HCL: Hybrid Contrastive Learning for Graph-based Recommendation

Xiyao Ma
University of Florida
maxiy@ufl.edu

Zheng Gao
Amazon Alexa AI
zhenggao@amazon.com

Qian Hu
Amazon Alexa AI
huqia@amazon.com

Mohamed AbdelHady
Amazon Alexa AI
mbdeamz@amazon.com

***Abstract*—Graph-based collaborative filtering for recommendation has attracted great attention recently, due to its effectiveness of capturing high-order proximity among users and items. To further improve its model robustness and alleviate label-sparsity issue, contrastive learning has been introduced to polish user and item representation by contrasting different views of user/item nodes, learning necessary and robust representation for recommendation. However, we argue that prior contrastive learning approaches only explore its unsupervised intrinsic nature as a plug-in without leveraging available user-item interactions, failing to exploit the huge potential of contrastive learning. In this paper, to alleviate the above issues, we propose Hybrid Contrastive Learning for graph-based recommendation that integrates unsupervised and supervised contrastive learning. Specifically, to improve model robustness, we first present bipartite graph augmentation operations from the perspectives of node attributes and topology to generate incomplete and noisy graph views. Then, we propose a hybrid contrastive learning module that conducts unsupervised and supervised contrastive learning together. Last, we present an approach to perform hybrid contrastive learning permutationally among multiple views. Extensive experiments show that our proposed model not only outperforms state-of-the-art baselines significantly on two public datasets and one internal dataset, but also demonstrates superiority regarding to model robustness over other strong baselines.**

I. Introduction

In modern recommendation systems, neural collaborative filtering (NCF) is a prevalent technique to estimate the likelihood that a user would adopt an item based on their historical interactions [1]. Unlike conventional collaborative filtering approaches that learn user/item latent representations from matrix decomposition or reconstruction [2], NCF models employ non-linear neural network layers to learn user/item low dimensional embeddings to predict their potential interactions in an efficient and effective manner.

Despite the advantages, NCF methods still suffer from three limitations: **First**, NCF approaches only take user-item direct (first-order) interactions into consideration but ignore their latent high-order connectivity. **Second**, the observed interactions are always too sparse to yield effective user/item embeddings. This is well known as label sparsity issue. **Third**, there is inevitable noise in user-item interactions, e.g. a user is misled to click an item.

To solve foregoing limitations, graph-based collaborative filtering (GCF) methods ground a user-item bipartite graph [3], [4] by treating users and items as nodes and their interactions as edges. Then, graph convolutional networks are applied on the bipartite graph to capture high-order connectivity between users and items, learning node embeddings through information propagation and integration from their neighbor nodes. To further alleviate label sparsity issue, Self-supervised Graph Learning (SGL) and other contrastive learning-based methods [5], [6] generate two different views of the input graph and minimize discrepancies between two views of the same user (item) node and maximize discrepancies between two views of two different user (item) nodes.

Recently, SGL proposes to apply the unsupervised contrastive learning on the user-item bipartite graph as a supplement to the supervised learning loss [5]. However, available user-item interactions are only utilized to compute the supervised ranking loss, but overlook their potential usefulness in the paradigm of contrastive learning, resulting in suboptimal performance. In this work, we root in graph-based recommendation systems and propose a **H**ybrid **C**ontrastive **L**earning (HCL) model integrating unsupervised and supervised contrastive learning that is tied closer to graph-based recommendation, making full use of contrastive learning. In detail, bipartite graph augmentation operations are proposed to generate not only incomplete but also noisy graph views from the perspectives of node embeddings and topology. A hybrid contrastive learning module is proposed to equip contrastive learning with the ability of better adapting to graph-based recommendation problem. The insight is to effectively leverage observed user-item interactions to compute supervised contrastive learning loss, in addition to unsupervised contrastive loss. Furthermore, we introduce a method to perform the hybrid contrastive learning across multiple views permutationally.

Experimental results on two public and one internal datasets show the effectiveness of HCL that significantly outperforms strong state-of-the-art (SOTA) baselines. Auxiliary studies also show the benefits of HCL regarding to model robustness and generated embedding quality. To summarize, the contributions of this work are three-folds:

- We identify the limitation of existing contrastive learning methods for recommendation and propose Hybrid Contrastive Learning. To the best of our knowledge, this is the first work to jointly uncover and exploit unsupervised and supervised contrastive learning for recommendation.

This work was done during Xiyao Ma's internship in Amazon Alexa.
Zheng Gao is the corresponding author.

- We generalize a permutational approach that performs hybrid contrastive learning across multiple views which are generated to convey incomplete and noisy information with respect to node embeddings and topology.
- Extensive experiments show the superiority of HCL regarding to model generalization and robustness over SOTA baselines on two public and one internal dataset.

II. Problem Formulation

Generally, in a recommendation scenario, we have a set of users $\mathcal{U}$, a set of items $\mathcal{I}$, and observed interactions with implicit feedback $\mathcal{Y} = \{y_{ui} = 1 | u \in \mathcal{U}, i \in \mathcal{I}\}$. Following the recent graph-based collaborative filtering approaches [3], [4], we build a user-item bipartite graph $g = (\mathcal{V}, \mathcal{E})$ where node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ involves all users and items, and the edge set $\mathcal{E}$ includes all observed interactions $\mathcal{Y}$.

