Hypergraph Contrastive Learning with Graph Structure Learning for Recommendation

Yuma Dose
Osaka University
Osaka, Japan
douse.yuma@ist.osaka-u.ac.jp

Shuichiro Haruta
Human-Centered AI Laboratories
KDDI Research, Inc.
Fujimino, Japan
sh-haruta@kddi.com

Yihong Zhang
Osaka University
Osaka, Japan
zhang.yihong@ist.osaka-u.ac.jp

Takahiro Hara
Osaka University
Osaka, Japan
hara@ist.osaka-u.ac.jp

2024 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-7488-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICMLA61862.2024.00062

***Abstract*—In this paper, to address the insufficient capturing of high-order correlations and the vulnerability to noise in conventional collaborative filtering models, we propose HyperGraph Contrastive Learning with graph structure learning for recommendation (HGCL). HGCL employs user and item hypergraphs as contrastive views in contrastive learning, where edges connect the item and user's k-order reachable neighbors. This approach allows HGCL to model relationships with distant nodes and explicitly capture high-order correlations, alleviating the issue of over-smoothing. Subsequently, HGCL performs contrastive learning between representations obtained from the user-item interaction graph and hypergraphs. This integrates the user-item relationship features from the interaction graph with the high-order correlations for each user and item from the hypergraphs, resulting in more effective representations. Furthermore, to address the weakness of hypergraphs against noise, we modify the hypergraph structure learning method for the recommendation task and incorporate it into HGCL. Based on the user and item representations, HGCL detects potential node relationships and noise in the hypergraph for the recommendation task. By adding or removing these nodes from the hypergraph, HGCL acquires a denoised hypergraph. By applying these processes to user and item hypergraphs, HGCL obtains improved hypergraphs with reduced noise effects and achieves more effective recommendations. Experimental results on real-world datasets demonstrate that HGCL outperforms baseline models, achieving up to 5.25% improvement in NDCG@20.**

***Index Terms*—recommender system, collaborative filtering, hypergraph, contrastive learning**

I. Introduction

Recommender systems now serve as essential utilities, offering personalized item suggestions to users based on their learned preferences [1]. Collaborative Filtering (CF) is one of the most widely utilized architectures for recommendation and has drawn significant research attention for over two decades [2]. The fundamental principle behind the CF paradigm is to project users and items into low-dimensional latent embedding representations based on observed interactions between users and items [3]. Until now, a variety of CF techniques have been suggested, including matrix factorization [4], autoencoder [5], and attention mechanism [6]. Recently, with the success of Graph Neural Networks (GNN) [7], [8], several graph-based CF models have been proposed [9]–[11]. These models utilize a user-item interaction graph, where edges represent users' preferences like purchase or search histories, effectively modeling the interaction relationships between users and items. Then they use GNN to obtain representations for each user and item by aggregating characteristics from their local neighborhoods.

Despite their effectiveness, graph-based CF models continue to face challenges for the sparsity and noise of interaction data. To address this issue, recent research has incorporated contrastive learning (CL) as the auxiliary task to learn more robust representations [12], [13]. These CL-based models construct contrastive views through data augmentations and then provide self-supervised signals by maximizing the mutual information between these contrastive views [14]. In recent years, various data augmentation techniques for constructing contrastive views have been investigated as a key to improving accuracy [15]–[18].

While the aforementioned CL-based models have achieved state-of-the-art results, their performance remains limited due to insufficient modeling of high-order correlations among users. Capturing features of both near neighbors and more distant nodes is essential for enhancing recommendation performance. Nonetheless, graph CL-based models cannot effectively capture high-order features of distant nodes because of the over-smoothing problem [19], [20]. Although models such as NCL [17] and HCCF [18] partially consider high-order correlations with their own data augmentations, they utilize high-order information only implicitly and fail to leverage the full potential of high-order correlations [21]. On the other hand, hypergraph is known as an effective way to explicitly capture high-order correlations [22]–[24]. The hypergraph flexibly edits the graph structure of the interaction graph and directly connects k-order reachable nodes at the edges, enabling the capture of high-order information [23]. However, hypergraphs are vulnerable to noise. When noisy interactions are included, subsequent nodes that are not originally connected become k-order reachable and are connected by edges. This tends to make hypergraphs noisy, and the direct convolution of these noisy node features has a significant impact, causing the node representations to deteriorate.

In this paper, we explore the potential of contrastive learning using the hypergraph to further improve recommendation accuracy. Specifically, to address the insufficient capturing

1946-0759/24/$31.00 ©2024 IEEE
DOI 10.1109/ICMLA61862.2024.00062

of high-order correlations and the vulnerability to noise in conventional models, we propose **H**yper**G**raph **C**ontrastive **L**earning with graph structure learning (**HGCL**). HGCL employs user and item hypergraphs as contrastive views in contrastive learning, where edges connect the item and user's k-order reachable neighbors. This approach allows HGCL to model relationships with distant nodes and explicitly capture high-order correlations, alleviating the issue of over-smoothing. Subsequently, HGCL performs contrastive learning between representations obtained from the interaction graph and hypergraph. This integrates the user-item relationship features from the interaction graph with the high-order correlations for each user and item from the hypergraphs, resulting in more effective representations. Furthermore, to address the weakness of hypergraphs against noise, we modify the hypergraph structure learning method [25] for the recommendation task and incorporate it into HGCL. Based on the user and item representations, HGCL detects potential node relationships and noise in the hypergraph for the recommendation task. By adding or removing these nodes from the hypergraph, HGCL acquires a denoised hypergraph. By applying these processes to user and item hypergraphs, HGCL obtains improved hypergraphs with reduced noise effects. Experiments on three real-world datasets demonstrate that HGCL achieves an improvement in 5.25% improvement in NDCG@20 compared to the state-of-the-art baseline model in the Yelp dataset.

