness of learning from the predictive outputs of teacher model using our distillation loss $\mathcal{L}_1$.

- **Effect of Embedding-Level Distillation**: We then test the effect of embedding-level KD with the variant -$\mathcal{L}_2$ by removing $\mathcal{L}_2$ on user/item embeddings. In some cases the alignment between users and the alignment between items have different effect on the performance. What's more, the results reveal not only the contribution of $\mathcal{L}_2$ to the final performance, but also its prominent accelerating effect in model training shown in Fig 3.
- **Effect of Contrastive Regularization**: We ablate SimRec without the contrastive regularization in variant -$\mathcal{L}_3$. The regularization for user-item, user-user, and item-item relatedness are individually ablated. We observe the importance of $\mathcal{L}_3$ for the superior performance, especially on Amazon data. We ascribe this to the larger scale of Amazon data which makes it more likely to over-smooth with irrelevant high-order neighbors. The incorporation of $\mathcal{L}_3$ can cancel out over-smoothing signals.
- **Comparison to Student and Teacher Models**: From the learning curves in Fig 3, we can observe the great performance gap between simple MLP student and advanced GNN teacher. The three augmented tasks greatly minimizes this gap by effectively distilling useful knowledge. Additionally, the distillation tasks accelerate the training to surpass the original teacher model.

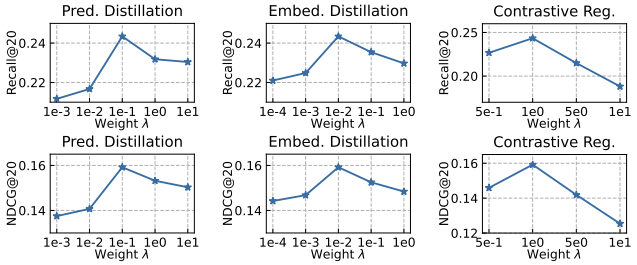
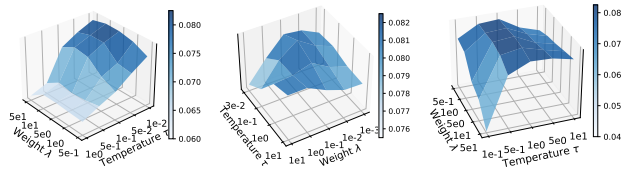


Figure 4: Hyperparameter study for our SimRec model on Gowalla dataset, in terms of *Recall@20* and *NDCG@20*.



(a) Pred. Distillation (b) Embed. Distillation (c) Contrastive Reg.

Figure 5: Impact of weights and temperature in different learning objectives on Yelp, in terms of *Recall@20*.

4.4 Model Scalability Study (RQ3)

To validate the efficiency of our SimRec in handling large-scale real-world data, we compare SimRec with the best performed baselines on a e-commerce data collected from Tmall platform. The dataset contains around 40 million records of user clicks. To successfully run on this dataset, GNN-based methods have to sample subgraphs for information propagation. In contrast, graph sampling is not required by the MLP-based inference model of our SimRec. The performance and the inference time are shown in Table 4, where we run the baselines using graph sampling strategy [15] with two scales (*i.e.*, subgraphs contain 1.6M edges and 2.9M edges, respectively). We have mainly two key observations shown as follows:

- **More Accurate Recommendations**: SimRec achieves better recommendation performance in terms of Recall and NDCG. This reflects the higher probability of over-smoothing on the large but sparse interaction graph. Our SimRec avoids this problem without explicit graph message passing. Instead, informative knowledge is distilled from GNNs for model compression.
- **Much Higher Efficiency**: SimRec greatly reduces the inference time on the large Tmall data. *Firstly*, the embedding process of our MLP predictor is agnostic to the holistic interaction graph, thus the large-scale graph does not increase much overhead for embedding processing. No graph sampling is required in comparison to GNNs. *Secondly*, SimRec infers user-item relations based on simple MLPs. The computational costs of fully-connected layers in MLPs are much lower than the cost of GNNs.