In general, for top-k item recommendation, we aim to learn a function that predicts the probability $\hat{y}_{ui}$ that user u would interact with item i.

III. Preliminary

In this section, we revisit LightGCN [4], a strong GCF baseline that captures the high-order connectivity from the user-item bipartite graph, and the model is trained in a supervised learning paradigm.

Each user and item is associated with an ID embedding, denoted as e_u^0 and e_i^0. Generally, LightGCN is applied on the user-item bipartite graph to learn user and item representations by aggregating the representation of its direct neighbors $\mathcal{N}$ and with the defined graph convolution operations:

$$e_u^l = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{l-1},$$
$$e_i^l = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|N_u|}} e_u^{l-1}. \quad (1)$$

where e_u^l and e_i^l indicate user and item node representation in the l-th layer ($l > 0$), respectively.

To capture the l-order connectivity among users and items, representations of the l-order neighbors are allowed to propagate through the edges to the target node by stacking l-layers of LightGCN. For example, with 2 layers of LightGCN, the second-order connectivity between two users who interacted with the same items can be learnt.

Then, an average pooling layer is applied on the learnt user and item representations from L layers to generate the final user and item vectors.

$$e = avgpooling\left(\left\{e^l \mid l = [0, \cdots, L]\right\}\right) \quad (2)$$

Next, the probability of user u would adopt item i is estimated by taking the inner product of a user vector e_u and an item vector e_i. In supervised learning paradigm, a common practice is to employ Bayesian Personalized Ranking (BPR) loss [7] that assigns higher probability to observed interactions than its unobserved interactions:

$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i) \in \mathcal{Y}, (u,j) \notin \mathcal{Y}} -\log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right), \quad (3)$$
$$\hat{y}_{ui} = e_u^\top e_i. \quad (4)$$

where (u, i) is a positive pair that is observed in the training data; while j is randomly sampled from the unobserved items of user u to build a negative pair (u, j).

IV. Proposed Method

In this work, motivated by the great success achieved by prior works [5], [8]–[10], we propose a novel method called Hybrid Contrastive Learning as illustrated in Figure 1, where LightGCN is utilized as the backbone model that is trained with BPR loss. In general, the proposed HCL has three steps: (1) We propose novel bipartite graph augmentation strategies by taking node embeddings and topology into consideration to generate different incomplete and noisy views for the input user-item graph. (2) The proposed hybrid contrastive learning performs unsupervised and supervised contrastive learning on homogeneous nodes and observed user-item interactions, respectively. (3) We conduct the hybrid contrastive learning among multiple views permutationally.

A. Bipartite Graph Augmentation

Distinct from computer vision (CV) and natural language processing (NLP) tasks where data points are isolated [8], [11], nodes in user-item bipartite graph are connected and interdependent to each other, making it challenging to apply data augmentation strategies directly from CV and NLP tasks. An existing work [5] introduces an effective edge dropout operation. However, it only considers node topology, making downstream contrastive learning easily capture the incompleteness.

In this work, we move steps further and propose bipartite graph augmentation strategies to generate different graph views that contain incomplete and noisy information about *node embedding* and *node topology* to boost downstream contrastive learning. Essentially, four bipartite graph augmentation operations O are sampled and applied k times on graph g to generate different graph views $g^{(k)} = O(g), k \in [1, 2, ..., k]$.

Node Embedding Dropout: From the perspective of creating graph views that contain incomplete node embedding, we propose to randomly zero out the node embedding values with a prior probability p. This is represented by the pink circle and square of u_1 and i_1, respectively, in Figure 1.

Edge Dropout: To generate incomplete node topology information, we randomly discard some observed edges between users and items with the same probability p. This is indicated by the dashed edge between u_1 and i_1 in Figure 1.

Edge Moving: To learn more robust representations via contrastive learning, we propose to inject some noise with a proper ratio into the generated graph views, which can also help exploit contrastive learning. To this end, we randomly move some edges by changing their ending points with the

