Simplify to the Limit! Embedding-Less Graph Collaborative Filtering for Recommender Systems

YI ZHANG, YIWEN ZHANG, and LEI SANG, Anhui University, Hefei, China
VICTOR S. SHENG, Texas Tech University, Lubbock, TX, USA

The tremendous positive driving effect of Graph Convolutional Network (GCN) and Graph Contrastive Learning (GCL) for recommender systems has become a consensus. GCN encoders are extensively used in recommendation models for capturing high-order connectivities between users and items, whereas GCL accelerates the training of recommendation tasks by adding extra supervision signals from contrastive objectives. However, little attention has been paid on corresponding theories that are truly tailored to recommendation tasks. From the technical perspective, Collaborative Filtering (CF) is seen as an important factor in recommender systems. It is applied to measure user–user, item–item, and user–item similarities rather than to achieve better clustering or node classification results. Besides, heuristic-based data augmentation may not be hold true in the field of recommender systems as it requires additional training costs and introduces noises that will corrupt the interaction graph structure and the semantic information of nodes. To tackle these limitations, we propose a novel Embedding-Less Graph Collaborative Filtering (EGCF) for recommendation, which is tailor-made for the problem mentioned for CF and further simplifies existing solutions. Structurally, it consists of two parts: embedding-less GCN and embedding-less GCL. The former improves user–item affinity by streamlining user-type embeddings and carrying out iterative graph convolution. And the latter utilizes three-type contrastive objectives to directly measure the alignment and the uniformity of users, items, and interaction pairs, respectively, avoiding any type of data augmentation or multi-view construction. Even though EGCF has been extremely streamlined, extensive experimental results on three classical datasets demonstrate the effectiveness of EGCF in terms of recommendation accuracy and training efficiency. The code and used datasets are released at https://github.com/BlueGhostYi/ID-GRec.

CCS Concepts: • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: Collaborative Filtering, Graph Convolutional Network, Graph Contrastive Learning, Recommendation

ACM Reference format:
Yi Zhang, Yiwen Zhang, Lei Sang, and Victor S. Sheng. 2024. Simplify to the Limit! Embedding-Less Graph Collaborative Filtering for Recommender Systems. *ACM Trans. Inf. Syst.* 43, 1, Article 22 (December 2024), 30 pages.
https://doi.org/10.1145/3701230

This work is supported by the National Science Foundation of China (Nos. 62272001 and 62206002), the Anhui Provincial Natural Science Foundation (No. 2208085QF195), and the Xunfei Zhiyuan Digital Transformation Innovation Research Special for Universities (No. 2023ZY001).
Authors' Contact Information: Yi Zhang, Anhui University, Hefei, China; e-mail: zhangyi.ahu@gmail.com; Yiwen Zhang (corresponding author), Anhui University, Hefei, China; e-mail: zhangyiwen@ahu.edu.cn; Lei Sang, Anhui University, Hefei, China; e-mail: sanglei@ahu.edu.cn; Victor S. Sheng, Texas Tech University, Lubbock, TX, USA; e-mail: victor.sheng@ttu.edu.

ACM 1558-2868/2024/12-ART22
https://doi.org/10.1145/3701230

1 Introduction

A broad consensus has been reached on **Graph Convolutional Network (GCN)** [18] for modeling user–item interactions. Users, items, and the interactions between them are naturally modeled as bipartite graph structures, such as NGCF [33] and LightGCN [10] attempting to iteratively explore higher-order interests. Based on LightGCN, the introduction of **Graph Contrastive Learning (GCL)** [48] in recommender systems also receives attention. For example, SGL [38], as a pilot study, increases the embedding consistency of the same node under various views created by random masking. Instead, SimGCL [47] does away with unnecessary **Graph Augmentation (GA)** and deftly builds views of a same node via random noise. GCN- and GCL-based recommender systems have demonstrated superior competitiveness over vector- or neural network-based techniques [4, 11, 25] via extensive practices.

It is noteworthy that the evolutions from NGCF to LightGCN and from SGL to SimGCL reflect some objective facts. First, recognizing the distinctions among various fields is necessary for adapting the same technology to different application scenarios as needed (e.g., the feature transformation of vanilla GCN is not applicable to ID-based **Collaborative Filtering (CF)** tasks [10]). Second, simplified model designs frequently outperform complex and redundant ones because they are easier to understand and optimize by both researchers and model training (e.g., SimGCL removes the laborious GA from SGL). On the one hand, a simplified model design is easier to be understood and optimized; on the other hand, a simplified model design may reduce the over-fitting and enhance the generalizability of the model with faster training iterations. The aforementioned results motivate us to think again: *Are the current GCN- and GCL-based recommendation methods truly optimized for CF tasks, and can their designs be further simplified?*

Regarding this issue, we provide a condensed user–item interaction bipartite graph in a movie scenario, as shown in Figure 1(a). Revisiting the user–item interaction bipartite graph, we note two different types of nodes (i.e., user and item) and the different semantic information they contain. However, GCN is widely used in clustering or node classification tasks for homogeneous graphs [18], which is very different from CF tasks [11, 26]. Figure 1(b) provides 2D **t-distributed stochastic neighbor embedding (t-SNE)** results [27] of user and item embeddings obtained by a fully trained LightGCN [10], which in part bridges the gap between model design and understanding the CF task. It is evident that the items which a user interacts with constantly remain nearby, which is especially pronounced for users with more specialized interests (e.g., Arthur, who only likes horror movies), while Eddie, who has a wider variety of interaction types, shows the opposite results (the items he interacts with are dispersed throughout the feature space). *This demonstrates that the CF task does not only measure user–user or item–item similarity, in terms of more prominently measuring user–item similarity.* That is, we will look for potential user–item interactions that are as close as feasible, rather than trying to categorize users or items into various groups within the feature space.

In addition, introducing GCL into recommender systems faces many challenges as well. On the one hand, whether using manual or automatic GA [14, 38] or **Feature Augmentation (FA)** [44, 47] techniques, most traditional GCL-based methods will introduce multiple auxiliary computational processes to construct **Multi-Views (MV)**, which requires additional space and time costs [47] and will inevitably corrupt the input data [44]. On the other hand, CF tasks are centered on measuring user–item similarity rather than on clustering [18] or image recognition [3, 30] tasks. Augmentation-based contrastive strategies have made breakthroughs in many verticals, but they are not really tailored for recommendation, which makes traditional GCL have limited enhancement for CF tasks.