4.5 Hyperparameter Study (RQ4)

In this section, we examine the influence of different hyperparameters on the performance of SimRec. The effect of loss weights $\lambda_1, \lambda_2, \lambda_3$ are shown in Figure 4. The composite effect of loss weights and corresponding temperatures τ_1, τ_2, τ_3 are shown in Figure 5. The effect of the size $|\mathcal{T}_1|$ for the prediction-level distillation is shown in Table 5. Our observations are as follows:

Table 5: Investigation on the impact of batch size in the prediction-oriented distillation of the proposed SimRec.

Data	Metric	Batch Size $\|\mathcal{T}_1\|$ in Prediction-Level Distillation					
		$1e3$	$5e3$	$1e4$	$5e4$	$1e5$	$5e5$
Gowalla	Recall	0.2208	0.2361	0.2399	0.2420	0.2434	0.2448
	NDCG	0.1441	0.1530	0.1554	0.1577	0.1592	0.1597
Yelp	Recall	0.0443	0.0730	0.0773	0.0802	0.0823	0.0822
	NDCG	0.0210	0.0372	0.0392	0.0407	0.0414	0.0414

- **Strength of Prediction-Level Distillation**. λ_1, τ_1: This weight λ_1 and temperature τ_1 jointly control the strength of the prediction-level KD λ_1. We first study the influence of λ_1 in Figure 4 with τ_1 fixed. When λ_1 is small, not enough knowledge is distilled to the student model which results in deficient performance. When λ_1 is too large, $\mathcal{L}_1$ cover up the optimization of main loss and yield degraded performance. Additionally, Figure 5(a) shows the positive effect of applying smaller τ_1 to produce larger gradients.
- **Strength of Embedding-Level Distillation**. λ_2, τ_2: The parameters control the strength of SimRec in restricting the embeddings in MLP to be close to embeddings in GNN. From Figure 5(b) it can be observed that λ_2 and τ_2 jointly adjust the strength of embedding KD to have modest influence on optimization, to prevent from insufficient knowledge distillation and too-strict embedding regularization. Either large weight with low temperature or small weight with high temperature causes performance decay.
- **Strength of Contrastive Regularization** λ_3, τ_3: These parameters determine the strength of push-away regularization for preventing over-smoothing. The results show that either too small weight λ_3 or too high temperature τ_3 causes insufficient regularization and produces over-smoothed embeddings. Meanwhile, strong regularization may damage the modeling of node-wise affinity, and also yields worse performance.
- **Per-Batch Number of Samples to Distill** $|\mathcal{T}_1|$: This hyperparameter determines how many instances are sampled to conduct the prediction-level distillation in each training step. According to the results in Table 5, increasing batch size brings better KD performance until the performance saturates. We ascribe this to the effect that larger batch size filters low-frequency noise in predictions made by the teacher model in SimRec.

4.6 Over-Smoothing Investigation (RQ5)

To investigate whether our graph-less SimRec framework is able to mitigate the over-smoothing effect in graph-structured relation learning for CF, we compare representative baselines and our SimRec model on the Mean Average Distance (MAD) values [3] over embeddings for the most popular users and items. The evaluation results are shown in Table 6. Our SimRec has higher MAD values on both user and item embeddings for Gowalla and Yelp data, in comparison to not only GCN model GCCF, but also state-of-the-art SSL frameworks. It can be concluded that our SimRec framework better addresses the over-smoothing issue, by learning more uniform-distributed embeddings for users and items, to better characterize their unique interaction patterns. This should be attributed to the MLP-based inference framework, and the contrastive regularization that adaptively alleviates over-smoothing signals.