We summarize the contributions of this work as follows:

- We propose HyperGraph Contrastive Learning with graph structure learning for recommendation, named HGCL. HGCL performs contrastive learning between representations obtained from the interaction graph and the hypergraph, integrating user-item relationship features with high-order correlations from the hypergraph.
- To the best of our knowledge, this work is the first to incorporate hypergraph structure learning techniques into the recommender system. To address the weakness of hypergraphs against noise in interaction data, we design a hypergraph structure learning architecture that acquires improved hypergraphs with reduced noise effects.
- Extensive experimental results on three real-world datasets reveal that HGCL significantly outperforms the baseline methods. This result confirms the effectiveness of contrastive learning using hypergraph and hypergraph structure learning.

The remainder of this paper is organized as follows. In Section Ⅱ, we describe the definition of some important notations. The proposed framework HGCL is presented in Section Ⅲ. The results of evaluation experiments are reported in Section Ⅳ. We introduce related work in Section V. Finally, Section VI concludes this paper.

II. PRELIMINARIES

A. Definition of interaction graph

Graph-based collaborative filtering models utilize a user-item interaction graph, where edges represent users' preferences. The interaction graph is a bipartite graph with the set of users U and the set of items I as nodes. Edges $\mathcal{E}$ are the observed interactions between U and I. Formally, the interaction graph $\mathcal{G}$ is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the node set $\mathcal{V} = \{U \cup I\}$. The fundamental principle of graph-based collaborative filtering models is to employ a neighborhood aggregation approach on $\mathcal{G}$. The representation of each node is updated by aggregating the representations of its neighboring nodes.

B. Definition of hypergraph

In a hypergraph, a hyperedge connects two or more nodes [26]. A hypergraph is usually defined as $\mathcal{H} = \{\mathcal{V}, \mathcal{E}_H\}$, where $\mathcal{V}$ represents the node set, and $\mathcal{E}_H$ denotes the hyperedge set. An adjacency matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}_H|}$ is used to represent the connections among nodes on the hypergraph, where $\mathbf{H}_{ve}$ = 1 indicates node v belongs to hyperedge e. In summary, compared to interaction graphs, hypergraphs naturally handle high-order correlations.

Given a hypergraph constructed from datasets, the adjacency matrix $\mathbf{H}$ and node feature $\mathbf{X}^{(l)}$ are fed into HyperGraph Neural Networks (HGNN) [23], [27], defined as

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1/2}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\Theta^{(l)} + \mathbf{X}^{(l)}\right), \tag{1}$$

where the $\Theta^{(l)} \in \mathbb{R}^{C^{(l)}\times C^{(l+1)}}$ is a trainable parameter and $C^{(l)}/C^{(l+1)}$ is the input/output node feature dimension at layer l. Two diagonal matrices $\mathbf{D}_e \in \mathbb{N}^{|\mathcal{E}_H|\times|\mathcal{E}_H|}$ and $\mathbf{D}_v \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ represent hyperedge degrees the node degrees respectively. $\sigma(\cdot)$ denotes an arbitrary nonlinear activation function (e.g., ReLU (·)). This passage describes a two-stage refinement process for performing node-hyperedge-node feature transformations on hypergraph structures. Initially, features of nodes connected by the hyperedge are aggregated using message passing guided by $\mathbf{H}^T$, forming the hyperedge representations. Subsequently, refined node representations are generated by aggregating related hyperedge representations through $\mathbf{H}$. Finally, trainable parameters Θ and a nonlinear activation function $\sigma(\cdot)$ are applied. Hypergraph convolutions, as effective and deep operations, facilitate high-level information interaction among nodes by leveraging the node-hyperedge-node transform. After *L*-th layer propagation, the final node representation is calculated as

$$\mathbf{X} = \mathbf{X}^{(0)}||\mathbf{X}^{(1)}||\mathbf{X}^{(2)}||\ldots||\mathbf{X}^{(L)}, \tag{2}$$

where $\mathbf{X}^{(0)}$ is initial embedding and $||$ is concatenation operation.

III. METHODOLOGY

In this paper, we propose HyperGraph Contrastive Learning with graph structure learning, named HGCL. Fig. 1 provides the overview of HGCL. HGCL consists of three steps. First, user and item hypergraphs $\mathcal{H}_u$ and $\mathcal{H}_i$ are created based on the interaction between users and items. Next, using hypergraph structure learning, potential nodes are added to the hyperedges, and unrelated nodes that could introduce noise are removed from the hyperedges. This results in improved hypergraph