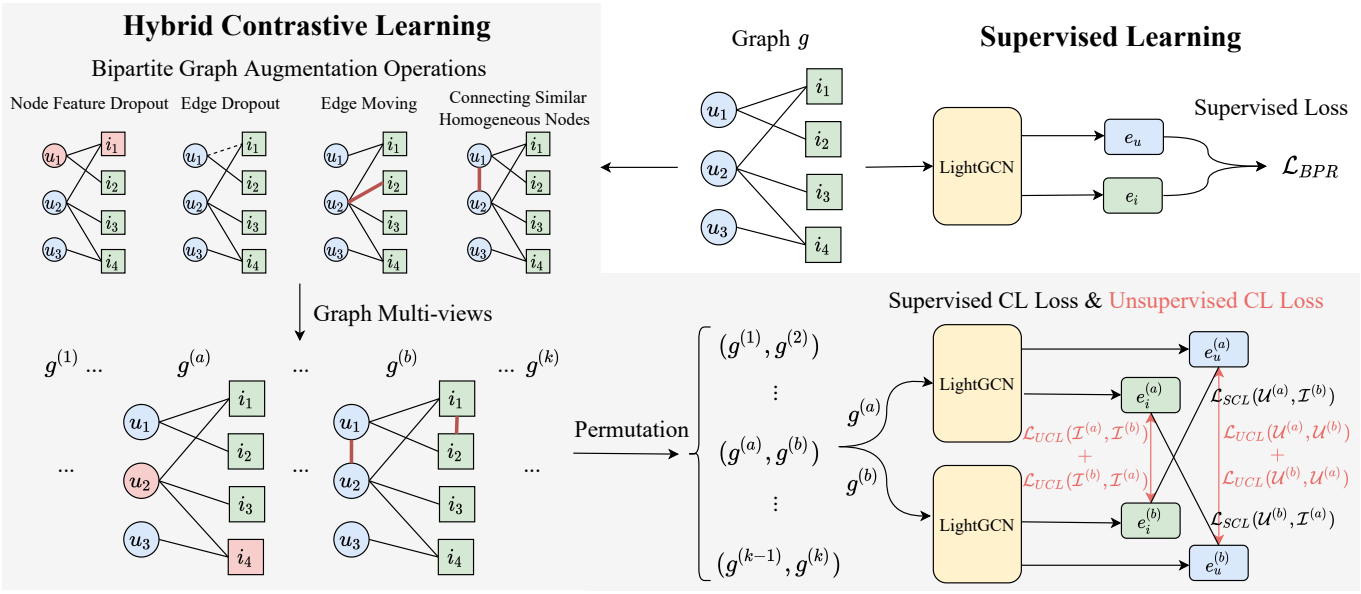


Fig. 1: The overall framework of our proposed HCL. The model is trained in multi-task learning fashion with supervised learning BPR loss and hybrid contrastive learning loss.

same probability p. For example, we can move the edge between u_1 and i_2 and let it connect u_2 to i_2 in Figure 1.

Connecting Similar Homogeneous Nodes: Another schema to inject light perturbation of node topology is to connect two similar homogeneous (same-type) nodes. For instance, u_1 and u_2 are connected with a red edge in Figure 1 as they show similar behaviors. To achieve this, we first randomly select a set of users with the probability $p/2$. Then, for each target user u in the user set, we find her top-5 similar users from all users based on their pairwise similarity. Given two users u and j and their interaction history set S_u and S_j, their pairwise similarity is computed by Jaccard Index leskovec2020mining:

$$s = \frac{|S_u \cap S_j|}{|S_u \cup S_j|} \tag{5}$$

Last, we randomly pick one similar user out of the five and connect it to user u. Analogously, we apply the same operations to connect two similar item nodes.

B. Hybrid Contrastive Learning

a) Unsupervised Contrastive Learning: Recent work explores contrastive learning on graph to alleviate the label-sparsity issue and improve model robustness [5], [6] given a pair of generated graph views. For example, SGL proposes to pull together the different views of the same node and push apart those of different nodes [5]. In detail, two generated views $((g^{(a)}, g^{(b)}),\ 1 \leqslant a < b \leqslant k)$ are feed into the same LightGCN model to obtain the user node representations $e_u^{(a)}$ and $e_u^{(b)}$. InfoNCE loss [12] is employed to compute unsupervised contrastive learning (UCL) loss by node discrimination:

$$\mathcal{L}_{UCL}(\mathcal{U}^{(a)},\mathcal{U}^{(b)}) = \sum_{u\in\mathcal{U}} -\log \frac{\exp\left(f\left(e_u^{(a)}, e_u^{(b)}\right)/\tau\right)}{\sum_{v\in\mathcal{U}} \exp\left(f\left(e_u^{(a)}, e_v^{(b)}\right)/\tau\right)} \tag{6}$$

where $f(\cdot,\cdot)$ indicates the cosine similarity function, and τ is the temperature hyper-parameter. Different views of the same user are considered as positive pairs $(e_u^{(a)}, e_u^{(b)})$, being encouraged to perform consistently; while views of different users are treated as negative pairs $(e_u^{(a)}, e_v^{(b)})$ that their mutual agreements should be minimized. We obtain item-side UCL loss $\mathcal{L}_{UCL}(\mathcal{I}^{(a)},\mathcal{I}^{(b)})$ similarly.

b) Supervised Contrastive Learning: Although great improvement, unsupervised contrastive learning overlooks the usefulness of available user-item interactions when it comes to contrastive learning given two different graph views. Herein we propose a hybrid contrastive learning module that also effectively leverage available user-item interactions. As demonstrated in Figure 1, in addition to unsupervised contrastive learning on $(e_u^{(a)}, e_u^{(b)})$ and $(e_i^{(a)}, e_i^{(b)})$, we propose to encourage the consistency of the embeddings of the users and the interacted items by computing supervised contrastive learning (SCL) loss given the observed user-item interactions $(e_u^{(b)}, e_i^{(a)})$. Following the same intuition behind contrastive learning, on the one hand, given an observed user-item interaction y_{ui}, we maximize the agreement between the user representation $e_u^{(b)}$ and the item representation $e_i^{(a)}$ generated from different views (see the numerator in Equation 7). On the other hand, we minimize the agreement between unobserved user-item pairs by uniformly sampling unobserved items for

user u (see the denominator in Equation 7):

$$\mathcal{L}_{SCL}(\mathcal{U}^{(b)},\mathcal{I}^{(a)}) = \sum_{(u,i)\in\mathcal{Y}} -\log \frac{\exp\left(f\left(e_u^{(b)}, e_i^{(a)}\right)/\tau\right)}{\sum_{q\in\mathcal{Q}}\exp\left(f\left(e_u^{(b)}, e_q^{(a)}\right)/\tau\right)} \quad (7)$$

where $\mathcal{Q}$ indicates the item set that includes one observed item and sampled unobserved items for user u.

