Multi-View Denoised Graph Collaborative Filtering for Third-Party Library Recommendation

Yuxiang Kuang[1,2], Xiaoxiang Liao[1,2], Guosheng Kang[1,2(✉)], Ye Cao[1,2], Xiao Cui[3], Rong Hu[1,2], Jiayan Xiang[1,2]
[1]Hunan Provincial Key Lab. for Services Computing and Novel Software Technology, HNUST, Xiangtan, Hunan, China
[2]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China
[3]Leiden Institute of Advanced Computer Science, Faculty of Science, Leiden University, Leiden, The Netherlands
Email: {yuxiangkuang1001, liaoxiaoxiang05, guoshengkang, caoye6233, x.cui1211, ronghu, jiayanxiang02}@gmail.com

***Abstract*—Third-party libraries are crucial for accelerating software development, yet identifying suitable candidates within the expanding open-source ecosystem remains challenging. Graph neural network (GNN)-based collaborative filtering emerges as a promising solution due to its ability to effectively model high-order interaction relationships. However, data sparsity leads to biased representations, hindering effective node representation learning. Moreover, standard multi-layer convolutions blend irrelevant node features, causing noise accumulation and sub-optimal embeddings. To address these challenges, a Multi-View Denoised Graph Collaborative Filtering (MDGCF) approach is proposed for TPL recommendation. Specifically, MDGCF constructs complementary graph views to alleviate sparsity and employs an implicit supervision module to guide representation learning with auxiliary signals. To mitigate noise, we propose a denoised graph convolution that calculates normalized weights for even-order nodes to suppress irrelevant propagation, alleviating noise accumulation. Furthermore, a sampled softmax loss with graph topology-based negative sampling is carefully designed to replace the conventional BPR loss, strengthening the differentiation between positive and negative samples and enhancing recommendation accuracy. Experiments on a real-world dataset demonstrate that MDGCF outperforms state-of-the-art baselines, validating the model's effectiveness in improving the performance of TPL recommendation.**

***Index Terms*—Similarity Supervision, Denoised Convolution, Graph Neural Network, Third-Party Library Recommendation**

I. Introduction

In modern software development, reusing third-party libraries (TPLs) is essential for accelerating development and reducing cost [1, 2]. However, the rapid expansion of open-source repositories offers thousands of candidate libraries with overlapping functionalities. Manual selection from such a vast pool is labor-intensive and error-prone, directly impacting software quality. Consequently, automated TPL recommendation systems emerge as critical tools to assist developers in efficiently identifying suitable libraries.

While early approaches employ collaborative filtering [3] or matrix factorization, graph neural networks (GNNs) recently advance the field by modeling App-TPL interactions as bipartite graphs. Methods such as GRec [4], HCF [5], and ISGCF [6] propagate embeddings to capture collaborative signals. Despite their success, several limitations still exist. First, data sparsity remains a persistent bottleneck. Scarce interaction records hinder neighborhood aggregation, especially for long-tail TPLs, resulting in suboptimal embeddings that fail to capture complex latent dependencies in the graph. Although self-supervised learning methods such as SGL [7], NCL [8], and SimGCL [9] are proposed to augment data views, they often introduce noise or false negatives. Second, multi-layer convolution aggregation in standard GNNs [10] tends to blend irrelevant node features, causing noise accumulation and over-smoothing issues [11]. While ISGCF [6] attempts to mitigate the over-smoothing problem via implicit supervision, it primarily operates on similarity graphs. Third, from an optimization perspective, the widely used Bayesian Personalized Ranking (BPR) loss [12] relies on a uniform negative sampling strategy pairing a single negative sample with a positive one, often failing to provide sufficient contrast in real-world applications.

To address these challenges, we propose a novel method, **MDGCF** (**M**ulti-View **D**enoised **G**raph **C**ollaborative **F**iltering for Third-Party Library Recommendation). MDGCF alleviates data sparsity by constructing complementary graph views and leveraging implicit supervision to guide representation learning with collaborative signals. To mitigate noise, a denoised interaction graph convolution mechanism is designed for even-order nodes to suppress information propagation between irrelevant nodes, effectively alleviating noise accumulation from multi-layer aggregation. Additionally, a sampled softmax loss (SSM) [13] with graph topology-based negative sampling strategy is designed to replace the conventional BPR loss. By leveraging multi-negative contrast, this mechanism strengthens the differentiation between invoked and non-invoked TPLs for more discriminative optimization. In general, the main contributions of this work are summarized as follows.

- We propose a multi-view denoised graph collaborative filtering approach that captures complementary structural information to mitigate data sparsity.
- To enhance the representation of App-TPL interaction graph, a novel denoised graph convolution is proposed to suppress noise during the convolution process.
- A sampled softmax loss with graph topology-based negative sampling is carefully designed to replace the BPR loss, enhancing differentiation between positive and negative samples to improve recommendation performance.
- Comprehensive experiments are conducted on the real-world dataset, demonstrating that MDGCF outperforms state-of-the-art baselines in TPL recommendation.

The remainder of this paper is organized as follows. Section II reviews the related work on third-party library recommendation. Section III introduces the proposed method MDGCF in detail. Section IV evaluates the effectiveness of MDGCF through experiments. Finally, Section V concludes our work.

II. Related Work

Third-party library recommendation aims to assist developers in efficiently selecting suitable libraries that meet their development needs. While early approaches focus on mining historical patterns, graph representation learning is increasingly adopted in recommender systems to capture high-order collaborative signals. Accordingly, related work on TPL recommendation and graph-based models is introduced in detail.