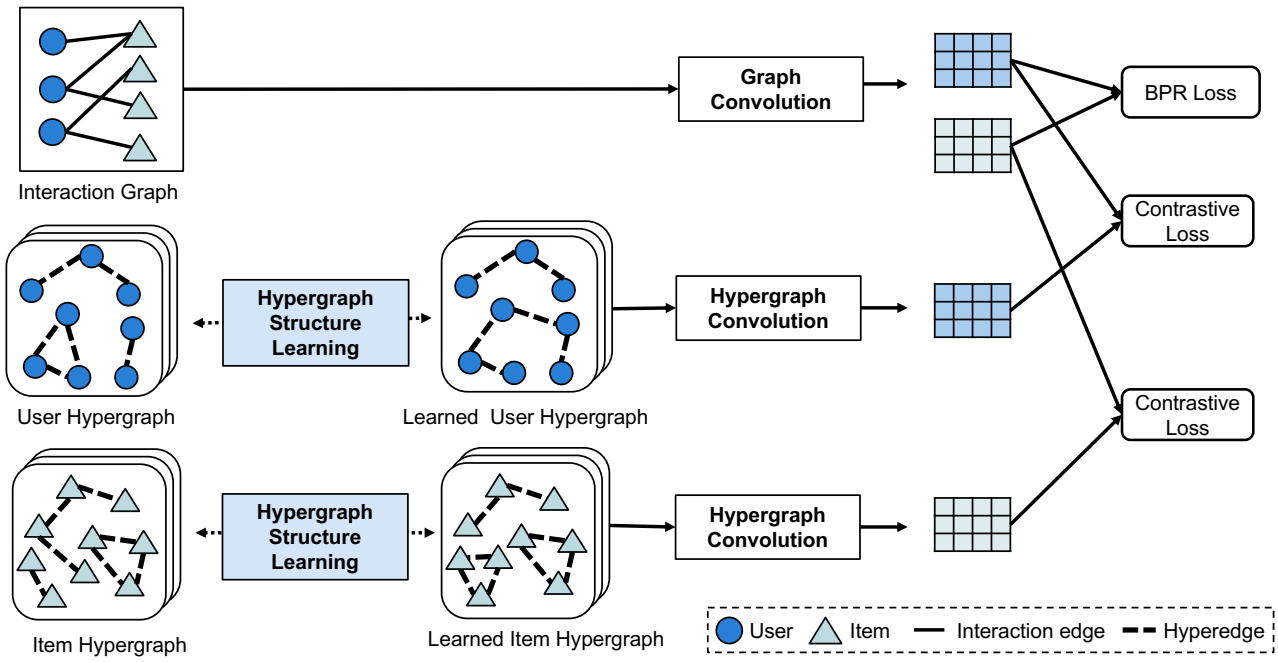

Fig. 1: The overall architecture of HGCL. By utilizing hypergraph structure learning and contrastive learning, we develop a recommendation model that effectively captures high-order correlations and is robust to noise.

structures $\mathcal{H'}_u$ and $\mathcal{H'}_i$. Finally, utilizing the representations obtained from $\mathcal{G}$, $\mathcal{H'}_u$, and $\mathcal{H'}_i$, the model is trained through multi-task training, which simultaneously optimizes the recommendation task, contrastive learning, and hypergraph structure learning. Details of the specific operations in each step are introduced in the following sections.

A. *Construction of hypergraphs*

According to [23], HGCL constructs a hypergraph for each user and item ($\mathcal{H}_u$, $\mathcal{H}_i$), where edges connect k-order reachable nodes of items and users.

*1) **User hypergraph** ($\mathcal{H}_u$):* First, we introduce the definition of item's k-order reachable users as follows.

Definition 1 (Item's k-order Reachable Users) *In a user-item bipartite graph, U_j is k-order reachable from I_i if at least one direct path between U_j and I_i involves k or fewer users.*

An example of user hypergraph construction is shown in Fig. 2. The basic operations for constructing a user hypergraph can be summarized in the following three steps.

(1) Extract the k-order reachable user sets of items. For item I_i, its k-order reachable user set is denoted as $J_U^k(I_i)$.

(2) Combine the k-order reachable user sets of all items to generate the k-order reachable user hypergroup $\mathbf{H}_U^k = \{J_U^k(I_i)|I_i \in I\}$. The matrix representation for constructing H_U^k is defined as

$$\mathbf{H}_U^k = R \cdot \min(1, \text{pow}(R^T \cdot R, k-1)), \tag{3}$$

where R is the user-item rating matrix, $\text{pow}(R,k)$ is a function that computes the k-th power of matrix R, $\min(x,R)$ is a function that replaces all elements in matrix R with values greater than x, and $(\cdot)$ denotes matrix multiplication.

(3) Aggregate all k-order reachable user hypergroups for $k = 1, 2, ..., k_{hyper}$ to generate the user hypergraph. k_{hyper} indicates the furthest reachable node we consider. In this work, a simple concatenation operator $||$ is used to aggregate all hypergroups. Formally, it is represented as

$$\mathbf{H}_U = \mathbf{H}_U^1 || \mathbf{H}_U^2 || \dots || \mathbf{H}_U^{k_{hyper}}. \tag{4}$$

*2) **Item hypergraph** ($\mathcal{H}_i$):* $\mathcal{H}_i$ is constructed using similar operations as $\mathcal{H}_u$. First, the definition of item's k-order reachable user is as follows.

Definition 2 (User's k-order reachable Items) *In a user-item bipartite graph, I_i is k-order reachable from U_j if at least one direct path between U_j and I_i involves k or fewer users.*

Next, we combine the k-order reachable item sets of all users to generate the k-order reachable item hypergroup $\mathbf{H}_I^k$. Formally, $\mathbf{H}_I^k$ is formulated as

$$\mathbf{H}_I^k = R^T \cdot \min(1, \text{pow}(R \cdot R^T, k-1)). \tag{5}$$

Finally, aggregate all k-order reachable user hypergroups for $k = 1, 2, ..., k_{hyper}$, and the item hypergraph is constructed as

$$\mathbf{H}_I = \mathbf{H}_I^1 || \mathbf{H}_I^2 || \dots || \mathbf{H}_I^{k_{hyper}}. \tag{6}$$