As we have previously concluded, GCN and GCL for recommender systems not only need to be tailored but are expected to be done in a way as tiny and convenient as possible so that they can

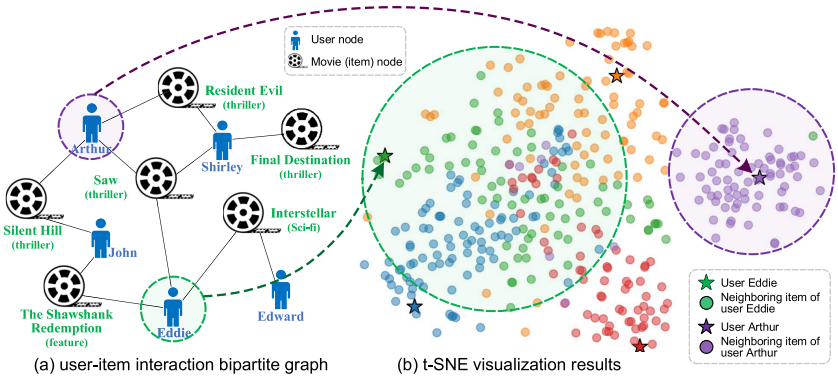


Fig. 1. Illustration of (a) a simplified user–item bipartite graph in movie recommendation scenario and (b) t-SNE of user and item embeddings obtained by well-trained LightGCN. Stars represent users, and points of the same color represent items the user interacted with (best view in color).

be more easily implemented and applied. Inspired by traditional item-based approaches [16, 42], we enhance the affinity between users and items by simplifying user-type embedding modeling via iterative information aggregation from structural neighbors (i.e., items that interacted with). This process is considered as the parameter reuse, which greatly reduces the training burden of the model and does not require additional techniques to prevent over-fitting. Furthermore, some recent studies [28, 30] expose that the centerpiece of GCL lies in aligning positive samples and uniformizing all nodes. To adapt to CF tasks, we further simplify the GCL process by proposing three augmentation-agnostic contrastive processes, which in essence enhance the consistency between user and positive samples while keeping the entire feature space uniform and realizing the dynamic tradeoffs. We refer to the aforementioned procedures as *embedding-less GCN* and *embedding-less GCL*, respectively. The proposed recommendation framework is named **Embedding-Less Graph Collaborative Filtering (EGCF)**, as shown in Figure 2(c). On the whole, EGCF optimizes the model design to the limit while maintaining competitive performance and does not introduce additional hyper-parameters or training costs. The major contributions of this article are summarized as follows:

—We empirically highlight the shortcomings in current GCN- and GCL-based recommender systems that do not take into account the unique characteristics of CF tasks, which center on computing the similarity between users and items rather than on clustering or node classification.
—We propose a minimalist graph recommendation framework called EGCF, which centers on modeling item-type representations in feature space and adapts to CF tasks by considering the alignment and the uniformity of users, items, and interaction pairs without any type of data augmentation.
—We conduct extensive experiments on three classical million-scale datasets, and our experimental results demonstrate the effectiveness and potential of EGCF in terms of recommendation performance, training efficiency, number of parameters, sparsity resistance, and robustness.

2 Preliminary and Related Work

2.1 GCN-Based Recommender Systems

Benefiting from the powerful graph structure modeling capability of GCNs [18], GCN-based recommender systems are beginning to attract the attention of researchers and are emerging in

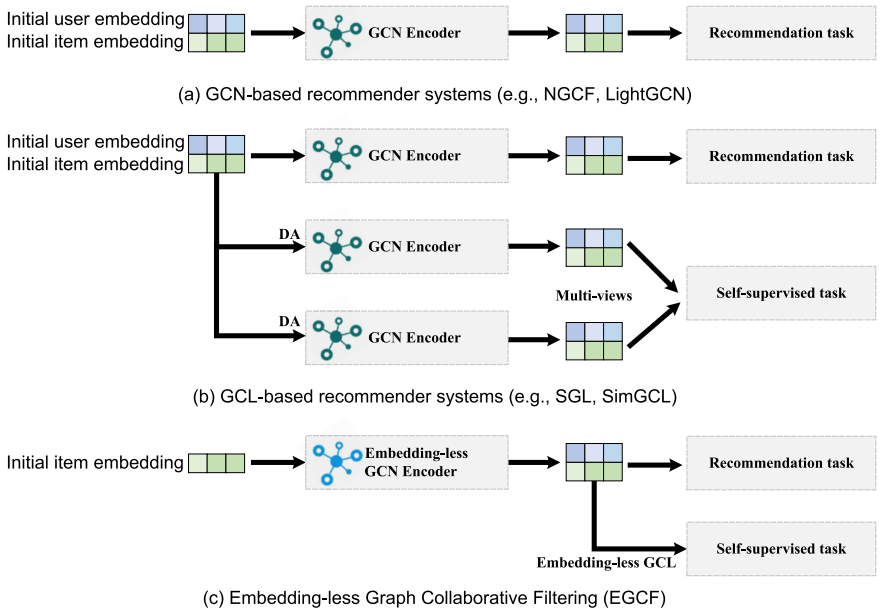


Fig. 2. (a) Framework for GCN-based recommender systems; (b) framework for GCL-based recommender systems, and "DA" denotes any type of data augmentation; (c) framework of the proposed EGCF model.