5 RELATED WORK

Graph-based Collaborative Filtering Inspired by the success of GNNs, a lot of research works have designed various graph neural architectures to build collaborative recommender systems [10, 33]. For example, to model user-item interactions graph, many efforts have been devoted to developing powerful GNN models for message passing, *e.g.* NGCF [28], STGCN [42] and GCMC [1]. GCCF [5] and LightGCN [12] enrich GNNs in CF by simplifying the GCN architecture. To increase model scalability and prevent over-smoothing in making recommendations, our SimRec abandons graph encoders in the inference model, and conducts soft embedding smoothing by distilling useful knowledge from the GNN-based teacher model.

Table 6: Investigation on the ability to address the over-smoothing effect on Gowalla and Yelp data in terms of MAD.

Data		GCCF	LightGCN	SGL	NCL	HCCF	*SimRec*
Gowalla	User	0.8276	0.8203	0.8412	0.8088	0.8394	**0.8576**
	Item	0.7579	0.7614	0.7702	0.8169	0.7905	**0.8335**
Yelp	User	0.9226	0.9610	0.9755	0.9640	0.9749	**0.9819**
	Item	0.6288	0.7095	0.7191	0.6953	0.6246	**0.7662**

Self-Supervised Learning (SSL) for Recommendation. To tackle the challenge of noise and sparsity in recommendation systems, recent research has explored various types of SSL techniques for data augmentation [6, 30, 35, 41]. For instance, some studies, such as SGL [32], introduce random perturbation to generate additional views for CL. Other approaches, such as HCCF [34] and NCL [20], incorporate global views to produce semantically related pairs for CL. While these SSL methods have shown promise in addressing the issues caused by noisy and sparse data, they often heavily rely on graph neural networks (GNNs) to generate embeddings. This can result in an over-smoothing effect, limiting the overall representation ability of the recommendation framework.

Knowledge Distillation for Recommendation. Knowledge distillation aims to transfer knowledge from a complex and well-trained teacher model to a simpler student model [43]. It utilizes the predictions of the teacher model to generate informative soft targets for the student model to learn from. In the context of recommender systems, knowledge distillation has been used to develop simpler yet effective models [17, 19, 26, 36]. As examples, Tang *et al.* [26] proposes a method to leverage knowledge distillation for ranking tasks in recommender systems. Xia *et al.* [36] develop highly-efficient models for on-device recommendations with effective knowledge transferring. Unlike the works that primarily focus on model reduction, our proposed approach, SimRec, aims to address the over-smoothing issue in state-of-the-art GNN-based CF models. By distilling unbiased signals from GNNs to simple multilayer perceptrons (MLPs), we can reduce the over-smoothing effect and enhance the model representation ability.

6 CONCLUSION

In this paper, we propose a contrastive knowledge distillation model which adaptively transfers knowledge from the GNN-based teacher model to a small feed-forward network, significantly improving the efficiency and robustness of recommender models. Our designed adaptive contrastive regularization generate unbiased self-supervision signals to alleviate the over-smoothing and noise effects commonly exist in recommender systems. Our comprehensive experiments demonstrate the effectiveness of our method in improving recommendation accuracy and achieving better efficiency when compared to state-of-the-art learning techniques.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In *International Conference on Knowledge Discovery & Data Mining (KDD)*.

[2] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *ICLR*.

[3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *International Conference on Artificial Intelligence (AAAI)*. 3438–3445.

[4] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 21–30.

[5] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *International Conference on Artificial Intelligence (AAAI)*, Vol. 34. 27–34.

[6] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous Graph Contrastive Learning for Recommendation. In *International Conference on Web Search and Data Mining (WSDM)*. 544–552.

[7] Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling Knowledge Learned in BERT for Text Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 7893–7905.

[8] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc Le. 2019. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 5931–5937.

[9] Claudio Gallicchio and Alessio Micheli. 2020. Fast and deep graph neural networks. In *International Conference on Artificial Intelligence (AAAI)*, Vol. 34. 3898–3905.