A. TPL Recommendation

Existing TPL recommendation methods can be generally categorized into traditional mining-based approaches and deep learning-based approaches. Thung et al. [14] propose LibRec to recommend libraries based on feature requests by combining association rule mining with collaborative filtering. Yu et al. [3] introduce AppLibRec to integrate topic modeling with collaborative filtering for uncovering implicit usage patterns. He et al. [1] develop LibSeek by using an adaptive weighting mechanism to balance the importance of different collaborative signals. Nguyen et al. [15] propose CrossRec to model project-library relationships by mining cross-project collaborative patterns. With the advancement of deep learning, graph neural networks are increasingly adopted to capture high-order signals [16]. Li et al. [4] propose GRec to learn node embeddings by applying GNNs to the App-TPL interaction graph. Jin et al. [17] introduce a neighbor-aware GNN to enhance representation learning by differentiating the importance of neighbors. Chen et al. [5] propose HCF to capture complex correlations beyond simple pairwise interactions by utilizing hypergraph convolutions. Li et al. [18] explore embedding project-library knowledge graphs to enrich semantic representations with external knowledge. Chen et al. [6] propose ISGCF to mitigate the over-smoothing issue [11] by constructing similarity graphs and leveraging implicit supervision. However, these works only focus on single view, i.e., interaction graph view or similarity graph view, hindering the sufficient feature extraction.

B. Graph Representation Learning for Recommendation

Graph representation learning emerges as a dominant paradigm in recommender systems, improving performance by encoding complex bipartite interactions. He et al. [19] propose LightGCN to simplify standard GCNs by removing nonlinear activation and feature transformation. To mitigate data sparsity, self-supervised learning and graph contrastive learning are widely explored to augment interaction data. Wu et al. [7] propose SGL to generate augmented views via node or edge dropout for maximizing the mutual information between views. Yu et al. [9] propose SimGCL to simplify augmentation by injecting uniform noise into embeddings rather than altering graph structures. Lin et al. [8] propose NCL to incorporate structural and semantic neighbors into the contrastive learning framework. Cai et al. [20] propose LightGCL to utilize singular value decomposition for guiding contrastive augmentation with global structural signals. Although these methods effectively alleviate data sparsity, they typically rely on stochastic augmentations and may lack specific mechanisms for mitigating noise accumulation caused by multi-layer convolution aggregation.

In summary, while graph-based methods advance TPL recommendation, they face persistent challenges regarding data sparsity and noise accumulation. Although implicit supervision strategies [6] demonstrate efficacy, exploring denoised convolution on the interaction graph is vital for robust signal propagation. Furthermore, the single negative sampling strategy of the BPR loss [12] often fails to provide sufficient contrast. These observations motivate our work to enhance both representation robustness and discriminative optimization.

III. Multi-View Denoised Graph Collaborative Filtering for Third-Party Library Recommendation

The framework of the proposed multi-view denoised graph collaborative filtering for third-party library recommendation is illustrated in Fig. 1. The framework consists of three core components: *Multi-View Graph Representation Learning with Denoised Interaction Graph Convolution*, *Implicit Supervision for Similarity Graph Representation*, and *Sampled Softmax Loss with Graph Topology-Based Negative Sampling*.

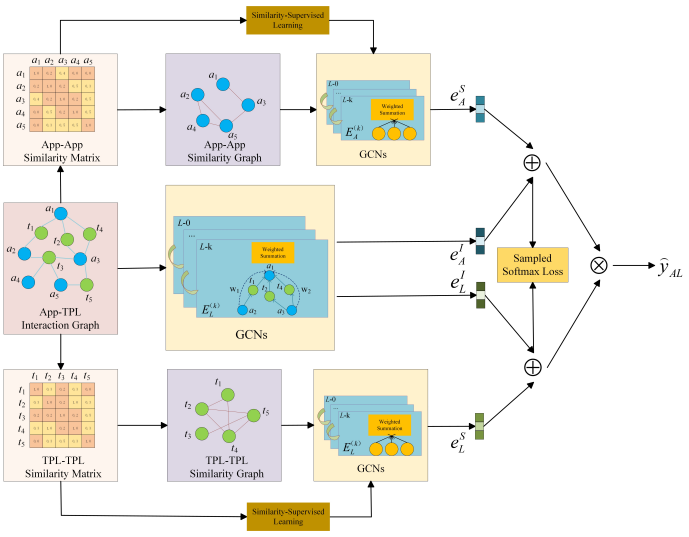

Fig. 1: The overall framework of MDGCF

In the *Multi-View Graph Representation Learning with Denoised Interaction Graph Convolution* module, to alleviate data sparsity and extract complementary features, three distinct views are constructed: App-App similarity graph, TPL-TPL similarity graph and the App-TPL interaction graph. Standard light convolutions are applied to similarity graphs to capture implicit collaborative signals, while a denoised convolution

mechanism is carefully designed for the representation of interaction graph to mitigate noise accumulation. By leveraging normalized weights to perform differentiated neighbor aggregation for even-order nodes, the mechanism suppresses information propagation between irrelevant nodes, ensuring that node representations remain discriminative. The *Implicit Supervision for Similarity Graph Representation* module then utilizes pre-computed similarity matrices as soft supervision signals to align the learned embedding space with the intrinsic collaborative topology, ensuring consistency with underlying topological structures. This process penalizes structural inconsistencies and preserves fine-grained semantic closeness. Finally, the model is optimized via the *Sampled Softmax Loss with Graph Topology-Based Negative Sampling* module. We propose a multi-negative sampling strategy based on graph topology structure, which strengthens the differentiation between invoked and non-invoked TPLs, enabling more discriminative optimization.

A. Multi-View Graph Representation Learning with Denoised Interaction Graph Convolution

To alleviate data sparsity and the noise propagation inherent in multi-layer convolutions, a multi-view denoised graph learning framework is established. Complementary structural information is extracted from three constructed graphs: App-App similarity graph, TPL-TPL similarity graph and the App-TPL interaction graph. Distinct propagation mechanisms are deployed for different views. Specifically, similarity graphs utilize standard light graph convolutions to capture implicit collaborative signals, while the interaction graph adopts a denoised convolution that performs differentiated aggregation. Through differentiated aggregation on even-order nodes, the mechanism suppresses irrelevant propagation to alleviate noise and preserve node discriminability. Ultimately, the representations learned from these diverse views are fused to generate robust node embeddings. This module contains four submodules: *Multi-View Construction*, *Representation Learning on Similarity Graphs*, *Representation Learning on Interaction Graph*, and *Multi-View Embedding Fusion*.