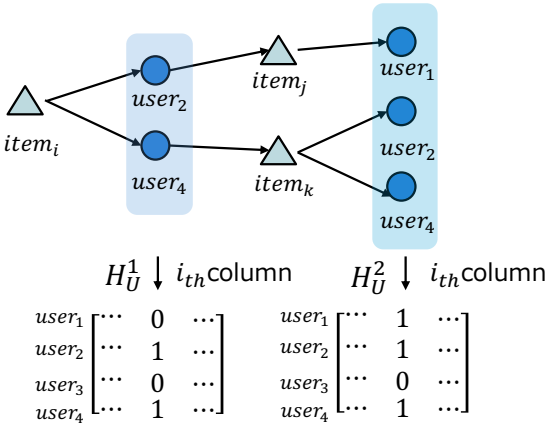

Fig. 2: An example of constructing a user hypergraph. This figure illustrates one example of k-order reachable user hypergroups when k = 1 and 2, respectively.

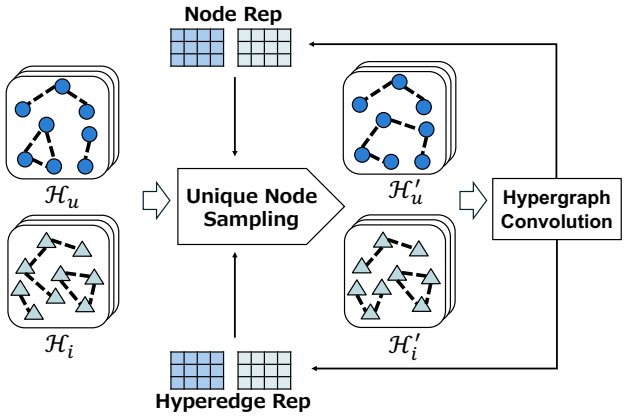

Fig. 3: An overview of hypergraph structure learning. Using node and hyperedge representations, potential nodes are added to hyperedges, and nodes that might introduce noise are removed.

B. *Hypergraph Structure Learning*

In this paper, we develop hypergraph structure learning to strengthen the hypergraph's robustness against noise. By sampling unique nodes, such as potential nodes and noisy nodes, and adding them into or removing them from the hyperedges of $\mathcal{H}$, we obtain an improved hypergraph $\mathcal{H}'$.

An overview of hypergraph structure learning is shown in Fig. 3. First, based on the cosine similarity between node representations and hyperedge representations, the highly relevant node-hyperedge pairs are detected. This allows the extraction of potential relationships between nodes and hyperedge. Formally, the highly relevant node-hyperedge pairs $\Delta\mathbf{H}$ is expressed as

$$\Delta\mathbf{H}_{ij} = \begin{cases} 1, & \text{if } \mathbf{H}_{ij} = 0 \text{ and } S_{ij} \in \text{top}(S, p_{\text{add}}), \\ 0, & \text{if } \mathbf{H}_{ij} = 1, \end{cases} \tag{7}$$

where the similarity matrix between nodes and hyperedges is denoted as $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}_H|}$. $\text{top}(S, p_{\text{add}})$ represents a function that selects the largest elements in matrix S based on the ratio p_{add}.

Next, node-hyperedge pairs that could be noises are extracted. If the cosine similarity between the representations is low, it is likely to be a noisy relationship [13]. Therefore, by extracting pairs with low cosine similarity, nodes that could be noise within hyperedges are removed. Formally, noisy node-hyperedge pairs $\nabla\mathbf{H}$ is expressed as

$$\nabla\mathbf{H}_{ij} = \begin{cases} 1, & \text{if } \mathbf{H}_{ij} = 1 \text{ and } S_{ij} \in \text{bottom}(S, p_{\text{drop}}), \\ 0, & \text{if } \mathbf{H}_{ij} = 0, \end{cases} \tag{8}$$

where $\text{bottom}(S, p_{\text{drop}})$ represents a function that selects the smallest elements in matrix S based on the ratio p_{drop}.

Then, by adding potential relationships $\Delta\mathbf{H}$ and removing noise $\nabla\mathbf{H}$, the improved hypergraph $\mathbf{H}'$ is obtained. The calculation formula for $\mathbf{H}'$ is defined as

$$\mathbf{H}' = \mathbf{H} + \Delta\mathbf{H} - \nabla\mathbf{H}. \tag{9}$$

Finally, using the representations obtained from $\mathcal{H}'$ and $\mathcal{H}$, the loss for the hypergraph structure learning task is designed. The loss for the hypergraph structure learning task is defined as

$$\mathcal{L}_{HSL} = \sum_{n\in\mathcal{V}_H} -\log \frac{\exp(s(\boldsymbol{x}_n, \boldsymbol{x'}_n)/\tau)}{\sum_{m\in\mathcal{V}_H}\exp(s(\boldsymbol{x}_n, \boldsymbol{x'}_m)/\tau)}, \tag{10}$$

where $\boldsymbol{x}$ is the node representation generated from $\mathcal{H}$, and $\boldsymbol{x'}$ is the node representation generated from $\mathcal{H}'$. τ is a temperature hyper-parameter. The above operations are performed for each user hypergraph $\mathcal{H}_u$ and item hypergraph $\mathcal{H}_i$.

C. *Model Optimization*

HGCL is trained through multi-task training that incorporates contrastive learning and hypergraph structure learning for the recommendation task.