[10] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, et al. 2021. Graph neural networks for recommender systems: challenges, methods, and directions. *Transactions on Information Systems (TOIS)* (2021).

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *International Conference on Computer Vision and Pattern (CVPR)*. 770–778.

[12] Xiangnan He, Kuan Deng, Xiang Wang, et al. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 639–648.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *The Web Conference (WWW)*. 173–182.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *The Web Conference (WWW)*. 2704–2710.

[16] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *International Conference on Artificial Intelligence (AAAI)*, Vol. 35. 4115–4122.

[17] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A knowledge distillation framework for recommender system. In *International Conference on Information and Knowledge Management (CIKM)*. 605–614.

[18] Yehuda Koren, Robert Bell, et al. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[19] Jae-woong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative distillation for top-N recommendation. In *International Conference on Data Mining (ICDM)*. IEEE, 369–378.

[20] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning. In *The Web Conference (WWW)*. 2320–2329.

[21] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 338–348.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461.

[23] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Recsys*. 240–248.

[24] Muhamad Risqi U Saputra, Pedro PB De Gusmao, Yasin Almalioglu, Andrew Markham, and Niki Trigoni. 2019. Distilling knowledge from a deep pose regressor network. In *International Conference on Computer Vision and Pattern (CVPR)*. 263–272.

[25] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *The Web Conference (WWW)*. 111–112.

[26] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 2289–2298.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *International Conference on Research and Development in Information Retrieval (SIGIR)*.

[29] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 1001–1010.

[30] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive meta learning with behavior multiplicity for recommendation. In *International Conference on Web Search and Data Mining (WSDM)*. 1120–1128.

[31] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*. PMLR, 6861–6871.

[32] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 726–735.

[33] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[34] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 70–79.

[35] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 2100–2109.

[36] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. 2022. On-Device Next-Item Recommendation with Self-Supervised Knowledge Distillation. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 546–555.

[37] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems.. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 17. Melbourne, Australia, 3203–3209.

[38] Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. 2020. Tinygnn: Learning efficient graph neural networks. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 1848–1856.

[39] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, et al. 2021. Self-supervised Learning for Large-scale Item Recommendations. In *International Conference on Information and Knowledge Management (CIKM)*. 4321–4330.

[40] Rex Ying, Ruining He, Kaifeng Chen, et al. 2018. Graph convolutional neural networks for web-scale recommender systems. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 974–983.

[41] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In *The Web Conference (WWW)*. 413–424.

[42] Jiani Zhang, Xingjian Shi, Shenglin Zhao, et al. 2019. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*.

[43] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2022. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *International Conference on Learning Representations (ICLR)*.

[44] Yiman Zhang, Hanting Chen, Xinghao Chen, Yiping Deng, Chunjing Xu, and Yunhe Wang. 2021. Data-free knowledge distillation for image super-resolution. In *International Conference on Computer Vision and Pattern (CVPR)*. 7852–7861.

[45] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal intervention for leveraging popularity bias in recommendation. In *International Conference on Research and Development in Information Retrieval (SIGIR)*. 11–20.

[46] Chenguang Zheng, Hongzhi Chen, Yuxuan Cheng, Zhezheng Song, Yifan Wu, Changji Li, James Cheng, Hao Yang, and Shuai Zhang. 2022. ByteGNN: efficient graph neural network training at large scale. *International Conference on Very Large Data Bases (VLDB)* 15, 6 (2022), 1228–1242.

[47] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. 2020. Towards deeper graph neural networks with differentiable group normalization. *Neural Information Processing Systems (NeurIPS)* 33 (2020), 4917–4928.

[48] Hao Zhu and Piotr Koniusz. 2020. Simple spectral graph convolution. In *International Conference on Machine Learning (ICML)*.