Multi-View Construction. To capture the multifaceted structural dependencies within the TPL ecosystem, historical interactions between m Apps and n TPLs are encoded into a binary matrix $\boldsymbol{R} \in \{0,1\}^{m\times n}$. An entry $R_{ij}=1$ indicates that App A_i utilizes TPL L_j, whereas $R_{ij}=0$ indicates no such usage. Based on $\boldsymbol{R}$, the TPL usage profile for App A_i is represented by the vector $\vec{A_i} \in \{0,1\}^n$, and the App usage profile for TPL L_j corresponds to the vector $\vec{L_j} \in \{0,1\}^m$.

High-degree nodes often dominate similarity measurements due to the long-tail distribution prevalent in TPL data [1]. To mitigate this popularity bias, a popularity-aware weighting scheme is integrated into the representation process. First, the popularity p_{A_i} of an App A_i is determined by its degree as $p_{A_i} = \sum_{j=1}^{n} R_{ij}$, reflecting the total number of integrated TPLs. Similarly, the popularity p_{L_j} of a library L_j is calculated as $p_{L_j} = \sum_{i=1}^{m} R_{ij}$, representing the number of Apps that integrate this library. Logarithmic weights are then assigned to both Apps and TPLs as defined in Eq. (1) and Eq. (2), where $\lambda > 0$ serves as a smoothing factor to stabilize weights for low-degree nodes. These weights are then aggregated into the global weight vectors $\vec{W}_A = [w_{A_1}, ..., w_{A_m}]^T \in \mathbb{R}^m$ and $\vec{W}_L = [w_{L_1}, ..., w_{L_n}]^T \in \mathbb{R}^n$ for Apps and TPLs, respectively.

$$w_{A_i} = \ln(p_{A_i} + \lambda) \tag{1}$$

$$w_{L_j} = \ln(p_{L_j} + \lambda) \tag{2}$$

Based on these weights, a weighted Jaccard similarity metric is employed to quantify the collaborative affinity between nodes. The similarity $Sim_{App}(A_i, A_j)$ between Apps A_i and A_j, and $Sim_{TPL}(L_i, L_j)$ between TPLs L_i and L_j are calculated in Eq. (3) and Eq. (4), where $\odot$ denotes the element-wise product and $\|\cdot\|_1$ represents the L_1 norm.

$$Sim_{App}(A_i, A_j) = \frac{\|\vec{A_i} \odot \vec{A_j} \odot \vec{W}_L\|_1}{\|\vec{A_i} \odot \vec{W}_L\|_1 + \|\vec{A_j} \odot \vec{W}_L\|_1 - \|\vec{A_i} \odot \vec{A_j} \odot \vec{W}_L\|_1} \tag{3}$$

$$Sim_{TPL}(L_i, L_j) = \frac{\|\vec{L_i} \odot \vec{L_j} \odot \vec{W}_A\|_1}{\|\vec{L_i} \odot \vec{W}_A\|_1 + \|\vec{L_j} \odot \vec{W}_A\|_1 - \|\vec{L_i} \odot \vec{L_j} \odot \vec{W}_A\|_1} \tag{4}$$

The App-App similarity graph G_A is constructed by connecting App pairs where $Sim_{App}(A_i, A_j) \geq \alpha_1$. The TPL-TPL similarity graph G_L is constructed by connecting TPL pairs where $Sim_{TPL}(L_i, L_j) \geq \alpha_2$, effectively filtering out weak correlations. Complementing these similarity views, the original interaction matrix $\boldsymbol{R}$ is directly modeled as a bipartite graph G_{AL}, where an edge connects A_i and L_j if $R_{ij}=1$. Subsequently, these three graph views serve as the structural inputs for the proposed model.

Representation Learning on Similarity Graphs. To capture implicit collaborative signals and functional semantics within the similarity views, a lightweight graph convolutional network is applied to the constructed App-App similarity graph G_A and TPL-TPL similarity graph G_L. Learnable embeddings $\boldsymbol{e}_{A_i}^{(0)}$, $\boldsymbol{e}_{L_j}^{(0)} \in \mathbb{R}^d$ are initialized for App A_i and TPL L_j respectively, serving as the 0-th layer embeddings for similarity graph propagation (i.e., $\boldsymbol{e}_{A_i}^{S,(0)} = \boldsymbol{e}_{A_i}^{(0)}, \boldsymbol{e}_{L_j}^{S,(0)} = \boldsymbol{e}_{L_j}^{(0)}$), where d is the embedding dimension. On these similarity graphs, the embedding of a node at the l-th layer is updated by aggregating normalized representations of its neighbors from the $l-1$-th layer. For an App node A_i, the aggregation process is defined in Eq. (5), where $\mathcal{N}_{A_i}$ denotes the neighborhood of A_i. The symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_{A_i}|}\sqrt{|\mathcal{N}_{A_j}|}}$ stabilizes the training process. Similarly, for a TPL node L_i, the propagation rule is defined in Eq. (6).

$$\boldsymbol{e}_{A_i}^{S,(l)} = \sum_{A_j \in \mathcal{N}_{A_i}} \frac{1}{\sqrt{|\mathcal{N}_{A_i}|}\sqrt{|\mathcal{N}_{A_j}|}} \boldsymbol{e}_{A_j}^{S,(l-1)} \tag{5}$$

$$\boldsymbol{e}_{L_i}^{S,(l)} = \sum_{L_j \in \mathcal{N}_{L_i}} \frac{1}{\sqrt{|\mathcal{N}_{L_i}|}\sqrt{|\mathcal{N}_{L_j}|}} \boldsymbol{e}_{L_j}^{S,(l-1)} \tag{6}$$

After L layers of propagation, the final representations for App A_i and TPL L_j are obtained by average pooling over embeddings from all layers, as defined in Eq. (7) and Eq. (8) respectively, where $\boldsymbol{e}_{A_i}^{S}$ and $\boldsymbol{e}_{L_j}^{S}$ represent the App and TPL embeddings from similarity graphs.

$$\boldsymbol{e}_{A_i}^{S} = \frac{1}{L+1} \sum_{l=0}^{L} \boldsymbol{e}_{A_i}^{S,(l)} \tag{7}$$

$$\boldsymbol{e}_{L_j}^{S} = \frac{1}{L+1}\sum_{l=0}^{L}\boldsymbol{e}_{L_j}^{S,(l)} \tag{8}$$

Representation Learning on Interaction Graph. While the similarity graphs capture implicit semantic correlations, the explicit App-TPL interactions in the bipartite graph serve as the primary source of collaborative signals. However, standard multi-layer convolution aggregation tends to blend irrelevant node features, leading to noise accumulation and over-smoothing issues, which reduce the discriminative power of node representations. To address these challenges, we propose a denoised graph convolution on the interaction graph that specifically targets even-order node aggregation while retaining standard light graph aggregation for odd layers.