various sub-fields. In an arbitrary recommendation scenario, the interaction between the set of users $\mathcal{U}$ and the set of items $\mathcal{I}$ can be abstracted as an interaction graph $\mathcal{G}$. Early works attempt to model high-order relationships on graph $\mathcal{G}$ via basic GCN paradigms, such as PinSage [45] and NGCF [33]. Some subsequent studies [2, 10] point out the negative effects of non-linear activation and feature transformation in GCN on ID-based CF tasks. For example, GCCF [2] proposes a linear propagation strategy, while LightGCN [10] completely removes self-connections, non-linear activation functions, and feature transformation matrices. Considering the simplicity and popularity of LightGCN, we briefly recap the operation of it on interaction graph $\mathcal{G}$, which propagates the node information directly to its neighbors and removes the redundant feature transformation process:

$$\mathbf{e}_u^{(k)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{e}_i^{(k)}, \quad \mathbf{e}_i^{(k)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{e}_u^{(k)}, \tag{1}$$

where $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_i^{(k)}$ are the embedding representations of the user u and item i at kth layer, respectively. $\mathcal{N}_u$ and $\mathcal{N}_i$ are the first-order neighbors of the user u and the item i, respectively. Equation (1) attempts to propagate information in the form of path decay to the central node, which makes neighboring nodes (e.g., items that a user has interacted with) more likely to be close to the central node inside the feature space (as shown in Figure 1). Note that each propagation layer is given embeddings corresponding to the user u and item i, respectively, which contain different semantic information. LightGCN constructs the final representations via a mean function:

$$\mathbf{e}_u = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{e}_u^{(k)}, \quad \mathbf{e}_i = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{e}_i^{(k)}, \tag{2}$$

where K is a hyper-parameter controlling the number of layers. It is widely practiced to compute the prediction scores of user u for item i through inner product, i.e., $\hat{y}_{u,i} = \mathbf{e}_u^\top \mathbf{e}_i$. In general, the

larger $\hat{y}_{u,i}$ is, the more probability the user u has to like the item i and the closer the two nodes are in the feature space. Pairwise ranking loss (e.g., **Bayesian Personalized Ranking (BPR)** [25]) is widely used in GCN-based recommendation models:

$$\mathcal{L}_{\text{main}} = \sum_{<u,i,j>\in\mathcal{B}} -\ln\sigma(\hat{y}_{u,i,j}) = \sum_{<u,i,j>\in\mathcal{B}} -\ln\sigma(\mathbf{e}_u^\top\mathbf{e}_i - \mathbf{e}_u^\top\mathbf{e}_j), \tag{3}$$

where $\hat{y}_{u,i,j} = \mathbf{e}_u^\top\mathbf{e}_i - \mathbf{e}_u^\top\mathbf{e}_j$, $\mathcal{B}$ is a sampled mini-batch, and σ is the Sigmoid function. $i \in \mathcal{I}$ is an item that user $u \in \mathcal{U}$ has interacted with, and $j \in \mathcal{I}$ is any uninteracted one. The core idea of BPR is to constrain the positive samples to have a larger score than the negative samples, thus providing supervised signals for the recommendation task.

Subsequent GCN-based works focus more on further optimizing the details of LightGCN, including prevention of over-smoothing [15, 21], intent disentanglement [24, 35], attention [53], robustness [6], non-Euclidean spatial aggregation [51], and better negative sampling strategies [13]. The design of LightGCN also affects a range of sub-domains, including social network [54], knowledge graph [34], and cross-domain [22] recommendations. Despite the great breakthroughs of these approaches, we argue some inherent flaws centered on the conflicting nature between vanilla GCN and CF tasks. Specifically, CF lies in exploring the probabilistic relationship between users and items [11], rather than a node classification or clustering task, and the application of homogeneous GCN to the user–item interaction bipartite graph with two types of nodes may have varying degrees of negative impact [26].

2.2 GCL-Based Recommender Systems

As one of the most popular paradigms in **Self-Supervised Learning (SSL)**, GCL learns the intrinsic connections between representations through data augmentation [46]. In recent years, GCL-based recommender systems have begun to attract attention [48]. The GCL-based recommender system introduces additional graph data augmentation techniques to the original GCN model and constructs additional embedding representations $\mathbf{e}'_u$ ($\mathbf{e}'_i$) and $\mathbf{e}''_u$ ($\mathbf{e}''_i$) of user u (item i) through data augmentation techniques and provides additional supervision signals through InfoNCE loss [3, 8, 47]:

$$\mathcal{L}_{\text{InfoNCE}}^{\text{user}} = \sum_u -\log\frac{\exp\left(\mathbf{e}_u'^\top\mathbf{e}_u''/\tau\right)}{\sum_v \exp\left(\mathbf{e}_u'^\top\mathbf{e}_v''/\tau\right)}, \quad \mathcal{L}_{\text{InfoNCE}}^{\text{item}} = \sum_i -\log\frac{\exp\left(\mathbf{e}_i'^\top\mathbf{e}_i''/\tau\right)}{\sum_j \exp\left(\mathbf{e}_i'^\top\mathbf{e}_j''/\tau\right)} \tag{4}$$

where τ is the temperature parameter. The above contrastive losses work to strengthen the consistency between different views of the same node (e.g., $\mathbf{e}'_u$ and $\mathbf{e}''_u$), while keeping the negative samples (e.g., $\mathbf{e}''_v$) as far away from the positive samples (e.g., $\mathbf{e}'_u$) as possible. It should be noted that this process is independent of the main recommendation task. In widespread practice, most GCL-based recommender systems first need to define a primary recommender model (e.g., LightGCN) and then perform joint training [38, 47]:

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \lambda_1\left(\mathcal{L}_{\text{InfoNCE}}^{\text{user}} + \mathcal{L}_{\text{InfoNCE}}^{\text{item}}\right), \tag{5}$$

where $\mathcal{L}_{\text{main}}$ is the main recommendation loss defined in Equation (3), Θ is the set of model parameters, and λ_1 is a tradeoff coefficient for confirming the influence of SSL on the main recommendation task [5, 38].

As can be expected, data augmentation is central to the design of GCL, which can be categorized as either automatic or manual, with manual augmentation being sub-divided into structure augmentation and FA. Early works use graph data augmentation strategies to construct different views of the same node, e.g., SGL [38] proposes three stochastic dropout strategies, which are simple and

effective in principle but do not perform well in practice and introduce additional time and space costs. Subsequent works explore the design of non-GA strategies, e.g., SimGCL [47] introduces noise for node embedding to adaptively construct different views, while NCL [20] and CGCL [9] construct structural neighbors with the help of graph structure. Some more recent works present more novel attempts, such as SimRec [39], which constructs contrasts between teacher and student models through knowledge distillation, and VGCL [44], which combines GCL and graph variation to construct new views of nodes via graph-generative strategies [14, 37, 52].

Despite their effectiveness, we argue that most existing GCL-based methods are limited by various types of data augmentation and MV strategies that ignore the properties of CF tasks. Practically, recommendation models need to be more inclined to measure user–item similarity rather than just user–user or item–item relationships. On opposite, EGCF utilizes non-data-augmented GCL to achieve alignment and uniformity between user and item nodes, which avoids any type of additional computational cost and does not introduce additional parameters or hyper-parameters to the model design.