A APPENDIX

A.1 Learning Algorithm of SimRec

The parameter learning for our SimRec is elaborated in Algorithm 1

Algorithm 1: Learning Process of SimRec

Input: User-item interaction matrix $\mathbf{A}$, loss weights and temperature factors $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda^{(t)}, \tau_1, \tau_2, \tau_3$, learning rate η, maximum training epochs E, number of graph iterations L, number of MLP layers L'.

Output: Trained embeddings $\bar{\mathbf{H}}^{(s)}$ and MLP parameters $\mathbf{W}$.

Initialize model parameters $\bar{\mathbf{H}}^{(s)}, \bar{\mathbf{H}}^{(t)}, \mathbf{W}$

Train the GCN teacher model for well-trained $\bar{\mathbf{H}}^{(t)}$ (Eq 11)

for $e = 1$ *to* E **do**

  for *mini-batch* $\mathcal{T}_2$ *drawn from* $\mathcal{E}$ **do**

    Sample a batch of triplet $\mathcal{T}_1$

    Calculate preference difference $z^{(s)}_{i,j,k}, z^{(t)}_{i,j,k}$ for samples in $\mathcal{T}_1$ (Eq 6)

    Compute loss $\mathcal{L}_1$ for prediction-level KD (Eq 7)

    Calculate loss $\mathcal{L}_2$ for embedding-level KD (Eq 8)

    Calculate the adjustment factor ω_i, ω_j for users and items in $\mathcal{T}_2$ (Eq 10)

    Compute loss $\mathcal{L}_3$ for contrastive regularization (Eq 9)

    Calculate $\mathcal{L}_{\text{rec}}$ for recommendation task

    Calculate $\mathcal{L}_4$ for weight-decay regularization

    Calculate overall loss $\mathcal{L}^{(s)}$ for the student (Eq 15)

    for *each parameter* θ *in* $\{\bar{\boldsymbol{H}}^{(s)}, \boldsymbol{W}\}$ **do**

      $\theta = \theta - \eta \cdot \partial \mathcal{L}^{(s)} / \partial \theta$;

    end

  end

end

return all parameters $\bar{\mathbf{H}}^{(s)}, \mathbf{W}$

A.2 Implementation Details

For fair comparison, we present the hyperparameter settings for implementing the proposed SimRec framework and the baseline methods. Specifically, our SimRec is implemented with PyTorch, using Adam optimizer and Xavier initializer with default parameters. Training batch size is set as $|\mathcal{T}_1| = 100000, |\mathcal{T}_2| = 4096$. The dimensionality of embedding vectors is set as 32. The number of MLP layers is selected from $\{1, 2, 3\}$. The number of graph iterations for the teacher model is selected from $\{2, 4, 6\}$. The loss weights $\lambda_1, \lambda_2, \lambda_3$ are tuned from $\{10, 3, 1, 0.3, 0.1, 0.03, 0.01\}$, and the weights $\lambda_4, \lambda^{(t)}$ for weight-decay regularization are tuned from $\{1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}, 1e^{-8}, 0\}$. The temperatures τ_1, τ_2, τ_3 are chosen from $\{10, 3, 1, 0.3, 0.1, 0.03, 0.01\}$. Parameter ε for contrastive regularization adjustment is set as 0.2.

For the baseline methods, we apply the same Adam optimization algorithm, Xavier parameter initializer, and batch size of 4096 as our SimRec. The hidden dimensionality for all baselines is also set as 32. Hyperparameters that are shared by baseline methods and our SimRec, are tuned in the same range as above. Such hyperparameters include the number of GNN layers, the weight for weight-decay regularizer. Specifically, for NCL, HCCF, SGL, SLRec, the weight for supplementary tasks are tuned from $\{1e^{-k}, 3e^{-k} | -1 \leq k \leq 6\}$. The temperature hyperparameters are tuned from $\{1e^{-k}, 3e^{-k} | 2 \leq k \leq -1\}$. For NCL, which conducts K-Means clustering every n epochs, we tune n from $\{1, 2, 3, 4, 5\}$. For baseline methods that employs random message dropout (*e.g.*, LightGCN, SGL), the dropout rate is tuned from $\{0.1, 0.2, 0.3, 0.5, 0.8, 0.9\}$. For models that were trained for rating predictions in the original paper (*e.g.*, AutoR, ST-GCN), we train these methods using pair-wise BPR loss for implicit feedback. For NCF, we adopt the NeuMF version which combines MLPs with Generalized MF.