Specifically, this module leverages normalized weights for differentiated aggregation on even-order nodes to suppress irrelevant propagation and distinguish valid collaborative signals from noise. For a target App A_i and its even-order node A_p, the normalized weight $w(A_i, A_p)$ is computed relative to the global node set $\mathcal{V}_{App}$, as defined in Eq. (9). Similarly, for a target TPL L_j and its even-order node L_q, the weight $w(L_j, L_q)$ is defined in Eq. (10).

$$w(A_i, A_p) = \frac{Sim_{App}(A_i, A_p)}{\sum_{A_k \in \mathcal{V}_{App}} Sim_{App}(A_i, A_k)} \tag{9}$$

$$w(L_j, L_q) = \frac{Sim_{TPL}(L_j, L_q)}{\sum_{L_k \in \mathcal{V}_{TPL}} Sim_{TPL}(L_j, L_k)} \tag{10}$$

The interaction view is initialized with the same learnable embeddings as the similarity views, i.e., $\boldsymbol{e}_{A_i}^{I,(0)} = \boldsymbol{e}_{A_i}^{(0)}$ and $\boldsymbol{e}_{L_j}^{I,(0)} = \boldsymbol{e}_{L_j}^{(0)}$. Leveraging the computed weights, the propagation rule for even layers (e.g., $l = 2, 4, \dots$) employs differentiated aggregation, whereas odd layers follow the standard light graph propagation. For App node A_i, the update rule aggregating features from even-order nodes is defined in Eq. (11), where $\mathcal{N}_{A_i}^{L}$ denotes the set of TPLs interacting with App A_i. Similarly, the update rule for TPL node L_j is defined in Eq. (12).

$$\boldsymbol{e}_{A_i}^{I,(l)} = \sum_{A_p \in \mathcal{V}_{App}} \frac{w(A_i, A_p)}{\sqrt{|\mathcal{N}_{A_i}^{L}| \cdot |\mathcal{N}_{A_p}^{L}|}} \boldsymbol{e}_{A_p}^{I,(l-1)} \tag{11}$$

$$\boldsymbol{e}_{L_j}^{I,(l)} = \sum_{L_q \in \mathcal{V}_{TPL}} \frac{w(L_j, L_q)}{\sqrt{|\mathcal{N}_{L_j}^{A}| \cdot |\mathcal{N}_{L_q}^{A}|}} \boldsymbol{e}_{L_q}^{I,(l-1)} \tag{12}$$

After L layers of propagation, the final representations for App A_i and TPL L_j are obtained by average pooling over embeddings from all layers, as defined in Eq. (13) and Eq. (14) respectively, where $\boldsymbol{e}_{A_i}^{I}$ and $\boldsymbol{e}_{L_j}^{I}$ represent the App and TPL embeddings from the interaction view.

$$\boldsymbol{e}_{A_i}^{I} = \frac{1}{L+1}\sum_{l=0}^{L}\boldsymbol{e}_{A_i}^{I,(l)} \tag{13}$$

$$\boldsymbol{e}_{L_j}^{I} = \frac{1}{L+1}\sum_{l=0}^{L}\boldsymbol{e}_{L_j}^{I,(l)} \tag{14}$$

Multi-View Embedding Fusion. After generating two sets of embeddings from the similarity views and interaction view, these embeddings are fused through averaging to obtain final node representations that encode comprehensive structural information. This fusion strategy ensures balanced contributions from both implicit semantics and explicit interaction patterns. Based on the final node embeddings $\boldsymbol{e}_{A_i}$ and $\boldsymbol{e}_{L_j}$, their inner product $\hat{y}_{A_i L_j}$ is computed as the preference score, which quantifies the likelihood of App A_i adopting TPL L_j. The fusion and prediction processes are defined in Eq. (15), Eq. (16) and Eq. (17), respectively.

$$\boldsymbol{e}_{A_i} = \tfrac{1}{2}(\boldsymbol{e}_{A_i}^{S} + \boldsymbol{e}_{A_i}^{I}) \tag{15}$$

$$\boldsymbol{e}_{L_j} = \tfrac{1}{2}(\boldsymbol{e}_{L_j}^{S} + \boldsymbol{e}_{L_j}^{I}) \tag{16}$$

$$\hat{y}_{A_i L_j} = \boldsymbol{e}_{A_i}^{T}\boldsymbol{e}_{L_j} \tag{17}$$

B. Implicit Supervision for Similarity Graph Representation

Standard LightGCN-based representation learning primarily relies on neighborhood aggregation, which may fail to explicitly align the latent representation space with the underlying graph topology, potentially resulting in suboptimal embeddings. To mitigate this limitation, an implicit supervision mechanism [6] is incorporated, leveraging the Jaccard similarity matrices derived during graph construction as soft supervision signals. The fundamental objective is to align the learned embeddings with intrinsic collaborative structures. For any pair of nodes (e.g., Apps A_i and A_j), if their structural similarity is below threshold α_1, their representations are encouraged to separate in the embedding space. Conversely, node pairs with similarity above the threshold remain unconstrained to preserve their intrinsic closeness. This mechanism ensures that the learned embedding space faithfully reflects the underlying topological structure of the similarity graphs. Formally, for any pair of Apps (A_i, A_j) in the App-App similarity graph and TPLs (L_i, L_j) in the TPL-TPL similarity graph, the element-wise implicit supervision losses are defined in Eq. (18) and Eq. (19), respectively.