3 EGCF: Simplify to the Limit

The current section presents our solution EGCF in details, comprising the embedding-less GCN and the embedding-less GCL. Figure 2 displays the training flow of the EGCF and comparisons with existing GCN- and GCL-based solutions [10, 47].

3.1 Embedding-Less GCN

In an arbitrary recommendation scenario, the interactions between the set of users $\mathcal{U}$ and the set of items $\mathcal{I}$ can be abstracted as an interaction graph $\mathcal{G}$. The widely used recommendation loss in GCN- and GCL-based recommendation methods is BPR [25], which assumes the scores of positive samples are larger than those of negatives. Given the model parameters Θ, the following derivation can be obtained:

$$\frac{\partial \mathcal{L}_{\text{BPR}}}{\partial \Theta} = \sum_{<u,i,j>\in\mathcal{B}} -\frac{\partial}{\partial \Theta} \ln \sigma(\hat{y}_{u,i,j}) = \sum_{<u,i,j>\in\mathcal{B}} \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \frac{\partial}{\partial \Theta} \hat{y}_{u,i,j}. \tag{6}$$

Noting that the prediction score $\hat{y}_{u,i} = \mathbf{e}_u^\top \mathbf{e}_i$ is computed via embedding representations $\mathbf{e}_u$ and $\mathbf{e}_i$, we have the following derivations:

$$\frac{\partial \mathcal{L}_{\text{BPR}}}{\partial \mathbf{e}_u} = -\frac{\partial}{\partial \mathbf{e}_u} \ln \sigma(\hat{y}_{u,i,j}) = \frac{\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot (\mathbf{e}_j - \mathbf{e}_i). \tag{7}$$

$$\frac{\partial \mathcal{L}_{\text{BPR}}}{\partial \mathbf{e}_i} = -\frac{\partial}{\partial \mathbf{e}_i} \ln \sigma(\hat{y}_{u,i,j}) = \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \mathbf{e}_u. \tag{8}$$

From the derivations above, the gradient $\nabla_{\mathbf{e}_u} \mathcal{L}_{\text{BPR}}$ of user u depends on the discrepancy between item i and j, while the gradient of the positive sample $\nabla_{\mathbf{e}_i} \mathcal{L}_{\text{BPR}}$ depends on the user u. If the score of the negative $\hat{y}_{u,j}$ is large enough and the score of the positive one $\hat{y}_{u,i}$ is less, a larger gradient can be provided for the optimization of item i. The item i is also forced to move toward the position of the user u [12]. However, user and item embeddings are two separate vectors for optimization, and the embedding update of user u receives the influence of negative j, which makes user u not necessarily close to item i in the feature space (and possibly moving in the direction of the negative samples). As a result, user u and the items that he has interacted with not always show clustering effect within the feature space (e.g., node Eddie in Figure 1).

As we stated before, vanilla GCN as a homogeneous graph encoding technique may not be fully applicable to CF tasks with two types of nodes [10, 26]. This forces us to change our mindset and

Table 1. Performance Comparison of MF and Variant after Removal of Initial User Embedding on Yelp2018 and Amazon-Book Datasets

	Yelp2018		Amazon-Book	
	Recall@20	NDCG@20	Recall@20	NDCG@20
MF [25]	0.0539	0.0439	0.0308	0.0239
$\text{MF}_{\text{w/o user embedding}}$	**0.0642**	**0.0532**	0.0380	0.0302
NGCF [33]	0.0560	0.0456	0.0342	0.0261
LightGCN [10]	0.0639	0.0525	**0.0411**	**0.0315**

MF, matrix factorization. The best-performing models are shown in bold.

make the user u fuse with items he interacted with, which makes the prediction score $\hat{y}_{u,i}$ based on inner product or cosine similarity substantially higher than that of the negative $\hat{y}_{u,j}$, thereby effectively improve training efficiency. Based on the above analysis, we try to consider a direct measure of similarity between user and positive item while taking the gradient problem associated with items into consideration. Specifically, similar to item-based CF methods [16, 19], we further simplify the design of LightGCN [10] by discarding initial user embedding and instead constructing user representation $\mathbf{e}_u$ using first-order neighbor items:

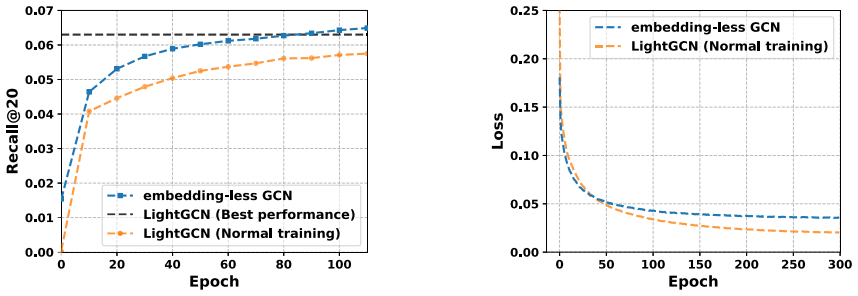
$$\mathbf{e}_u = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \mathbf{e}_i^{(0)}, \tag{9}$$

where $\mathbf{e}_i^{(0)}$ is the initial embedding of item i. With the first-order neighbor construction, we can rewrite Equation (8) as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{BPR}}}{\partial \mathbf{e}_i} &= -\frac{\partial}{\partial \mathbf{e}_i} \ln \sigma(\hat{y}_{u,i,j}) = \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \frac{\partial \left(\mathbf{e}_u^\top \mathbf{e}_i - \mathbf{e}_u^\top \mathbf{e}_j\right)}{\partial \mathbf{e}_i} \\
&= \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \left(\frac{\partial \left((\sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'})^\top \mathbf{e}_i \right)}{\partial \mathbf{e}_i} - \sum_{i' \in \mathcal{N}_u} \mathbb{I}[i'=i] \beta_{u,i'} \mathbf{e}_j \right) \\
&= \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \left(\sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'} + \sum_{i' \in \mathcal{N}_u} \mathbb{I}[i'=i] \beta_{u,i'} \mathbf{e}_i - \sum_{i' \in \mathcal{N}_u} \mathbb{I}[i'=i] \beta_{u,i'} \mathbf{e}_j \right) \\
&= \frac{-\exp(-\hat{y}_{u,i,j})}{1+\exp(-\hat{y}_{u,i,j})} \cdot \left(\sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'} + \beta_{u,i} (\mathbf{e}_i - \mathbf{e}_j) \right),
\end{aligned} \tag{10}$$