A.3 Ablation Study

We show more results of ablation study in Figure 6, including the Recall@20 and NDCG@20 results on Yelp data, and NDCG@20 results on Gowalla and Amazon data. We can observe that the dual-level knowledge distillation schema and the adaptive contrastive regularization in our SimRec framework significantly improves the performance of the simple MLP model, to even surpass the performance of the GCN-based teacher model. From the results on Yelp data, it can be observed that removing the prediction-level KD causes severe over-fitting. This strongly validates the importance of distilling from the predictions made by the teacher model. Removing the embedding-level distillation, also causes significant performance drop and prominently lower learning efficiency on Yelp data. In comparison, the CL regularization contributes less to the performance of SimRec on Yelp data, which is due to its smaller interaction set that makes it less likely to over-smooth embeddings.

A.4 Visualization for Embeddings Distribution

We show more visualization results for the embedding distribution *w.r.t* NCL in Figure 7. The visualization is done by first compressing the learned embeddings into a 2-d space using t-SNE dimension reduction. Then the scatter plot is smoothed using Gaussian kernel density estimation (KDE) to estimate the distribution of the embeddings. As shown by Figure 1(c) and Figure 7, our SimRec learns to allocates users into a bigger sub-space. In contrast, the baseline methods rely on iterative graph information propagation, which over-smooths the node embeddings to be too similar. From the visualization for the baselines, we can observe that the GNN frameworks over-smooth the user embeddings too much, such that users are split into several prominent subspaces disconnected to each other. This greatly hinders the CF models from learning relations between users from different subspaces.

A.5 Hyperparameter Study

We further investigate the influence of hidden dimensionality in our SimRec for the model performance. Specifically, we first train GCN-based teacher models with different hidden dimensionality (8, 16, 32, 64), and then distill the teacher model to a MLP-based student model with the same embedding size. As shown by results in Figure 8, the performance shows a typical under-fitting to over-fitting curve *w.r.t* the hyperparameter d on different datasets. After d reaches the default embedding size 32, the performance increases slightly on Yelp dataset. Instead, the performance still prominently

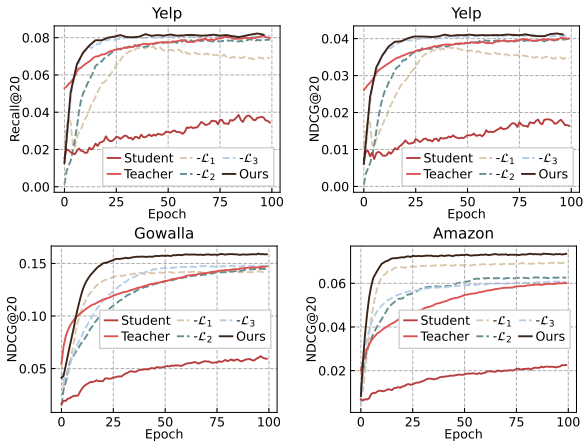
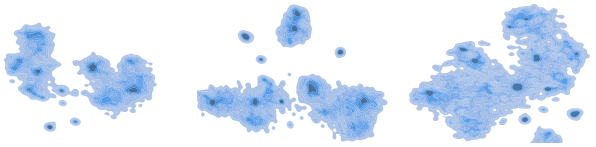


Figure 6: Test performance in each epoch for ablated models on three experimental datasets in terms of Recall and NDCG.



(a) NCL (b) HCCF (c) SimRec

Figure 7: KDE visualization for distribution of embeddings learned by NCL, HCCF and the proposed SimRec.