In the recommendation task, the model is optimized by minimizing the Bayesian Personalized Ranking (BPR) loss [28], which uses the predicted preference scores of users for items and the actual preference scores. The BPR loss is defined as

$$\mathcal{L}_{BPR} = \sum_{u\in U}\sum_{i\in N_u}\sum_{j\in I, j\notin N_u} -\log\sigma(\hat{y}_{ui} - \hat{y}_{uj}), \tag{11}$$

where $\sigma(\cdot)$ represents the sigmoid function. $\hat{y}_{ui}$ indicates the predicted preference score between user u and item i. HGCL calculates $\hat{y}_{ui}$ using a simple and efficient inner product based on the representations $\boldsymbol{r}$ obtained by applying GNN to the interaction graph $\mathcal{G}$. The formula is given by $\hat{y}_{ui} = \boldsymbol{r}_u^\top \cdot \boldsymbol{r}_i$.

TABLE I: Statistics of the datasets.

Dataset	$\#Users$	$\#Items$	$\#Interactions$	Densities
Gowalla	5,432	7,236	114,108	2.9e-3
Yelp	5,465	5,236	163,190	5.7e-3
Amazon-Book	5,890	6,236	128,572	3.5e-3

In learning with BPR loss, the model is trained so that the preference scores for pairs of users and items with interactions are relatively higher than those without interactions.

For contrastive learning, the InfoNCE loss [29] is calculated using representations acquired from $\mathcal{G}$ from and $\mathcal{H}'$. InfoNCE loss is defined as

$$\begin{aligned}\mathcal{L}_{NCE} = &\sum_{u\in U} -\log \frac{\exp(s(\boldsymbol{r}_u, \boldsymbol{x'}_u)/\tau)}{\sum_{v\in U}\exp(s(\boldsymbol{r}_u, \boldsymbol{x'}_v)/\tau)} \\ &+ \sum_{i\in I} -\log \frac{\exp(s(\boldsymbol{r}_i, \boldsymbol{x'}_i)/\tau)}{\sum_{j\in I}\exp(s(\boldsymbol{r}_i, \boldsymbol{x'}_j)/\tau)},\end{aligned} \tag{12}$$

where $\boldsymbol{r}$ is the representations acquired from $\mathcal{G}$, and $\boldsymbol{x'}$ is the representations acquired from $\mathcal{H}'$. The symbol $s(.)$ is a function that measures the similarity between the two representations, and τ is a temperature hyper-parameter. By minimizing InfoNCE loss, HGCL maximizes the consistency between the representations of the same nodes $(i.e., \{(\boldsymbol{r}_u, \boldsymbol{x'}_u)|u \in U\})$ and minimize the consistency between the representations of the different nodes $(i.e., \{(\boldsymbol{r}_u, \boldsymbol{x'}_v)|u, v \in U, u \neq v\})$.

The overall loss is defined as

$$\mathcal{L} = \mathcal{L}_{BPR} + \lambda_1\mathcal{L}_{NCE} + \lambda_2\mathcal{L}_{HSL} + \lambda_3||\Theta||_2^2, \tag{13}$$

where λ_1 adjusts the influence of the contrastive loss and λ_2 regulates the impact of the hypergraph structure learning. λ_3 is a hyperparameter that controls the strength of L_2 regularization, and Θ represents all learnable model parameters.

IV. Experiments

In this section, we validate the effectiveness of HGCL through experiments using real-world datasets.

A. Experimental Settings

*1) **Datasets**:* By following the settings of [12], we conduct experiments on three public benchmark datasets: Gowalla, Yelp, and Amazon-Book. The basic statistics of these datasets are summarized in Table I. Gowalla dataset originates from a social networking platform where users share their locations through check-ins. Yelp and Amazon-Book datasets consist of 5-point ratings provided for restaurants and books, respectively. The datasets are partitioned into training, validation, and test sets in a 7:1:2 ratio, consistent with prevailing collaborative filtering research practices [18].

*2) **Baseline**:* In our experiments, we compare our HGCL with the following baseline models:

Conventional Collaborative Filtering Model

- **MF** [4]: It is a widely adopted baseline method that extends matrix factorization by incorporating user and item bias.
- **LightGCN** [11]: It proposes to simplify the burdensome NGCF [10] framework by removing the non-linear projection and embedding transformation during the message passing.

Contrastive Collaborative Filtering Model

- **SGL** [14]: It adopts self-supervised learning to enhance recommendation. We adopt SGL-ED as the instantiation of SGL.
- **SimGCL** [30]: It develops a graph augmentation-free CL method to improve the recommendation performance. It constructs the contrastive pairs by simply perturbing the learned node representations.
- **NCL** [17]: It designs structure-aware contrastive learning that pulls the representations of a node (a user or item) and the representative embedding for its k-hop structural neighbors.
- **HCCF** [18]: It uses implicitly learned hypergraph and cross-view hypergraph contrastive learning to learn both local and global collaborative relationships.
- **LightGCL** [31]: It develops SVD-based data augmentation for contrastive learning.

Hypergraph-based Collaborative Filtering Model

- **DHCF** [23]: The jump hypergraph convolution is introduced into the dual-channel collaborative filtering to perform message passing with prior information. The divide-and-conquer scheme is used for dual-channel learning.

*3) **Evaluation**:* By following the settings of [18], we adopt two widely used evaluation metrics for recommender systems, i.e., Recall@N, and NDCG@N. Recall measures the proportion of user-preferred items that are covered by the recommendation list. NDCG evaluates the quality of a ranked list considering relevance and item position. To ensure a fair evaluation, all models are trained using Adam optimizer [32]. The size of the embedding is fixed at 32, and the batch size is fixed at 4096.