where $\beta_{u,i} = 1/\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}$. Due to $\hat{y}_{u,i,j} = \sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'}^\top \mathbf{e}_i - \sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'}^\top \mathbf{e}_j$, Equation (10) shows that the gradient update of item i is no longer directly affected by the user embedding $\mathbf{e}_u$, but the set of item embeddings $\{\mathbf{e}_{i'} | i' \in \mathcal{N}_u\}$ that interacted by user u. In other words, instead of trying to model both types of user and item nodes, we model the affinity among neighboring items. Comparing Equations (8) and (10), the gradient $\nabla_{\mathbf{e}_i} \mathcal{L}_{\text{BPR}}$ shifts from being affected by the user embedding $\mathbf{e}_u$ to being affected by the neighbor set and the negative sample j of user u. Considering that both i' and i are items that user u has interacted with (and the fact that i' can be i), the high similarity between i and i' leads to item i is more easily optimized in the direction of user u, and model can be fitted faster since the user-side backward propagation is no longer computed.

To investigate whether the above inference is true and valid, we verify it with the basic **Matrix Factorization (MF)** [25]. Specifically, MF predicts user preferences only by inner product, i.e., $\hat{y}_{\text{MF}} = \mathbf{e}_u^{(0)\top} \mathbf{e}_i^{(0)}$ [1]. By removing the initial user embedding $\mathbf{e}_u^{(0)}$ and combining it with Equation (9),



(a) Recall@20 (b) BPR Loss

Fig. 3. Training process of LightGCN and embedding-less GCN on Yelp2018 dataset.

we can obtain $\hat{y}_{\text{w/o user embedding}} = \sum_{i' \in \mathcal{N}_u} \beta_{u,i'} \mathbf{e}_{i'}^{(0)\top} \mathbf{e}_i^{(0)}$. The performance comparison of these two methods on Yelp2018 and Amazon-Book datasets [33] is shown in Table 1. The results show that removing the user embedding provides a significant improvement in the performance of MF. In this case, MF can even achieve comparable performance to GCN-based methods (e.g., NGCF [33] and LightGCN [10]). As in previous analyses, by removing user embeddings, user–item scores are more dependent on affinity between neighbors, and its possible to capture user preferences more efficiently. In addition, constructing user embeddings from neighbor information can also be considered as a parameter reuse trick as a way to reduce the training burden and mitigate over-fitting (refer to Figure 3). In Section 4.6.1, we have a more detailed analysis and validation of this idea.

Note that LightGCN still achieves competitive performance in Table 1, indicating the importance of high-order propagation on user–item interaction graph [33]. However, directly performing graph convolution like LightGCN is not feasible as there is no initial user embedding. We propose two solutions: parallel and alternating iterations.

Parallel Iteration. We first construct the user embedding $\mathbf{e}_u^{(0)*}$ via Equation (9), then concatenate it with the initial item embedding and subsequently perform light graph convolution:

$$\mathbf{e}_u^{(k)} = \mathbf{m}_{u \leftarrow i} = \sigma\left(\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{e}_i^{(k)}\right), \ \mathbf{e}_i^{(k)} = \mathbf{m}_{i \leftarrow u} = \sigma\left(\sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{e}_u^{(k)}\right), \tag{11}$$

where $k \in [1, K]$, σ is the Tanh function, and $\mathbf{m}_{a \leftarrow b}$ represents the information propagation from b to a. From a more macroscopic perspective, the parallel iteration process is an alternating matrix multiplication of Laplace matrix [18, 33]:

$$\mathbf{E}^{(0)} = \mathbf{E}_{\mathcal{U}}^{(0)*} || \mathbf{E}_{\mathcal{I}}^{(0)}, \quad \mathbf{E}^{(k)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{E}^{(k-1)}\right), \tag{12}$$

where $\mathbf{E}_{\mathcal{U}}^{(0)*}$ and $\mathbf{E}_{\mathcal{I}}^{(0)}$ are initial embedding tables of users[1] and items, respectively, and $||$ means concatenation operation. $\mathbf{A}$ is the adjacency matrix of graph $\mathcal{G}$, and $\mathbf{D}$ is a diagonal degree matrix of $\mathbf{A}$ that $D_{ii} = |\mathcal{N}_i|$.

[1]Please note that $\mathbf{E}_{\mathcal{U}}^{(0)*}$ is the initial user embeddings obtained by the construction of Equation (9), and not the trainable parameters from model initialization.

Alternating Iteration. We perform graph convolution in an iterative manner, which allows semantic information about users and items to be propagated back and forth between neighborhoods:

$$\mathbf{e}_u^{(k)} = \mathbf{m}_{u\leftarrow i} = \sigma\left(\sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\mathbf{e}_i^{(k-1)}\right), \quad \mathbf{e}_i^{(k)} = \mathbf{m}_{i\leftarrow u} = \sigma\left(\sum_{u\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\mathbf{e}_u^{(k)}\right). \tag{13}$$

From a more macroscopic perspective, the iterative propagation process is an alternating matrix multiplication of two Laplace matrices:

$$\mathbf{E}_{\mathcal{U}}^{(k)} = \sigma\left(\mathbf{D}_{\mathcal{U}}^{-\frac{1}{2}}\mathbf{R}\mathbf{D}_{\mathcal{I}}^{-\frac{1}{2}}\mathbf{E}_{\mathcal{I}}^{(k-1)}\right), \quad \mathbf{E}_{\mathcal{I}}^{(k)} = \sigma\left(\mathbf{D}_{\mathcal{I}}^{-\frac{1}{2}}\mathbf{R}^{\top}\mathbf{D}_{\mathcal{U}}^{-\frac{1}{2}}\mathbf{E}_{\mathcal{U}}^{(k)}\right), \tag{14}$$

where $\mathbf{E}_{\mathcal{U}}^{(k)}$ and $\mathbf{E}_{\mathcal{I}}^{(k)}$ are embedding tables of users and items in the kth layer, respectively. $\mathbf{R}$ is the user–item interaction matrix of graph $\mathcal{G}$, and $\mathbf{D}$ is a diagonal degree matrix that $D_{ii} = |\mathcal{N}_i|$. After propagating through K layers, we sum the embedding representations of user u and item i obtained from each layer to construct final embedding representations $\mathbf{e}_u$ and $\mathbf{e}_i$. Since we consider only one type of embedding for modeling (e.g., $\mathbf{E}_{\mathcal{I}}^{(0)}$), we refer to the process as *embedding-less GCN* for brevity. The validation experiments for the initialized item embedding as well as the two iterative methods are described in Sections 4.6.3 and 4.6.4, respectively.