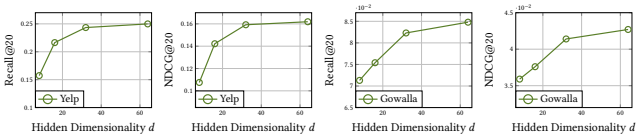


Figure 8: Hyperparameter study for hidden dimensionality of SimRec in terms of Recall and NDCG on Yelp and Gowalla.

grows when d increases from 32 to 64 on Gowalla data. This could be attributed to the larger scale of interaction records and the lower sparsity degree of Gowalla data.

A.6 Theoretical Analysis

A.6.1 **Detailed Complexity Analysis.** The complexity analysis is to answer the following two questions: i) How do GCNs compare to MLPs in efficiency? ii) What is the overhead of our KD paradigm? In each training step, GNN-based CF methods must conduct whole-graph information propagation for the embedding process. This takes $O(|\mathcal{E}| \times L \times d)$ complexity for our lightweight GCN. The prediction phase of our GCN takes $O(|\mathcal{T}_{\text{bpr}}| \times d)$ for computing dot-product. In comparison, the embedding process of MLP is not in graph-level but focus on one embedding vector at once. It costs $O(|\mathcal{T}_2| \times L' \times d^2)$ where $|\mathcal{T}_2| = |\mathcal{T}_{\text{bpr}}| \ll |\mathcal{E}|/d$. It also requires $O(|\mathcal{T}_{\text{bpr}}| \times d)$ computational cost to predict in each training batch.

Our prediction-level KD $\mathcal{L}_1$ requires $O(|\mathcal{T}_1| \times d)$ cost. The $\mathcal{L}_2$ KD takes $O(|\mathcal{T}_2| \times d)$ for the numerators, and $O(|\mathcal{T}_2| \times J \times d)$ for the denominators. Similar to the second term for $\mathcal{L}_2$, the contrastive regularization $\mathcal{L}_3$ costs $O(|\mathcal{T}_2| \times (I+J) \times d)$ computations. In conclusion, the KD of our SimRec has the total time complexity of $O(|\mathcal{T}_2| \times (I+J) \times d)$, which is comparable to the state-of-the-art CF methods (*e.g.*, self-supervised methods SGL [32], NCL [20]). Note that although the training process has the same complexity, our SimRec conducts inference with simple MLPs which is much more efficient as discussed above.

A.6.2 **Derivations for High-Order Smoothing.** In this section, we present details for the derivations related to Section 3.4.1. To begin with, we show the high-order smoothing effect of the GCN teacher in the perspective of gradients, which yield the results in Eq 13. Specifically, the gradients that maximize the similarity between $\bar{\mathbf{h}}_i^{(t)}$ and $\bar{\mathbf{h}}_j^{(t)}$, given by the loss $\mathcal{L}^{(t)}$ is as follows:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{i,j}}{\partial \bar{\mathbf{h}}_i} &= -\sum_{u_i,v_j,v_k} \frac{\partial \log \text{sigm}(z_{i,j,k})}{\partial \bar{\mathbf{h}}_i} = -\sum_{u_i,v_j,v_k} \sigma \frac{\partial z_{i,j,k}}{\partial \bar{\mathbf{h}}_i} \\
&= -\sum_{u_i,v_j,v_k} \sigma \frac{\partial \bar{\mathbf{h}}_i^\top \mathbf{h}_j}{\partial \bar{\mathbf{h}}_i} = -\sum_{v_k} \sigma \sum_{n_{i'},n_{j'}} \sum_{\mathcal{P}^L_{i,i'},\mathcal{P}^L_{j,j'}} \frac{\partial \bar{\mathbf{h}}_i^\top \mathbf{h}_{j'}}{\partial \bar{\mathbf{h}}_i} \\
&= -\sum_{v_k} \sigma \sum_{n_{i'},n_{j'}} \sum_{\mathcal{P}^L_{i,i'},\mathcal{P}^L_{j,j'}} \prod_{(n_a,n_b)\in\mathcal{P}^L_{i,i'}} \frac{1}{\sqrt{d_a d_b}} \\
&\quad \prod_{(n_a,n_b)\in\mathcal{P}^L_{j,j'}} \frac{1}{\sqrt{d_a d_b}} \frac{\partial \bar{\mathbf{h}}_i^\top \bar{\mathbf{h}}_j}{\partial \bar{\mathbf{h}}_i} \\
&= \sum_{v_k} -\sigma \cdot \Big(\sum_{\mathcal{P}^{2L}_{i,j}} \prod_{(n_a,n_b)\in\mathcal{P}^{2L}_{i,j}} \frac{1}{\sqrt{d_a d_b}} \Big) \cdot \frac{\partial \bar{\mathbf{h}}_i^\top \bar{\mathbf{h}}_j}{\partial \bar{\mathbf{h}}_i}
\end{aligned} \tag{16}
$$