By doing this, the model is enforced to learn the first-order user-item proximity explicitly across different incomplete and noisy views of users and items, improving the robustness and generalization of the model. Note that regarding to supervised contrastive learning, we intuitively choose to calculate $\mathcal{L}_{SCL}(\mathcal{U}^{(a)},\mathcal{I}^{(b)})$ on user and item nodes from different views instead of $\mathcal{L}_{SCL}(\mathcal{U}^{(a)},\mathcal{I}^{(a)})$ on user and item nodes from the same view, since computing SCL loss on user and item nodes from the same view is redundant when they have been used to compute BPR loss.

C. Multi-view Permutation

Recent work shows that computing the summation of contrastive leaning losses over all pairs of any two views across multiple incomplete views improves the overall performance [9]. Note that in the denominator of the the loss function $\mathcal{L}_{UCL}(\mathcal{U}^{(a)},\mathcal{U}^{(b)})$ 6, we fix one user node $\mathcal{U}^{(a)}$ in the graph view $g^{(a)}$ as the anchor and enumerates all user nodes $\mathcal{U}^{(b)}$ in the graph view $g^{(b)}$. Therefore, we also need to compute the symmetrical one, which is $\mathcal{L}_{UCL}(\mathcal{U}^{(b)},\mathcal{U}^{(a)})$ by anchoring at $\mathcal{U}^{(b)}$. Similarly, we compute $\mathcal{L}_{SCL}(\mathcal{U}^{(a)},\mathcal{I}^{(b)})$ and $\mathcal{L}_{SCL}(\mathcal{U}^{(b)},\mathcal{I}^{(a)})$ for supervised contrastive learning. The total multi-view HCL loss is the summation of the HCL loss terms computed on every pair of graph views permutationally, and each HCL loss term is the summation of unsupervised contrastive learning losses on user and item nodes and supervised contrastive learning losses.

$$\mathcal{L}_{HCL}^{multi-view} = \sum_{a,b} \mathcal{L}_{HCL}(g^{(a)}, g^{(b)}), \quad (8)$$

$$\begin{aligned}\mathcal{L}_{HCL}(g^{(a)}, g^{(b)}) =& \mathcal{L}_{UCL}(\mathcal{U}^{(a)},\mathcal{U}^{(b)}) + \mathcal{L}_{UCL}(\mathcal{U}^{(b)},\mathcal{U}^{(a)}) \\ &+\mathcal{L}_{UCL}(\mathcal{I}^{(a)},\mathcal{I}^{(b)}) + \mathcal{L}_{UCL}(\mathcal{I}^{(b)},\mathcal{I}^{(a)}) \\ &+\mathcal{L}_{SCL}(\mathcal{U}^{(a)},\mathcal{I}^{(b)}) + \mathcal{L}_{SCL}(\mathcal{U}^{(b)},\mathcal{I}^{(a)}).\end{aligned} \quad (9)$$

We trained the model in the multi-task learning fashion with the final loss, which is the summation of supervised BPR ranking loss, multi-view hybrid contrastive learning loss, and $L2$ regularization.

$$\mathcal{L}_{final} = \mathcal{L}_{BPR} + \lambda_1 \mathcal{L}_{HCL}^{multi-view} + \lambda_2 \|\Theta\|_2^2 \quad (10)$$

where λ_1 and λ_2 are two hyper-parameters to control the impacts of HCL loss and $L2$ regularization, respectively.

It is worth mentioning that although the proposed HCL requires longer training time, it delivers the same inference speed as LightGCN since contrastive learning is only leveraged during training.

V. Experiments

To validate the superiority of our proposed HCL regarding to effectiveness and robustness, we conduct extensive experiments to answer the following research questions:

- **RQ1**: How does HCL perform top-K recommendation task, compared with other SOTA models?
- **RQ2**: How do the components of HCL affect model performance with different settings?
- **RQ3**: What benefits does HCL bring for graph-based recommendation?

A. Datasets

We adopt two widely-used public datasets, Yelp2018 and Amazon-book, and one internal Recipe dataset to evaluate model performances across the experiments:

- **Yelp2018**: It is extracted from the 2018 edition of Yelp challenge. The local business like restaurants and bars are viewed as items.
- **Amazon-book**: It comes from the collection of datasets for product recommendation [13].
- **Recipe**: We collect one year (2020/6 - 2021/5) user-recipe interaction data from the production traffic of a voice assistant in the United State. Individual users are de-identified in this dataset. To ensure the dataset quality, we adopt 3-core setting that retains users and recipes with at least three interactions.

TABLE I: Statistics of the datasets

Datasets	# Users	# Items	# Interactions	Density
Yelp2018	31,668	38,048	1,561,406	0.0013
Amazon-book	52,643	91,599	2,984,108	0.00062
Recipe	57,023	26,285	347,041	0.00023

We report the statistics of three datasets in Table I. The three datasets cover different orders of magnitude about the quantity of users and items, and is expected to compare model fairly. For the two public datasets, we use the preprocessed datasets released by the authors of [3] [1]. For Recipe dataset, we extract the latest 20% of interactions for each user as the testing set, and create the validation set by taking the the latest 10% of the rest interactions of each user.