More intuitively, we show the Recall@20 and loss curves of embedding-less GCN and LightGCN [10] on Yelp2018 dataset [33], respectively, as shown in Figure 3. In the same number of iterations, embedding-less GCN performs better, and the loss starts to drop off more quickly at the beginning of the training phase. Benefiting from the graph CF strategy that removes user embeddings, it allows the graph convolution operation based only on item embeddings to provide more supervision signals for model optimization, which is consistent with our analysis. It should be noted that after a certain time of training, the BPR loss of EGCF stays in a high range while LightGCN continues to decrease, suggesting that LightGCN over-distinguishes between positive and negative samples (negative samples are not always items that the user does not like) and that CF tasks are not equivalent to traditional GCN-based clustering or categorization tasks, which is a critical aspect of the bottleneck in the performance of LightGCN.

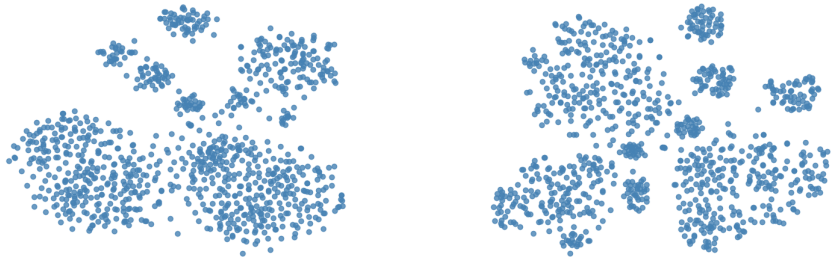
3.2 Embedding-Less GCL

Recent research has shown that ID-based recommender systems frequently take longer to converge and perform poorly since they are constrained by extremely scarce interaction data [38, 52]. Several innovative studies aim to introduce SSL [38, 47], which is independent of the primary recommendation task. Specifically, these works introduce the idea of contrastive learning to enhance coherence through different views of the same node via InfoNCE loss [3]:

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{a\in\mathcal{B}} -\log\frac{\exp\left(\mathbf{e}_a'^{\top}\mathbf{e}_a''/\tau\right)}{\sum_{b\in\mathcal{B}}\exp\left(\mathbf{e}_a'^{\top}\mathbf{e}_b''/\tau\right)}, \tag{15}$$

where $\mathbf{e}_a'$ and $\mathbf{e}_a''$ are additional embeddings of any node a, and τ is the temperature parameter. Most of the time, these embeddings need data augmentation [41], which adds additional time and space expenditures. More significantly, several studies have revealed that performance increases are not primarily driven by data augmentation [28, 29]. Recalling Equation (15), the contrastive loss essentially promotes the alignment of the positive views while moving the positive view $\mathbf{e}_a'$ away from the negative sample view $\mathbf{e}_b''$, which makes the feature space more uniform.

Measuring user–user (item–item) distance connections does not appear to be a priority when considering that the core of recommendation is to quantify the user–item similarity. In other words, rather than grouping users into similar communities, we need to bring user and positive item



(a) User embeddings (b) Item embeddings

Fig. 4. t-SNE visualization results of randomly sampled (a) user and (b) item embeddings from well-trained LightGCN on Yelp2018 dataset [33].

(i.e., interaction) closer together. With this in mind, we turn to leveraging InfoNCE to enhance the alignment and uniformity between positive interaction pairs:

$$\mathcal{L}_{\text{InfoNCE}}^{\text{inter}} = \sum_{<u,i>\in\mathcal{B}} -\log \frac{\exp\left(\mathbf{e}_u^\top \mathbf{e}_i/\tau\right)}{\sum_{o\in\mathcal{B}} \exp\left(\mathbf{e}_u^\top \mathbf{e}_o/\tau\right)}, \tag{16}$$

where o is any positive item in a batch $\mathcal{B}$. To the best of our knowledge, most of current GCL-based recommendation methods require multiple graph encoders for data augmentation or constructing different views using feature [47] or structural information [20, 38], which not only introduces additional cost but also loses the original semantic information of the graph structure [44]. Contrarily, Equation (16) no longer requires any form of data augmentation, and additional embedding and MV are also no longer necessary.

We note that $\mathcal{L}_{\text{InfoNCE}}^{\text{inter}}$ has a structural similarity to the list-pointwise loss. Specifically, some novel work directly applies the Softmax function as an alternative to the BPR loss [50]. We briefly describe the difference between Equation (16) and them. First, $\mathcal{L}_{\text{InfoNCE}}^{\text{inter}}$ adjusts for the alignment-uniformity of interaction pairs within a mini-batch, a process that does not require an additional sampling process. In contrast, softmax-based recommendation methods require sampling additional thousand negative samples. Moreover, $\mathcal{L}_{\text{InfoNCE}}^{\text{inter}}$ is only used to adjust the alignment and uniformity of feature space, and EGCF still uses only the BPR loss for recommendation, thus maintaining a fair comparison with existing works [10, 47].