B. Overall Performance Comparisons

Table II shows the overall performance comparison between HGCL and baseline models. The rightmost column shows the percent improvement in accuracy of the HGCL from the most competitive method. We have the following observations:

- When comparing the conventional collaborative filtering model with the CL-based model, we can confirm that all CL-based models outperformed MF and LightGCN. It confirms the benefits of integrating self-supervised learning into collaborative filtering. NCL showed the best results among CL-based baselines in many cases. We believe this is because NCL can implicitly capture high-order correlations through its own data augmentation. Additionally, we can see that DHCF demonstrates relatively low accuracy. We believe that this result is due to the inability to effectively construct the hypergraph in the presence of noise, leading to a decrease in accuracy.
- Comparing the HGCL to baseline models, we can see that HGCL outperforms baseline methodology in all cases.

TABLE II: Performance comparison of *Recall* and *NDCG* on Gowalla, Yelp, Amazon-Book datasets. The best-performing model on each dataset and metrics are highlighted in bold, and the second-best model is underlined.

Dataset	Metric	MF	LightGCN	SGL	SimGCL	NCL	HCCF	LightGCL	DHCF	HGCL	improv.
Gowalla	Recall@20	0.2011	0.2152	0.2281	0.2260	0.2260	0.2208	0.2247	0.1367	**0.2377**	+4.21%
	NDCG@20	0.1460	0.1556	0.1650	0.1658	0.1658	0.1580	0.1636	0.0952	**0.1707**	+2.96%
	Recall@40	0.2810	0.3002	0.3169	0.3152	0.3172	0.3045	0.3172	0.2057	**0.3228**	+1.76%
	NDCG@40	0.1710	0.1807	0.1909	0.1900	0.1920	0.1826	0.1910	0.1222	**0.1954**	+1.77%
Yelp	Recall@20	0.1585	0.1681	0.1776	0.1744	0.1783	0.1717	0.1719	0.1634	**0.1860**	+4.32%
	NDCG@20	0.1021	0.1112	0.1172	0.1152	0.1180	0.1122	0.1154	0.1069	**0.1242**	+5.25%
	Recall@40	0.2597	0.2608	0.2646	0.2711	0.2726	0.2580	0.2668	0.2079	**0.2823**	+3.55%
	NDCG@40	0.1374	0.1402	0.1426	0.1455	0.1476	0.1366	0.1435	0.1059	**0.1517**	+2.78%
Amazon-Book	Recall@20	0.1477	0.1712	0.1784	0.1777	0.1759	0.1480	0.1768	0.1333	**0.1873**	+4.99%
	NDCG@20	0.1003	0.1167	0.1224	0.1239	0.1226	0.1005	0.1252	0.0847	**0.1285**	+2.63%
	Recall@40	0.2262	0.2383	0.2481	0.2404	0.2394	0.2031	0.2426	0.1887	**0.2543**	+3.42%
	NDCG@40	0.1312	0.1367	0.1432	0.1415	0.1418	0.1172	0.1445	0.1031	**0.1490**	+3.11%

HGCL surpasses CL-based baselines like NCL, which partially consider high-order correlations, achieving a 5.25% accuracy improvement on the Yelp dataset in NDCG@20. We believe that using explicit hypergraphs as contrastive views is more effective than NCL data augmentation in capturing high-order correlations and further improving accuracy. Compared to DHCF, the proposed method is significantly more accurate. We believe this is due to the design that employs hypergraph structure learning, which improves the robustness of the hypergraph against noise. Moreover, comparing the results for @20 and @40 of the evaluation metric, HGCL shows greater enhancements in the short-length ranking task. This aligns better with real-world recommendation scenarios.

C. *Ablation Study*

To exploit the effectiveness of each component of the proposed HGCL, we conduct the ablation study on the Gowalla dataset. Fig. 4 shows the results of Recall and NDCG when each element is removed from HGCL. w/o HG is a version of HGCL that removes the hypergraph element and uses only interaction graphs. w/o HSL is a version of HGCL without the hypergraph structure learning component.

As we can see, the accuracy of w/o HG has decreased significantly from HGCL. We believe this difference in accuracy is due to the use of hypergraphs that capture high-order correlations that cannot be captured by the interaction graph alone. Comparing w/o HSL and HGCL, we can confirm that the accuracy decreases when hypergraph structure learning is removed. We believe that learning the hypergraph structure has improved the recommendation accuracy by detecting potential node relationships and noisy relationships and optimizing the hypergraphs for the recommendation task. From the above observations, all of the proposed components are beneficial to recommendations.

D. *Effect of HGCL on alleviating over-smoothing*

In this part, we analyze the effects of HGCL on alleviating over-smoothing. We compared the recommendation accuracy for different numbers of convolutional neighborhoods. The graph-based methods LightGCN and LightGCL varied the number of GCN layers L, while HGCL varied the value of k_{hyper}.

As we can see, LightGCN and LightGCL show a decrease in accuracy as the number of GCN layers is increased. In particular, the accuracy of LightGCL is greatly decreased when $L = 3$, confirming that over-smoothing has occurred. On the other hand, the accuracy of HGCL improves with increasing k-hop neighbors to convolute. We believe that this is because using hypergraphs, where hyperedges connect the item and the user's k-order reachable neighbors, allows for direct convolution of the k-hop neighborhoods. This approach acquires the features of distant nodes without causing over-smoothing and captures high-order correlations effectively.

E. *Noise immunity of the HGCL*

We conduct experiments to evaluate the robustness of HGCL across different noise ratios. We injected random noisy interactions into the training set of the Yelp dataset while keeping the testing set unchanged.