B. Baselines

In the experiment, we mainly adopt three categories of models as baselines for performance comparison: Non-GCF models (MF, NCF), GCF models (NGCF, LightGCN), and GCF model with contrastive learning (SGL):

- **MF-BPR** [7]: Matrix factorization is trained by BPR loss, modeling user-item direct interactions with an inner product interaction function.
- **NCF** [14]: It is a strong non-graph-based method that utilizes neural networks with non-linear activation function to model interactions between user and item embeddings.
- **NGCF** [3]: It is a graph-based collaborative filtering model that adds a feature interaction module with

[1] https://github.com/wujcan/SGL

TABLE II: Model Performance Comparison on Public Datasets.

Datasets		Yelp2018				Amazon-book			
Category	Models	Precision@20	Recall@20	HitRate@20	NDCG@20	Precision@20	Recall@20	HitRate@20	NDCG@20
Non-GCF	MF-BPR [7]	0.0223	0.0491	0.3224	0.0394	0.0119	0.0285	0.1801	0.0221
	NCF [14]	0.0203	0.0441	0.3	0.0357	0.0097	0.023	0.1532	0.0174
GCF	NGCF [3]	0.0229	0.0511	0.3323	0.0417	0.012	0.0294	0.182	0.0221
	LightGCN [4]	0.0259	0.0575	0.361	0.047	0.0143	0.0356	0.2134	0.027
GCF+CL	SGL [5]	0.0262	0.0581	0.366	0.0476	0.0147	0.0367	0.2176	0.0282
	HCL (k=2) [5]	0.0281	0.063	0.3846	0.0516	0.0162	0.0387	0.2247	0.0297
	HCL (k=3)	**0.0287**	**0.0643**	**0.3935**	**0.0525**	**0.0166**	**0.0395**	**0.2304**	**0.0307**
	Improvement (%)	9.5	10.7	7.5	10.3	12.9	7.6	5.9	8.9

element-wise product of user and item embedding into message passing.

- **LightGCN** [4]: It is a SOTA GCF baseline that presents a light convolution message aggregation function by removing the unnecessary parts from NGCF, greatly improving the model performance.
- **SGL** [5]: It is a SOTA GCF-based model that performs unsupervised contrastive learning as a supplement to the supervised BPR loss.

C. Main Results (RQ1)

We first compare model performances for top-K recommendation on the testing set of two public datasets in terms of widely-used metrics for recommendation, Precision@K, Recall@K, HitRate@K, and NDCG@K [15], and we adopt the same $K = 20$ following baseline papers.

Before comparing the model overall performances, we would like to point out the differences of training settings used in this work and baseline papers [3]–[5]. In the baseline papers, models are trained on the whole training set and evaluated on the testing set without validation set, and the models might be overfitted. Thus, we extract the latest 10% interactions of each user in the training set to build a validation set that is utilized to avoid overfitting during training. We re-train all models on the truncated training data, resulting in lower scores than the reported scores in baseline papers.

As reported in Table II, compared with Non-GCF methods like MF-BPR and NCF that treat each user-item interaction independently, GCF approaches like NGCF and LightGCN show higher performance by discovering and capturing the implicit high-order relations among users and items on the grounded user-item bipartite graph. SGL introduces contrastive learning to alleviate the label-sparsity issue and improve the performance further. Finally, given different numbers of graph views ($k = 2$ or $k = 3$), our proposed HCL consistently yields the best performance on both datasets. Specifically, when $k = 3$, HCL outperforms SGL in terms of Recall@20 and NDCG@20 by 10.7% and 10.3% on Yelp2018 dataset, respectively.

We further evaluate models on a real-world and internal Recipe dataset. We are not allowed to share the absolute performance metrics, so we instead show relative performance improvement of our method compared to a strong baseline SGL. Distinct from the observations from the two public datasets, NCF and LightGCN are two best baselines while SGL delivers inferior results. This might be due to two potential reasons: (1) SGL generates views only with edge dropout, which inherently limit the downstream contrastive learning. (2) SGL does not compute the unsupervised contrastive learning permutationally and prone to stuck at local minimum. On the contrary, the proposed HCL exploits contrastive learning comprehensively in both unsupervised and supervised learning settings and outperforms all strong baselines by a large margin.

TABLE III: Model Relative Performance Comparison on Recipe Dataset with SGL.

	Recipe			
Models	Precision@20	Recall@20	HitRate@20	NDCG@20
MF	+2.2%	+4.9%	+4.8%	+2.7%
NCF	+45.6%	+10.5%	+4.0%	+6.7%
NGCF	-2.2%	-46.0%	-50.2%	-55%
LightGCN	+23.9%	+9.4%	+3.9%	+5.5%
SGL	+0.0%	+0.00%	+0.00%	+0.0%
HCL (k=2)	+50%	+15.4%	+6.4%	10.2%
HCL (k=3)	**+54.3%**	**+16.3%**	**+9.1%**	**15.6%**

TABLE IV: Ablation Study.