$$\mathcal{L}_{ij}^{App} = (\phi(\alpha_1 - Sim_{App}(A_i, A_j)))^{\beta} \cdot [\phi(\cos(\boldsymbol{e}_{A_i}^{S}, \boldsymbol{e}_{A_j}^{S}))]^{\beta+1} \tag{18}$$

$$\mathcal{L}_{ij}^{TPL} = (\phi(\alpha_2 - Sim_{TPL}(L_i, L_j)))^{\beta} \cdot [\phi(\cos(\boldsymbol{e}_{L_i}^{S}, \boldsymbol{e}_{L_j}^{S}))]^{\beta+1} \tag{19}$$

where $\phi(\cdot)$ denotes the ReLU activation function and $\cos(\cdot,\cdot)$ represents the cosine similarity. The hyperparameter β is an integer that controls the nonlinearity strength. The first term $\phi(\alpha - Sim)^{\beta}$ serves as a similarity-aware weight that activates only when the structural similarity is below the threshold α. Specifically, this term assigns zero weight to structurally similar pairs ($Sim \geq \alpha$) while applying positive weights to dissimilar ones ($Sim < \alpha$). The second term $\phi(\cos(\cdot))^{\beta+1}$ penalizes positive embedding similarity between dissimilar nodes. This cumulative loss acts as a soft regularization term, guiding the model to discern fine-grained structural differences. The total implicit supervision loss is defined in Eq. (20) as the sum of the losses from the App view and the TPL view, where $\mathcal{E}_{App}$ and $\mathcal{E}_{TPL}$ denote the sets of all node pairs in the App-App similarity graph and TPL-TPL similarity graph, respectively. Through this mechanism, the embedding space is regularized to maintain consistency with the collaborative similarity patterns, thereby enhancing the discriminative power of the learned representations.

$$\mathcal{L}_{is} = \sum_{(A_i, A_j) \in \mathcal{E}_{App}} \mathcal{L}_{ij}^{App} + \sum_{(L_i, L_j) \in \mathcal{E}_{TPL}} \mathcal{L}_{ij}^{TPL} \tag{20}$$

C. Sampled Softmax Loss with Graph Topology-Based Negative Sampling

Traditional recommendation models predominantly rely on BPR loss to optimize the relative order of positive and negative pairs. However, the BPR loss is limited by its uniform negative sampling strategy, which typically pairs a single negative sample with a positive one. This one-to-one approach often fails to provide sufficient contrast, potentially limiting the model's ability to effectively distinguish between relevant and irrelevant features. To address these limitations, a sampled softmax loss [13] based on multi-negative sample contrast is carefully designed to replace the conventional BPR loss. By contrasting the positive sample against a broader set of negatives, this objective strengthens the differentiation between invoked and non-invoked TPLs, enabling more discriminative optimization and enhancing recommendation performance.

To further ensure the quality of negative samples, a graph topology-based negative sampling strategy is carefully designed. This strategy reduces false negatives by leveraging the structural information of the interaction graph to exclude structurally close nodes that are likely to be latent positive interactions. Specifically, for an App A_i, a valid candidate negative pool $\mathcal{C}(A_i)$ is constructed by excluding TPLs within the h-hop neighborhood from the global TPL set $\mathcal{L}$. This filtering mechanism is defined in Eq. (21), where $d_G(A_i, L_k)$ denotes the shortest path distance on the interaction graph. If this strict filtering results in an empty set, the condition is relaxed to exclude only direct interactions (i.e., $h = 1$).

$$\mathcal{C}(A_i) = \begin{cases} \mathcal{L} \setminus \{L_k : d_G(A_i, L_k) \leq h\}, & \text{if } \mathcal{C} \neq \emptyset \\ \mathcal{L} \setminus \mathcal{N}^L_{A_i}, & \text{otherwise} \end{cases} \tag{21}$$

An efficient in-batch sampling strategy is designed to generate negative samples. The set of all unique TPLs appearing in the current training batch $\mathcal{B}$ is denoted as $\mathcal{L}_\mathcal{B}$. Subsequently, the specific negative set $\mathcal{S}^-_{A_i}$ for App A_i is constructed by retaining only the TPLs from $\mathcal{L}_\mathcal{B}$ that are also valid candidates in the pool $\mathcal{C}(A_i)$. Formally, this intersection process is defined in Eq. (22).

$$\mathcal{S}^-_{A_i} = \mathcal{L}_\mathcal{B} \cap \mathcal{C}(A_i) \tag{22}$$

The objective of MDGCF is to maximize the probability of the ground-truth TPL L_j among the candidate set comprising the positive sample and the dynamically selected negatives. The graph topology-based SSM loss is defined in Eq. (23), where $\mathcal{D}_{batch}$ denotes the set of positive interaction pairs in the current batch, $\hat{y}$ denotes the predicted preference score, and τ is a temperature hyperparameter that controls the sharpness of the probability distribution.

$$\mathcal{L}_{SSM} = -\sum_{(A_i,L_j)\in\mathcal{D}_{batch}} \log \frac{\exp(\hat{y}_{A_iL_j}/\tau)}{\exp(\hat{y}_{A_iL_j}/\tau)+\sum_{L_k\in\mathcal{S}^-_{A_i}}\exp(\hat{y}_{A_iL_k}/\tau)} \tag{23}$$

D. Multi-Task Joint Training

To optimize recommendation performance, a multi-task joint training strategy is adopted. The sampled softmax loss $\mathcal{L}_{SSM}$ serves as the primary objective function for discriminative preference learning. Simultaneously, the implicit supervision loss $\mathcal{L}_{is}$ serves as an auxiliary geometric regularizer to preserve the intrinsic topological structure. Additionally, an L_2 regularization term $\|\Theta\|_2^2$ is included to prevent overfitting, where Θ denotes the model parameters. The overall objective function is defined in Eq. (24), where hyperparameters λ_1 and λ_2 control the contributions of implicit supervision loss and regularization term, respectively.

$$\mathcal{L}_{total} = \mathcal{L}_{SSM} + \lambda_1\mathcal{L}_{is} + \lambda_2\|\Theta\|_2^2 \tag{24}$$

IV. EXPERIMENTS

To evaluate the effectiveness of the proposed approach, extensive experiments are carried out on a real-world dataset. This section first outlines the experimental setup, followed by a performance comparison, and finally presents the ablation studies and hyperparameter analysis.