Going a step further, we then move on to consider the relationship among users (or items) that are at the second priority. Due to the over-optimization of BPR [25] and the over-smoothing of GCN [18, 21], this allows for a clustering effect of user and item embeddings [40]. As shown in Figure 4, users are segmented into different communities, and the unique semantic information of nodes is lost, which results in a dearth of possibilities for cold-start users and items to recommend and be recommended, i.e., the phenomenon of popularity bias [36]. Hereby, we further consider the embedding of the node itself as a positive sample (e.g., the embedding $\mathbf{e}_u$ of user u) whereas the embedding of other nodes of the same type is treated as a negative one (e.g., the embedding $\mathbf{e}_v$ of user v), and subsequently compute the contrastive loss via InfoNCE paradigm:

$$\mathcal{L}_{\text{InfoNCE}}^{\text{user}} = \sum_{u\in\mathcal{B}} -\log \frac{\exp\left(\mathbf{e}_u^\top \mathbf{e}_u/\tau\right)}{\sum_{v\in\mathcal{B}} \exp\left(\mathbf{e}_u^\top \mathbf{e}_v/\tau\right)}, \quad \mathcal{L}_{\text{InfoNCE}}^{\text{item}} = \sum_{i\in\mathcal{B}} -\log \frac{\exp\left(\mathbf{e}_i^\top \mathbf{e}_i/\tau\right)}{\sum_{j\in\mathcal{B}} \exp\left(\mathbf{e}_i^\top \mathbf{e}_j/\tau\right)}. \tag{17}$$

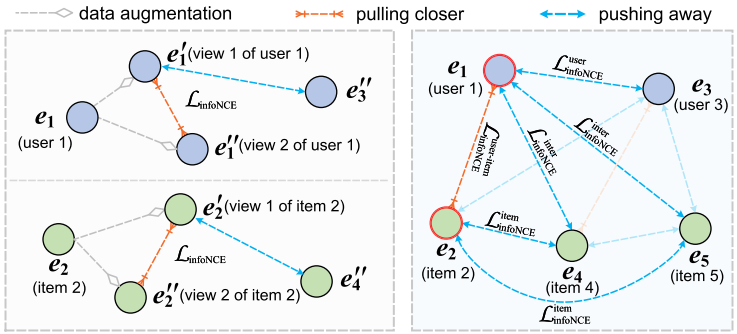


Fig. 5. Comparison of (a) traditional GCL and (b) embedding-less GCL regarding the contrastive process. Traditional GCL treats users and items independently of each other, while embedding-less GCL considers alignment and uniformity of users, items, and user–item pairs.

Considering that the embedding involved in calculating the contrastive loss computes the cosine similarity through L_2 regularization, then it satisfies $\mathbf{e}_u^\top \mathbf{e}_u = 1$. Thus, Equation (17) can be further expanded:

$$\begin{aligned}\mathcal{L}_{\text{InfoNCE}}^{\text{user}} &= \sum_{u\in\mathcal{B}} -\log\frac{\exp\left(\mathbf{e}_u^\top \mathbf{e}_u/\tau\right)}{\sum_{v\in\mathcal{B}}\exp\left(\mathbf{e}_u^\top \mathbf{e}_v/\tau\right)} \\ &= \sum_{u\in\mathcal{B}} -\log\frac{\exp\left(1/\tau\right)}{\exp\left(1/\tau\right)+\sum_{v\in\mathcal{B},v\neq u}\exp\left(\mathbf{e}_u^\top \mathbf{e}_v/\tau\right)} \\ &= \sum_{u\in\mathcal{B}}\left(-\frac{1}{\tau}+\log\left(\exp\left(\frac{1}{\tau}\right)+\sum_{v\in\mathcal{B},v\neq u}\exp\left(\mathbf{e}_u^\top \mathbf{e}_v/\tau\right)\right)\right).\end{aligned} \tag{18}$$

The item side has a similar definition and derivation. Since τ is a fixed coefficient, Equation (17) actually measures the similarity between nodes of the same type, which leads to a gradual uniformity of nodes within the feature space [28, 47]. A more uniform feature space ensures that cold-start items have more chances to be recommended. While doing so, we continue to preserve the simplicity by avoiding the addition of any new operations or parameters, and we refer to as *embedding-less GCL*. From the perspective of regulating feature distribution, we convert the GCL process into a contrastive regularization [23] based on graph structure to prevent nodes from being excessively similar. Figure 5 presents comparison between traditional GCL and embedding-less GCL.

Running Example. As shown in Figure 5, it is assumed that there is an existing set of users $\{u_1, u_3\}$ and a set of items $\{i_2, i_4, i_5\}$. The traditional GCL based on Equation (15) first requires constructing two additional views of all nodes via data augmentation (e.g., $\mathbf{e}_1'$ and $\mathbf{e}_1''$ for user u_1) and subsequently realizes the contrast of user u_1 and u_3 and the contrast of item i_2 and i_4, respectively. On the contrary, the embedding-less GCL does not perform data augmentation and only utilizes embedding representations obtained through embedding-less GCN to implement the contrast process among five user and item nodes. For instance, user u_1 draws closer to the positive item i_2 with which he interacted (red lines) while simultaneously pushing away from negative items i_4 and i_5 (blue line). In the meantime, item i_2 (user u_1) gradually moves away from with item i_4 and i_5 (user u_3). Finally, we summarize the positive effect of the above contrastive losses on the recommendation task:

—Assisting the main recommendation task further enhances user-interacted item similarity.
—Enhancing the uniformity of the feature space to provide more recommendation opportunities for cold-start users and items.

—Providing more supervision signals for the recommendation task to mitigate data sparsity.
—Providing more gradients for hard-negative samples greatly improves the model convergence speed while enhancing the model robustness.

3.3 Joint Learning

The optimization of EGCF consists of two parts: the recommendation task and the self-supervised task. As shown in Figure 1 in the Introduction, the individual behavior of users has complexity, with some users having similar preferences whereas others are more complex, which makes it necessary to make tradeoffs between alignment and uniformity. Based on this, we optimize the parameters of EGCF through a joint strategy:

$$\mathcal{L}_{\text{EGCF}} = \mathcal{L}_{\text{BPR}} + \lambda_1 \left(\mathcal{L}_{\text{InfoNCE}}^{\text{inter}} + \mathcal{L}_{\text{InfoNCE}}^{\text{user}} + \mathcal{L}_{\text{InfoNCE}}^{\text{item}} \right) + \lambda_2 \left\| \mathbf{E}_I^{(0)} \right\|_2^2, \tag{19}$$

where λ_1 and λ_2 are adjustable strength coefficients, respectively, and $\mathbf{E}_I^{(0)}$ is the initial item embedding table. The complete training process for EGCF is presented in Algorithm 1.