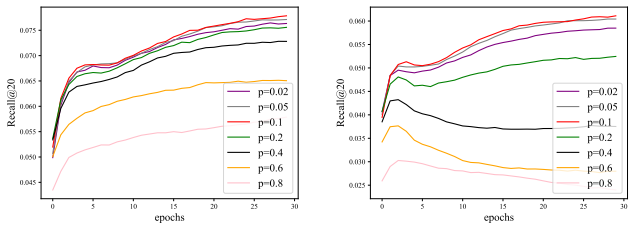
Datasets	Yelp2018		Amazon-book	
Methods	Recall@20	NDCG@20	Recall@20	NDCG@20
HCL	0.0643	0.0525	0.0395	0.0307
- DA	0.063 (-2.0%)	0.0510 (-2.8%)	0.0385 (-2.5%)	0.0298 (-3.0%)
- SCL	**0.0623 (-3.1%)**	**0.0506 (-3.6%)**	**0.0381 (-3.5%)**	**0.0292 (-4.9%)**
- MV	0.063 (-2.0%)	0.0516 (-1.7%)	0.0387 (-2.0%)	0.0297 (-3.3%)
- PL	0.0627 (-2.5%)	0.0508 (-3.2%)	0.0385 (-2.5%)	0.0296 (-3.6%)

D. Model Study (RQ2)

a) Abalation Study: We conduct ablation study to quantify the impact of the components in our proposed HCL and report the corresponding degradations in Table IV, where DA, SCL, MV, and PL indicate the three our proposed bipartite graph augmentation operations (e.g, node feature dropout, edge moving, and connecting similar homogeneous nodes), supervised contrastive learning, multiple views, and permutational losses, respectively. Each proposed component contributes in a certain degree to the improvement of model performance, which demonstrates the reasonability of our proposed architecture. Taking a close look at the results, removing SCL hurts the performance most among all the settings, which demonstrates that SCL can effectively leverage the available user-item interactions to learn better representation from different views.

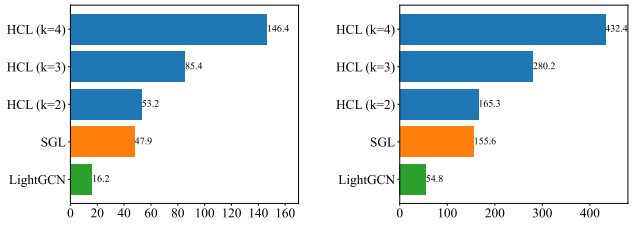
b) Effect of probability p: We further measure the impact of the probability p used for generating different incomplete

and noisy views in bipartite graph augmentation stage. As depicted in Figure 2, $p = 0.1$ has the best learning curve, while adopting a very large or a very small value of p hurts the model performance more or less. On one hand, removing (injecting) too much information (noise) restricts the model to learn meaningful representations; on the other hand, removing (injecting) too less information (noise) makes the contrastive learning tasks very easy to be learnt and fail to provide additional learning signals for model training. Therefore, it is crucial to select a suitable value for probability p to control the magnitude of incomplete and noisy information conveyed in each view of the input graph.



(a) Yelp2018 (b) Amazon-book

Fig. 2: Effect of probability p for bipartite graph augmentation.



(a) Yelp2018 (b) Amazon-book

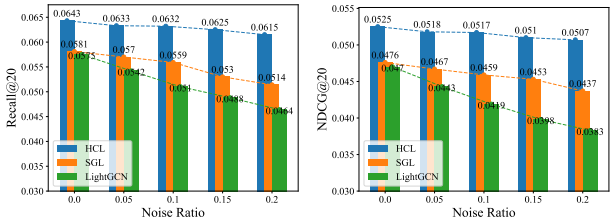
Fig. 3: Time (seconds) of training models for one epoch.

c) Effect of view quantity k: We report the time of training HCL with different numbers of graph views for one epoch in Figure 3. The testing platform is equipped with four Nvidia V100 (16GB) graphics cards and Intel Xeon CPU E5-2686 v4. Compared with LightGCN that is trained only in supervised learning manner, incorporating contrastive learning indeed incurs extra training time. When $k = 2$, HCL has comparable training time with SGL, but delivers much higher performance than SGL (shown in Table II). However, training time grows exponentially as k increases. Therefore, for computational efficiency, we adopt $k = 2$ and $k = 3$ during experiments, and we do not report the model performance when $k = 4$ as it takes too long to get the model trained.

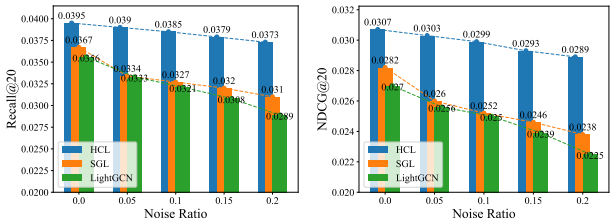
E. Benefits of HCL (RQ3)

a) Evaluating Model Robustness: As mentioned above, our proposed model are expected to improve model robustness by contrasting different incomplete and noisy views of the input graph. To validate this point, we conduct experiments that train models with noisy user-item interactions by poisoning the training set with different ratios of unobserved user-item interaction data. We remain the validation and testing set unchanged. We plot the model performance in terms of Recall@20 and NDCG@20 with different noise ratios on two public datasets in Figure 4.