A. Experimental Setup

Dataset. The experiments are conducted on the MaLib dataset[1], a widely adopted public benchmark for TPL recommendation. To ensure a high-quality and unbiased evaluation, the raw data is subjected to the following three preprocessing procedures. 1) Apps possessing fewer than five TPL interactions are discarded to ensure adequate historical data for capturing collaborative signals; 2) Clusters of functionally similar Apps are identified and eliminated to alleviate potential popularity bias towards specific libraries, encouraging the model to learn more generalizable relationships; 3) The refined interaction data is randomly split into training and testing sets with a ratio of 8:2.

Evaluation Metrics. To comprehensively assess the Top-K recommendation quality, three standard metrics are selected, i.e., $Recall@K$, $NDCG@K$, and $MAP@K$. For all metrics, higher values indicate better performance. Given the relatively small scale of the TPL repository in the test set, K is set to $\{3, 5, 10\}$ for detailed analysis.

Hyperparameter Settings. Baseline configurations are adopted from their respective original papers, with hyperparameters fine-tuned through grid search. For the proposed method, d=128, L=3, batch size=4096, α_1=0.55, α_2=0.5, β=3, λ_1=1e-4, λ_2=1e-3, τ=0.3 and learning rate=1e-4. Additionally, the hop distance threshold h is set to 1, as $h > 1$ leads to a scarcity of candidate negative samples in this dataset. The Adam optimizer is employed for model training.

Baseline Methods. To assess the effectiveness of MDGCF, comparison experiments are conducted with several state-of-the-art baseline approaches, i.e., **LightGCN** [19], **LightGCL** [20], **SimGCL** [9], **SGL** [7], **NCL** [8], **HCF** [5], and **ISGCF** [6]. Details of these baseline methods are presented below.

- **LightGCN** [19]. LightGCN simplifies standard GCNs by removing feature transformation and nonlinear activation. It relies solely on neighborhood aggregation to achieve high efficiency in collaborative filtering tasks.

[1]https://github.com/malibdata/MALib-Dataset/

- **LightGCL** [20]. LightGCL utilizes singular value decomposition for graph augmentation in contrastive learning. This approach preserves global collaborative signals to effectively tackle data sparsity and noise.
- **SimGCL** [9]. SimGCL generates contrastive views by injecting uniform noise into embeddings instead of altering graph structures. This approach enables robust representation learning without complex augmentations.
- **SGL** [7]. SGL augments GCN-based recommendation with self-supervised learning tasks. It generates multiple views via operators like node dropout and random walks to improve recommendation accuracy and robustness.
- **NCL** [8]. NCL aligns node embeddings with their structural and semantic neighbors through contrastive learning. This method integrates neighborhood contexts to enrich the semantic representation of nodes.
- **HCF** [5]. HCF constructs hypergraphs from App-TPL interaction records to model complex dependencies. It applies hypergraph convolution to capture high-order neighborhood information for both Apps and TPLs.
- **ISGCF** [6]. ISGCF utilizes auxiliary similarity graphs to provide implicit supervision for geometric regularization. It employs a popularity-debiased mechanism to construct denoised graphs, mitigating noise and over-smoothing.

B. Performance Evaluation

Table I presents the performance comparison between MDGCF and seven baseline methods in terms of Recall@K, NDCG@K, and MAP@K. Overall, MDGCF consistently achieves the best results across all metrics with Recall@3 of 0.6039, NDCG@3 of 0.8868, and MAP@3 of 0.8663. Specifically, it outperforms the second-best baseline (ISGCF) by 4.21%, 1.71%, and 2.06%, respectively. These consistent improvements across various K values (i.e., 3, 5, 10) validate the effectiveness of the proposed approach.

Examining the baselines reveals inherent constraints that limit their performance. GCN-based models (e.g., LightGCN and LightGCL) depend exclusively on the explicit App-TPL interaction graph. Consequently, they achieve modest Recall@3 scores of 0.5137 and 0.4646 respectively, highlighting their difficulty in learning robust representations under data sparsity. In contrast, contrastive learning approaches (e.g., SimGCL, SGL and NCL) utilize self-supervised signals and generally outperform standard GCN models. However, their reliance on random negative sampling often introduces false negatives, distorting the learned embedding space. HCF achieves notable performance by leveraging hypergraph structures but lacks the guidance of explicit similarity constraints. Among the baselines, ISGCF performs the second-best by incorporating implicit similarity supervision. However, ISGCF only operates message passing on similarity graphs, failing to leverage the signals inherent in explicit interaction patterns.

In contrast, MDGCF addresses these limitations through three synergistic mechanisms. First, multi-view graph learning augmented by implicit supervision effectively mitigates data sparsity. By regularizing the representation space with geometric constraints derived from similarity views, the model recovers latent collaborative signals that are often missed by single-view methods. Second, the proposed denoised GCN mitigates the noise accumulation caused by multi-layer aggregation. By leveraging normalized weights for differentiated aggregation on even-order nodes, the mechanism suppresses information propagation between irrelevant nodes, effectively alleviating the blending of irrelevant features and preserving node discriminability. Third, the sampled softmax loss replaces the conventional BPR loss, strengthening the differentiation between invoked and non-invoked TPLs via multi-negative contrast, which enables more discriminative optimization.

As shown in Table II, replacing the sampled softmax loss with the conventional BPR loss (MDGCF-BPR) leads to inferior recommendation performance, confirming that the multi-negative contrast in SSM loss provides stronger discriminative signals than the pairwise BPR loss. A notable exception is that MDGCF-BPR achieves a marginally higher Recall@10 than MDGCF. This indicates that while the BPR objective is effective for broad retrieval, it lacks the precision of SSM. The stricter contrast in SSM prioritizes pertinent TPLs at top positions, resulting in higher NDCG and MAP scores. Similarly, replacing the denoising GCN with standard LightGCN (MDGCF-LightGCN) leads to a consistent performance drop, validating the necessity of the denoised aggregation strategy. Standard multi-layer convolution often blends irrelevant features, whereas the denoised mechanism maintains discriminability by suppressing noise propagation between irrelevant nodes during even-order aggregation. The synergy of the denoised convolution and sampled softmax loss enables MDGCF to achieve superior overall performance.