Fig. 6 shows the results of comparing DHCF and HGCL in the polluted Yelp dataset. From the figure we can observe that increasing the ratio of noisy interactions significantly reduces the performance of DHCF. The performance degradation of HGCL is much smaller than that of DHCF. We believe that, in DHCF, the weakness of the hypergraph against noise is pronounced, resulting in a large loss of accuracy. In contrast, in HGCL, the hypergraph structure learning is effective in reducing the loss of accuracy. These observations further confirm the importance of denoising in recommendation, and demonstrate the robustness and effectiveness of HGCL.

V. RELATED WORK

A. *Hypergraph*

The hypergraph has been employed to model high-order correlations among data [22], [33]–[35]. Zhou et al. initially introduced hypergraph learning and designed a propagation process on the hypergraph [22]. Since then, the hypergraph

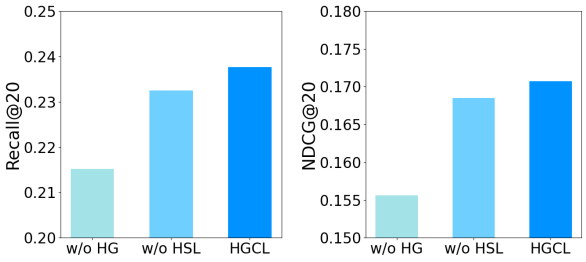

Fig. 4: Ablation study on key components of HGCL in the Gowalla dataset.

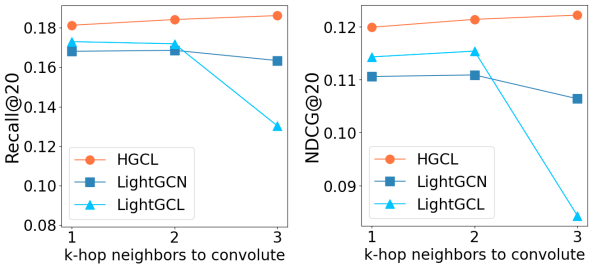

Fig. 5: Changes in recommendation accuracy when varying the number of k-hop neighbors being convoluted in the Gowalla dataset.

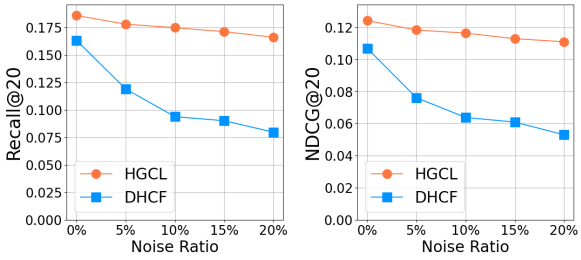

Fig. 6: Comparison of recommendation accuracy when varying the ratio of noise in the Yelp dataset.

has been utilized across multiple domains, such as video object segmentation tasks [36] and image retrieval tasks [37]. Moreover, Uthsav et al. introduced the spectral theory of hypergraphs with edge-dependent vertex weights using a random walk method [38]. Subsequently, Feng et al. proposed hypergraph convolution operations to more effectively leverage high-order data correlations for representation learning [27], enabling the handling of complex graph structures.

In recent years, recognizing the effectiveness of hypergraph, hypergraph-based recommendation models have been proposed [23], [24], [39], [40]. These models explore methods to capture high-order correlations between user and item and utilize them for recommendations, achieving good performance. However, the use of the hypergraph for contrastive learning remains an area yet to be explored.

B. *Contrastive Learning for Recommendation*

As a popular self-supervised learning paradigm, contrastive learning attempts to learn invariant representations through data augmentation. It generates contrastive views through data augmentation and maximizes mutual information to ensure consistency between views [41], [42]. Contrastive learning has demonstrated effective performance across various domains including visual data representations [43], natural language processing [44], and graph representation learning [45]. These approaches study task-specific data augmentation methods.

In recent years, contrastive learning has also been integrated into recommender systems [14], [17], [25], [30]. As a representative model, SGL [14] designs structural graph augmentation methods to generate contrasting views, thereby improving recommendation accuracy and model robustness. SimGCL [30] revisits structural augmentation methods and suggests simpler and more effective feature augmentation. NCL [17] proposes contrastive learning utilizing representations of neighboring nodes, using representations of k-hop adjacent nodes as contrasting views. HCCF [25] employs implicit learning of graph structure and cross-view contrastive learning to learn local and global collaborative relationships. However, these models have limitations in capturing high-order correlations between users and items, leading to constrained recommendation accuracy.

VI. Conclusion

In this paper, to address the insufficient capturing of high-order correlations and the vulnerability to noise in conventional collaborative filtering models, we propose HyperGraph Contrastive Learning with graph structure learning for recommendation, named HGCL. HGCL performs contrastive learning between representations from the user-item interaction graph and hypergraphs, integrating relationship features and high-order correlations to create more effective representations. Furthermore, to address the noise vulnerability of hypergraphs, we modify the hypergraph structure learning method for the recommendation task and incorporate it into HGCL, which detects and adjusts node relationships to create improved hypergraphs. In the evaluation experiment, HGCL outperforms the baseline model and demonstrates its effectiveness through experiments using three real-world datasets. In the future, we plan to investigate a more lightweight hypergraph-based recommendation model.

Acknowledgment

This work is partially supported by JST CREST Grant Number JPMJCR21F2.

References

[1] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, “Graph neural networks in recommender systems: a survey,” ACM Computing Surveys, vol.55, no.5, pp.1–37, 2022.
[2] X. Su, and T.M. Khoshgoftaar, “A survey of collaborative filtering techniques,” Advances in artificial intelligence, vol.2009, 2009.