We first observe that training with noisy data decreases the model performance on the two datasets. However, our proposed HCL has a much smaller degradation rate than SGL and LigthGCN as the noise ratio increases. Furthermore, HCL trained with noise ratio of 0.2 even outperforms SGL and LightGCN that are trained without noise by a large margin, showing the effectiveness and robustness of HCL.

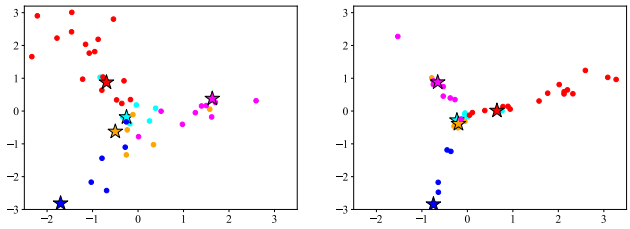


(a) On Yelp2018 dataset.



(b) On Amazon-book dataset.

Fig. 4: Model performance with noise ratios. The bar represents the results in terms of Recall@20 and NDCG@20, respectively. The dotted line indicates model performance degradation when trained with different ratios of noise data.



(a) SGL (b) HCL

Fig. 5: Visualization of users and items embeddings learnt by SGL and the proposed HCL, where the stars (★) represent different users and points (•) in same color indicate the interacted items of the user in the testing set.

b) Assessing Embedding Quality: To check whether the proposed HCL can generate good users and items embeddings, we examine the user and item embeddings generated by HCL and SGL. We randomly pick five users and their interacted items from the testing set of Yelp2018 dataset,

and the corresponding embeddings are projected into a two-dimensional space with Principal Component Analysis (PCA) [16] as visualized in Figure 5. Comparing the two figures, the item embeddings of the same user learnt by HCL are closer to each other and more prone to form clusters than SGL, indicating the better quality of embeddings generated by HCL than SGL.

F. Implementation Details

In our proposed HCL, we randomly sample 100 negative unobserved items for each user to compute the supervised contrastive learning loss. Following the baselines, with the same hyper-parameters, we train the models using Adam optimizer [17] with learning rate of $1e-3$. We set embedding dimension as 64 for both user and item embedding and 3 layers of graph neural networks for all GCF methods across the experiments. We empirically adopt different values of $\lambda_1 = 0.1, 0.1, 0.3$ and $\tau = 0.2, 0.2, 0.4$ for Yelp2018, Amazon-book, and Recipe datasets, respectively. To avoid model overfitting, we adopt $\lambda_2 = 1e-4$ for $L2$ regularization and early stop the training if the a higher recall score is not achieved for 50 continuous epochs.

VI. Related Work

a) Neural Collaborative Filtering: NCF techniques are prevalent used in modern recommender systems [1]. By learning user and item embeddings from their interactions, NCF approaches aims to predict their pairwise relationship as their final objective. NCF [14] is the pioneering work which generalizes conventional matrix factorization to neural network architecture. A series of follow-up works take advantage of NCF model architecture and involve more complicated features as auxiliary information. DIN [18] represents user embeddings leveraging not only interaction history but also profile and context features. DIEN [19] proposes an interest extractor layer to capture user temporal interests from history behavior sequence. DSIN [20] extends from DIEN model to capture user dynamic and evolving interests from session based behaviors. For other NCF variants, J-NCF [21] applies a joint neural network model which couples deep feature learning and deep interaction modeling on user-item rating matrix. DPLCF [22] preserves privacy issues in recommendation. And NICF [23] learns user interests from interactive feedback via a reinforcement approach. Beyond collaborative filtering, [24] addresses collaborative reasoning, which bridge differentiable neural networks and symbolic reasoning in a shared architecture.

b) Graph Collaborative Filtering: To uncover high-order connectivity between user and items, graph collaborative filtering (GCF) becomes an emerging topic which constructs user-item bipartite interaction graph and learns their pairwise relationships under graph convolutional network (GCN) paradigm [25]–[27]. PinSage [28] and IntentGC [29] are both graph convolutional network models which learn user & item embeddings through the information propagation in bipartite interaction graphs. NGCF [3] proposes a stacked propagation layer to learn node embedding from both node itself and neighbour nodes. Extended from this, LR-GCCF [30] removes non-linear transformations and LightGCN [4] only keeps normalized sum of neighbor embeddings to accelerate training speed. IMP-GCN [31] enhances LightGCN by passing user interests to learn node embeddings. Similarly, DGCF [32] involve user intents and SHCF [33] considers node auxiliary attributes into embedding learning process. DGCN and MixGCF [34], [35] both devise general negative sampling plugins to existing models.

c) Contrastive Learning: A surge of attention on contrastive learning has been dedicated to recommendation tasks to solve the label sparsity issue. SGL [5] generates multiple views of a node, maximizing the agreement between different views of the same node compared to that of other nodes. CCGL [36] and HeCo [37] are its two variants employed on either cascade or heterogeneous graphs. [6] integrates graph contrastive module and debiased contrastive module to reduce recommendation randomness. CLCRec [38] conducts a novel contrastive objective function to solve cold-start recommendation problem. NCE-PLRec [39] derives a closed-form of highly efficient linear recommendation algorithm to solve popularity recommendation bias. GCC [40] is a graph contrastive coding framework to encode and discriminate sampled subgraphs. DHCN [41] and HCGR [42] both propose hypergraph convolutional networks for session-based recommendation with multi-view contrastive learning. GraphCL [43] designs four types of graph augmentations to learn node embeddings in a contrastive and unsupervised manner. GroupIM [44] is a user-group recommendation task which contrastively regularizes user-group latent space to capture user social associations.