TABLE I: Performance comparison of different methods on TPL recommendation

Method	Recall@K			NDCG@K			MAP@K		
	K=3	K=5	K=10	K=3	K=5	K=10	K=3	K=5	K=10
LightGCN	0.5137	0.6337	0.7478	0.6442	0.6497	0.6957	0.7886	0.7704	0.7326
LightGCL	0.4646	0.5871	0.7139	0.5804	0.5929	0.6457	0.7309	0.7168	0.6789
SimGCL	0.4875	0.6130	0.7389	0.7915	0.8014	0.7958	0.7595	0.7439	0.7041
SGL	0.5075	0.6332	0.7545	0.8117	0.8175	0.8108	0.7792	0.7612	0.7222
NCL	0.4974	0.6259	0.7487	0.7950	0.8040	0.7991	0.7612	0.7457	0.7091
HCF	0.5757	0.7052	0.8120	0.8670	0.8681	0.8601	0.8429	0.8243	0.7893
ISGCF	0.5795	0.7148	0.8193	0.8719	0.8734	0.8646	0.8488	0.8299	0.7949
MDGCF	**0.6039**	**0.7327**	**0.8203**	**0.8868**	**0.8865**	**0.8790**	**0.8663**	**0.8487**	**0.8182**
Improv. (%)	4.21%	2.50%	0.12%	1.71%	1.50%	1.67%	2.06%	2.27%	2.93%

TABLE II: Performance comparison with model variants

Variant	Recall@K			NDCG@K			MAP@K		
	K=3	K=5	K=10	K=3	K=5	K=10	K=3	K=5	K=10
MDGCF-BPR	0.5934	0.7252	**0.8223**	0.8766	0.8777	0.8698	0.8535	0.8359	0.8033
MDGCF-LightGCN	0.5975	0.7251	0.8155	0.8802	0.8808	0.8737	0.8590	0.8419	0.8111
MDGCF	**0.6039**	**0.7327**	0.8203	**0.8868**	**0.8865**	**0.8790**	**0.8663**	**0.8487**	**0.8182**

C. Ablation Analysis

An ablation study is conducted to verify the necessity of each graph view and the implicit supervision module. Fig. 2 compares MDGCF against three variants: (1) *w/o Sim Graph*, which removes the App-App similarity graph and TPL-TPL similarity graph to rely solely on the App-TPL interaction graph; (2) *w/o Inter Graph*, which removes the App-TPL

interaction graph to rely solely on similarity graphs; and (3) *w/o IS*, which removes the implicit supervision module.

First, removing similarity graphs (*w/o Sim Graph*) causes a sharp performance drop, with MAP@10 plummeting by 11.66%. This confirms their foundational role in connecting semantically analogous nodes and facilitating robust collaborative filtering, particularly for long-tail Apps with sparse interactions. Second, removing the interaction graph (*w/o Inter Graph*) maintains stable Recall@10 but decreases ranking metrics (MAP@10 drops by 2.58%). This indicates that while similarity graphs retrieve broad candidates, the explicit interaction graph provides the fine-grained preference signals to distinguish genuine library adoptions from functionally similar ones. Third, removing implicit supervision (*w/o IS*) leads to a consistent decline across all metrics, with MAP@10 dropping by 1.38%. Without this geometric regularizer, the model fails to explicitly distance structurally dissimilar nodes, resulting in less discriminative embeddings. Thus, fusing multiple graph views and incorporating implicit supervision are essential to balance broad coverage with high ranking precision.

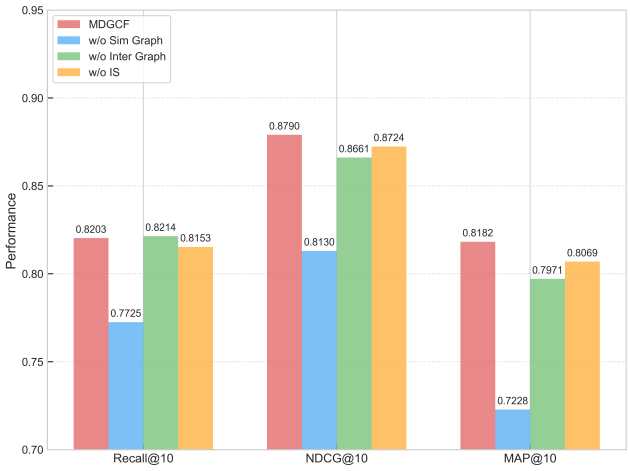

Fig. 2: The ablation study on MDGCF

D. Impact of Hyperparameters

Based on the optimal configuration, the impacts of six core hyperparameters on performance are investigated, i.e., embedding dimension d, number of GCN layers L, implicit supervision weight λ_1, temperature coefficient τ and similarity thresholds α_1 and α_2.

Impact of Model Configuration Parameters. As shown in Fig. 3 (a), the embedding dimension d determines the model's capacity to encode latent semantics. When d increases from 32 to 128, performance improves substantially, as higher dimensions capture more complex interaction patterns. However, the embedding size is set to $d = 128$ as a trade-off between performance and efficiency, since the marginal gain at $d = 256$ is observed to be disproportionate to the significant increase in computational overhead. As shown in Fig. 3 (b), the number of GCN layers L controls the neighborhood aggregation depth. Metrics improve as L increases to its optimum at 3, confirming the value of high-order collaborative signals. However, at $L = 4$, performance plateaus or drops due to the over-smoothing issue, where node representations become indistinguishable.