where σ denotes $1 - \text{sigm}(z_{i,j,k})$. For simplicity, we omit the (t) superscript. As $\mathcal{L}_{i,j}$ refers to the pull-close terms, $-\mathbf{h}_i^\top \mathbf{h}_k$ is omitted. Next, we show the details of derivations that obtain Eq 14 as follows:

$$
\begin{aligned}
\mathcal{L}_1 &= \sum_{u_i,v_j,v_k} -\bar{z}^{(t)}_{i,j,k} \cdot \log \bar{\mathbf{z}}^{(s)}_{i,j,k} + (\bar{z}^{(t)}_{i,j,k} - 1) \cdot \log(1 - \bar{\mathbf{z}}^{(s)}_{i,j,k}) \\
&= \sum_{u_i,v_j,v_k} \bar{z}^{(t)}_{i,j,k} \cdot \log \frac{1 - \bar{z}^{(s)}_{i,j,k}}{\bar{z}^{(s)}_{i,j,k}} - \log(1 - \bar{z}^{(s)}_{i,j,k}) \\
&= \sum_{u_i,v_j,v_k} -\bar{z}^{(t)}_{i,j,k} z^{(s)}_{i,j,k}/\tau_1 + \log(1 + \exp(z^{(s)}_{i,j,k}/\tau_1)) \\
\frac{\partial \mathcal{L}_1}{\partial \mathbf{h}_i^{(s)}} &= \sum_{u_i,v_j,v_k} \frac{\text{sigm}(z^{(s)}_{i,j,k}/\tau_1)}{\tau_1} \frac{\partial \mathbf{h}_i^{(s)\top} \mathbf{h}_j^{(s)}}{\partial \mathbf{h}_i^{(s)}} - \frac{1}{\tau_1} \bar{z}^{(t)}_{i,j,k} \frac{\partial \mathbf{h}_i^{(s)\top} \mathbf{h}_j^{(s)}}{\partial \mathbf{h}_i^{(s)}} \\
&= \sum_{u_i,v_j,v_k} -\frac{1}{\tau_1} \cdot (\bar{z}^{(t)}_{i,j,k} - \bar{z}^{(s)}_{i,j,k}) \cdot \frac{\partial \mathbf{h}_i^{(s)\top} \mathbf{h}_j^{(s)}}{\partial \mathbf{h}_i^{(s)}}
\end{aligned} \tag{17}
$$

From the derivation above, we can observe that GCN conduct high-order embedding smoothing using the cumulative product of node degrees as weights. This manner is restricted by the graph structures and may be affected by noisy edges. Instead, our developed SimRec uses knowledge distillation to perform adaptive high-order smoothing for any user-item pair u_i, v_j, using the teacher model's predictions as guidance during the model compression process. This allows the lightweight student model to effectively learn from the teacher's knowledge and make accurate predictions.