VII. Conclusion

In this paper, we proposed a novel framework, named HCL, with three contributions to well exploit contrastive learning for graph-based recommendation: (1) bipartite graph augmentation operations from the perspectives of node embeddings and topology, (2) hybrid contrastive learning that combines unsupervised and supervised contrastive learning, (3) performing hybrid contrastive learning permutationally across multiple views. Extensive experiments showed superiority of the proposed model regarding to model performance and robustness than several strong baselines on two public datasets and one internal dataset. In the future, we aims to improve the model further by exploring negative sampling methods and curriculum learning that gradually incorporates more difficult negative samples.

References

[1] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation," *arXiv preprint arXiv:2104.13030*, 2021.

[2] R. Zhang, Q.-d. Liu, J.-X. Wei *et al.*, "Collaborative filtering for recommender systems," in *2014 Second International Conference on Advanced Cloud and Big Data*. IEEE, 2014, pp. 301–308.

[3] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

[4] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[5] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 726–735.

[6] Z. Liu, Y. Ma, Y. Ouyang, and Z. Xiong, "Contrastive learning for recommender system," *arXiv preprint arXiv:2101.01317*, 2021.

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *ArXiv*, vol. abs/2002.05709, 2020.

[9] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *ECCV*, 2020.

[10] X. Ma, Z. Gao, Q. Hu, and M. AbdelHady, "Contrastive knowledge graph attention network for request-based recipe recommendation."

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, 2020.

[12] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *AISTATS*, 2010.

[13] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[15] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," *SIGIR Forum*, vol. 51, pp. 243–250, 2017.

[16] I. T. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*, 2011.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[18] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[19] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5941–5948.

[20] Y. Feng, F. Lv, W. Shen, M. Wang, F. Sun, Y. Zhu, and K. Yang, "Deep session interest network for click-through rate prediction," *arXiv preprint arXiv:1905.06482*, 2019.

[21] W. Chen, F. Cai, H. Chen, and M. D. Rijke, "Joint neural collaborative filtering for recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 4, pp. 1–30, 2019.

[22] C. Gao, C. Huang, D. Lin, D. Jin, and Y. Li, "Dplcf: Differentially private local collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 961–970.

[23] L. Zou, L. Xia, Y. Gu, X. Zhao, W. Liu, J. X. Huang, and D. Yin, "Neural interactive collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 749–758.

[24] H. Chen, S. Shi, Y. Li, and Y. Zhang, "Neural collaborative reasoning," in *Proceedings of the Web Conference 2021*, 2021, pp. 1516–1527.

[25] S. Wu, F. Sun, W. Zhang, and B. Cui, "Graph neural networks in recommender systems: a survey," *arXiv preprint arXiv:2011.02260*, 2020.

[26] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu, "Graph learning based recommender systems: A review," *arXiv preprint arXiv:2105.06339*, 2021.

[27] J. Sun, Z. Cheng, S. Zuberi, F. Pérez, and M. Volkovs, "Hgcf: Hyperbolic graph convolution networks for collaborative filtering," in *Proceedings of the Web Conference 2021*, 2021, pp. 593–601.

[28] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[29] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2347–2357.

[30] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 27–34.

[31] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing gcn for recommendation," in *Proceedings of the Web Conference 2021*, 2021, pp. 1296–1305.

[32] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1001–1010.

[33] C. Li, L. Hu, C. Shi, G. Song, and Y. Lu, "Sequence-aware heterogeneous graph neural collaborative filtering," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 64–72.

[34] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, and J. Tang, "Mixgcf: An improved training method for graph neural network-based recommender systems," 2021.

[35] Y. Zheng, C. Gao, L. Chen, D. Jin, and Y. Li, "Dgcn: Diversified recommendation with graph convolutional networks," in *Proceedings of the Web Conference 2021*, 2021, pp. 401–412.

[36] X. Xu, F. Zhou, K. Zhang, and S. Liu, "Ccgl: Contrastive cascade graph learning," *arXiv preprint arXiv:2107.12576*, 2021.

[37] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," *arXiv preprint arXiv:2105.09111*, 2021.

[38] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T.-S. Chua, "Contrastive learning for cold-start recommendation," *arXiv preprint arXiv:2107.05315*, 2021.

[39] G. Wu, M. Volkovs, C. L. Soon, S. Sanner, and H. Rai, "Noise contrastive estimation for one-class collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 135–144.

[40] 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," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[41] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4503–4511.

[42] N. Guo, X. Liu, S. Li, Q. Ma, Y. Zhao, B. Han, L. Zheng, K. Gao, and X. Guo, "Hcgr: Hyperbolic contrastive graph representation learning for session-based recommendation," *arXiv preprint arXiv:2107.05366*, 2021.

[43] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020.

[44] A. Sankar, Y. Wu, Y. Wu, W. Zhang, H. Yang, and H. Sundaram, "Groupim: A mutual information maximization framework for neural group recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1279–1288.