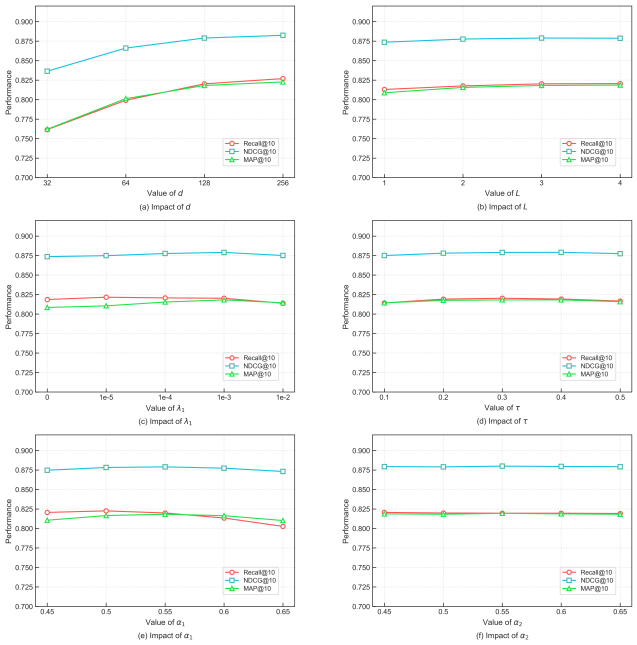

Fig. 3: The impact of hyperparameters

Impact of Training and Loss Parameters. As shown in Fig. 3 (c), the implicit supervision weight λ_1 reaches its optimum at 1e-3, confirming that an appropriate level of auxiliary supervision enhances representation robustness, while excessive weights ($1e-2$) lead to over-regularization. As shown in Fig. 3 (d), the temperature coefficient τ regulates the sharpness of the softmax distribution. The optimal range is $[0.3, 0.4]$. Lower τ values lead to an excessively sharp distribution, which potentially causes training instability due to extreme gradients. Conversely, higher τ values result in a smoother distribution, which dilutes the contrast between positive and negative samples, thereby reducing the model's discriminative power.

Impact of Graph Construction Parameters. As shown in Fig. 3 (e), the model is sensitive to the App similarity threshold α_1, with an optimum at $[0.5, 0.55]$. Specifically, suboptimal performance is observed at α_1 = 0.45, indicating that during propagation, effective signals are diluted by numerous low-quality edges introduced by the low threshold. Conversely, valuable neighborhood connections are removed by higher thresholds (≥ 0.6) as the graph is rendered overly sparse. In contrast, as shown in Fig. 3 (f), the model is highly robust to the TPL similarity threshold α_2 within [0.45, 0.65], primarily due to the stable Jaccard similarity estimations provided by dense TPL interactions. A performance trade-off is observed between recall at lower values and the highest NDCG and MAP scores at $\alpha_2 = 0.55$.

V. CONCLUSION

In this work, we propose MDGCF, a novel multi-view denoised graph collaborative filtering method for TPL recommendation. To alleviate data sparsity, MDGCF constructs

complementary graph views and leverages implicit supervision to capture latent collaborative signals. To mitigate noise, a denoised interaction graph convolution mechanism is proposed by performing differentiated aggregation to suppress irrelevant propagation. Additionally, a sampled softmax loss with graph topology-based negative sampling is carefully designed to replace the conventional BPR loss, utilizing multi-negative contrast to strengthen the differentiation between invoked and non-invoked TPLs. Extensive experiments on a real-world dataset demonstrate that MDGCF outperforms state-of-the-art baselines, validating its effectiveness. Future work will explore the integration of auxiliary information and more scalable computation to further enhance recommendation performance in evolving software ecosystems.

ACKNOWLEDGMENT

This work is partially supported by the Hunan Provincial Natural Science Foundation of China under Grant No: 2026JJ50519. *Guosheng Kang* is the corresponding author.

REFERENCES

[1] Q. He, B. Li, F. Chen, J. Grundy, X. Xia, and Y. Yang, "Diversified third-party library prediction for mobile App development," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 150–165, 2020.

[2] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of App store analysis for software engineering," *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 817–847, 2016.

[3] H. Yu, X. Xia, X. Zhao, and W. Qiu, "Combining collaborative filtering and topic modeling for more accurate android mobile App library recommendation," in *Asia-Pacific Symposium on Internetware*. ACM, 2017, pp. 1–6.

[4] B. Li, Q. He, F. Chen, X. Xia, L. Li, J. Grundy, and Y. Yang, "Embedding App-library graph for neural third party library recommendation," in *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2021, pp. 466–477.

[5] L. Chen, N. Mei, Y. He, W. Liu, G. Zhong, and M. Tang, "High-order collaborative filtering for third-party library recommendation," in *IEEE International Conference on Web Services*. IEEE, 2023, pp. 483–492.

[6] L. Chen, M. Tang, N. Mei, F. Xie, G. Zhong, and Q. He, "Implicit supervision-assisted graph collaborative filtering for third-party library recommendation," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1459–1471, 2025.

[7] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2021, pp. 726–735.

[8] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *ACM Web Conference*. ACM, 2022, pp. 2320–2329.

[9] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2022, pp. 1294–1303.

[10] J. Li, X. Qiu, G. Kang, Y. Li, and J. Liu, "TSSGCF: Textual similarity-supervised graph collaborative filtering for Web API recommendation," in *International Conference on Service-Oriented Computing*. Springer, 2025, pp. 102–117.

[11] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 3438–3445.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[13] J. Wu, X. Wang, X. Gao, J. Chen, H. Fu, and T. Qiu, "On the effectiveness of sampled softmax loss for item recommendation," *ACM Transactions on Information Systems*, vol. 42, no. 4, pp. 1–26, 2024.

[14] F. Thung, D. Lo, and J. Lawall, "Automated library recommendation," in *20th Working Conference on Reverse Engineering*, 2013, pp. 182–191.

[15] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "Crossrec: Supporting software developers by recommending third-party libraries," *Journal of Systems and Software*, vol. 161, no. 2020, pp. 110 460–110 474, 2020.

[16] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2019, pp. 165–174.

[17] Y. Jin, Y. Zhang, and Y. Zhang, "Neighbor library-aware graph neural network for third party library recommendation," *Tsinghua Science and Technology*, vol. 28, no. 4, pp. 769–785, 2023.

[18] B. Li, H. Quan, J. Wang, P. Liu, H. Cai, Y. Miao, Y. Yang, and L. Li, "Neural library recommendation by embedding project-library knowledge graph," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1620–1638, 2024.

[19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[20] X. Cai, C. Huang, L. Xia, and X. Ren, "LightGCL: Simple yet effective graph contrastive learning for recommendation," in *The Eleventh International Conference on Learning Representations*, 2023, pp. 1–15.