[3] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," Proc. AAAI, vol.31, no.1, 2017.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol.42, no.8, pp.30–37, 2009.

[5] S. Sedhain, A.K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," Proc. WWW, pp.111–112, 2015.

[6] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," Proc. SIGIR, pp.335–344, 2017.

[7] T.N. Kipf, and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in neural information processing systems, vol.30, 2017.

[9] R.v.d. Berg, T.N. Kipf, and M. Welling, "Graph convolutional matrix completion," arXiv preprint arXiv:1706.02263, 2017.

[10] X. Wang, X. He, M. Wang, F. Feng, and T.S. Chua, "Neural graph collaborative filtering," Proc. SIGIR, pp.165–174, 2019.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," Proc. SIGIR, pp.639–648, 2020.

[12] X. Ren, L. Xia, Y. Yang, W. Wei, T. Wang, X. Cai, and C. Huang, "Sslrec: A self-supervised learning framework for recommendation," Proc. WSDM, pp.567–575, 2024.

[13] C. Tian, Y. Xie, Y. Li, N. Yang, and W.X. Zhao, "Learning to denoise unreliable interactions for graph collaborative filtering," Proc. SIGIR, pp.122–132, 2022.

[14] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," Proc. SIGIR, pp.726–735, 2021.

[15] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.35, no.4, pp.849–862, 2012.

[16] Y. Yang, Z. Wu, L. Wu, K. Zhang, R. Hong, Z. Zhang, J. Zhou, and M. Wang, "Generative-contrastive graph learning for recommendation," Proc. SIGIR, pp.1117–1126, 2023.

[17] Z. Lin, C. Tian, Y. Hou, and W.X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," Proc. WWW, pp.2320–2329, 2022.

[18] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," Proc. SIGIR, pp.70–79, 2022.

[19] Y. Min, F. Wenkel, and G. Wolf, "Scattering gcn: Overcoming oversmoothness in graph convolutional networks," Advances in neural information processing systems, vol.33, pp.14498–14508, 2020.

[20] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," Advances in neural information processing systems, vol.33, pp.4917–4928, 2020.

[21] J. Sun, and Y. Zhang, "Multi-graph convolutional neural networks for representation learning in recommendation," Proc. IEEE ICDM, 2019.

[22] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," Advances in neural information processing systems, vol.19, 2006.

[23] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, "Dual channel hypergraph collaborative filtering," Proc. SIGKDD, pp.2020–2029, 2020.

[24] Z. Han, X. Zheng, C. Chen, W. Cheng, and Y. Yao, "Intra and inter domain hypergraph convolutional network for cross-domain recommendation," Proc. WWW, pp.449–459, 2023.

[25] D. Cai, M. Song, C. Sun, B. Zhang, S. Hong, and H. Li, "Hypergraph structure learning for hypergraph neural networks.," Proc. IJCAI, pp.1923–1929, 2022.

[26] A.R. Benson, D.F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," Science, vol.353, no.6295, pp.163–166, 2016.

[27] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," Proc. AAAI, vol.33, no.01, pp.3558–3565, 2019.

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

[29] M. Gutmann, and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," Proc. AISTATS, pp.297–304, 2010.

[30] 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," Proc. SIGIR, pp.1294–1303, 2022.

[31] X. Cai, C. Huang, L. Xia, and X. Ren, "Lightgcl: Simple yet effective graph contrastive learning for recommendation," arXiv preprint arXiv:2302.08191, 2023.

[32] D.P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[33] H. Chen, H. Yin, X. Sun, T. Chen, B. Gabrys, and K. Musial, "Multi-level graph convolutional networks for cross-platform anchor link prediction," Proc. SIGKDD, pp.1503–1511, 2020.

[34] Y. Gao, M. Wang, Z.J. Zha, J. Shen, X. Li, and X. Wu, "Visual-textual joint relevance learning for tag-based social image search," IEEE Transactions on Image Processing, vol.22, no.1, pp.363–376, 2012.

[35] T. Hwang, Z. Tian, R. Kuangy, and J.P. Kocher, "Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction," Proc. ICDM, pp.293–302, 2008.

[36] Y. Huang, Q. Liu, and D. Metaxas, "Video object segmentation by hypergraph cut," Proc. CVPR, pp.1738–1745, 2009.

[37] Y. Huang, Q. Liu, S. Zhang, and D.N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," Proc. CVPR, pp.3376–3383, 2010.

[38] U. Chitra, and B. Raphael, "Random walks on hypergraphs with edge-dependent vertex weights," Proc. ICML, pp.1172–1181PMLR, 2019.

[39] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," Proc. AAAI, vol.35, no.5, pp.4503–4511, 2021.

[40] J. Yu, H. Yin, J. Li, Q. Wang, N.Q.V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," Proc. WWW, pp.413–424, 2021.

[41] A.v.d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748, 2018.

[42] W. Wei, C. Huang, L. Xia, Y. Xu, J. Zhao, and D. Yin, "Contrastive meta learning with behavior multiplicity for recommendation," Proc. WSDM, pp.1120–1128, 2022.

[43] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," Proc. ICML, pp.1597–1607, 2020.

[44] D. Stojanovski, B. Krojer, D. Peskov, and A. Fraser, "Contracat: Contrastive coreference analytical templates for machine translation," Proc. COLING, pp.4732–4749, 2020.

[45] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," Proc. SIGKDD, pp.1150–1160, 2020.