3.4 Space Complexity

Benefiting from the proposed embedding-less GCN, EGCF only needs to initialize the item embeddings $\mathbf{E}_I \in \mathbb{R}^{N \times d}$, where N and d are the number of items and the embedding size, respectively. Compared with the traditional methods, EGCF has a significant advantage in terms of space complexity. Taking the Yelp2018 dataset [33] (containing 31,668 users and 38,048 items) as an example, MF [25] and LightGCN [10] have 4.26 million parameters, while EGCF has only 2.32 million parameters (embedding size $d = 64$).

The space complexity of EGCF also has significant advantages over traditional item-based and generative recommendation methods. Specifically, e.g., FISM [16], traditional item-based CF approaches often require the construction of additional item embeddings to characterize the user, whereas generative models (e.g., Mult-VAE [19], and CVGA [52]) require larger dimensional multilayer perceptrons for encoding and reconstruction. We will provide a detailed comparison in the experimental section.

3.5 Time Complexity

We present the time complexity of EGCF in this section. Specifically, we set the number of nodes and edges of the interaction graph $\mathcal{G}$ to be $|V|$ and $|E|$, respectively. K is the number of layers for GCN encoder, d is the embedding size, g is the number of user groups of IMP-GCN [21], ρ is the edge keep probability of SGL-ED [38], and S represent the node number in a batch. Considering that the time complexity of graph-based recommender systems mainly comes from graph convolution, the time complexity comparison of EGCF with other methods is shown in Table 2.

Obviously, the time complexity of EGCF for GCN encoder is in line with that of LightGCN since it does not require any type of data augmentation or additional operation. Whereas both IMP-GCN and SGL-ED require the construction of additional graph views, SimGCL requires repeated execution of GCN encoding to obtain different views of the same node, which makes the training time of these methods rise exponentially. NCL and VGCL, while maintaining the same time complexity as EGCF for the graph encoder, require additional operations (i.e., clustering) to perform FA. Therefore, the overall time complexity remains higher than EGCF. The time complexity of the EGCF in calculating losses is in line with that of the traditional CL methods, while it is only slightly higher than that of the non-CL methods.

Algorithm 1: The Overall Training Process of EGCF

Input: user–item ineraction matrix $\mathbf{R}$, learning rate, regularization strength λ_2, number of GCN layers K, contrastive loss weight λ_1, and temperature τ;

Output: model parameter $\mathbf{E}_I^{(0)}$;

Initialize parameter $\mathbf{E}_I^{(0)}$;
while EGCF not converge **do**
  Sample a mini-batch of user-item pairs $\mathcal{B}$;
  for $< u, i, j > \in \mathcal{B}$ **do**
    /* embedding-less GCN */
    if Using parallel iteration **then**
      Construct user embedding $\mathbf{e}_u^{(0)}$ via Equation (9) then construct initial embedding $\mathbf{e}^{(0)} = \mathbf{e}_u^{(0)} || \mathbf{e}_i^{(0)}$;
    end if
    for $k = 1, 2, ..., K$ **do**
      if Using parallel iteration **then**
        Update embeddings $\mathbf{e}_u^{(k)}$, $\mathbf{e}_i^{(k)}$, and $\mathbf{e}_j^{(k)}$ via light graph convolution;
      else
        Update embeddings $\mathbf{e}_u^{(k)}$, $\mathbf{e}_i^{(k)}$, and $\mathbf{e}_j^{(k)}$ via Equation (13);
      end if
    end for
    Construct final embeddings $\mathbf{e}_u$, $\mathbf{e}_i$, and $\mathbf{e}_j$ by layer-by-layer sums;
    Calculate BPR loss $\mathcal{L}_{\text{BPR}}$ via Equation (6);
    /* embedding-less GCL */
    Calculate interaction InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}^{\text{inter}}$ via Equation (16);
    Calculate user and item InfoNCE losses $\mathcal{L}_{\text{InfoNCE}}^{\text{user}}$ and $\mathcal{L}_{\text{InfoNCE}}^{\text{item}}$ via Equation (17);
    Calculate joint loss of EGCF $\mathcal{L}_{\text{EGCF}}$ via Equation (25);
    Average gradients from mini-batch;
    Update parameter by descending the gradients $\nabla_{\mathbf{E}_I^{(0)}} \mathcal{L}$;
  end for
end while
return model parameter $\mathbf{E}_I^{(0)}$;

Table 2. Comparisons of Theoretical Time Complexity

Model	GCN Encoder	BPR Loss	CL Loss	GA	FA	MV	AO
NGCF [33]	$O(2(\|E\| + \|V\|)Kd)$	$O(2Bd)$	-	×	×	×	×
LightGCN [10]	$O(2\|E\|Kd)$	$O(2Bd)$	-	×	×	×	×
IMP-GCN [21]	$O(2g\|E\|Kd)$	$O(2Bd)$	-	✓	×	×	×
SGL-ED [38]	$O(2(1+2\rho)\|E\|Kd)$	$O(2Bd)$	$O(Bd + BSd)$	✓	×	✓	×
NCL [20]	$O(2\|E\|Kd)$	$O(2Bd)$	$O(Bd + BSd)$	×	✓	✓	✓
SimGCL [47]	$O(6\|E\|Kd)$	$O(2Bd)$	$O(Bd + BSd)$	×	✓	✓	×
VGCL [44]	$O(2\|E\|Kd)$	$O(2Bd)$	$O(Bd + BSd)$	×	✓	✓	✓
EGCF (ours)	$O(2\|E\|Kd)$	$O(2Bd)$	$O(Bd + BSd)$	×	×	×	×

'GA', 'FA', and 'MV' are shorthand for graph augmentation, feature augmentation, and multi-views, respectively, and 'AO' stands for additional operations, such as clustering.

3.6 Alignment and Uniformity

Some recent works [29, 47] point out that the core of contrastive learning lies in alignment and uniformity. Alignment allows for closer distances between positive pairs, while uniformity promotes a uniform feature distribution on the unit hyper-sphere [30].

Given the distribution of positive pairs $p_{pos}(\cdot,\cdot)$, the alignment is the expected distance between the normalized representations of the positive pairs:

$$\mathcal{L}_{\text{align}} = \mathbb{E}_{(x,x^+)\sim p_{pos}} \left\| f(x) - f(x^+) \right\|^2, \tag{20}$$

where $f(\cdot)$ means L_2 normalization, and x and x^+ are positive pairs of each other. In contrast, uniformity is based on the Gaussian potential kernel:

$$\mathcal{L}_{\text{uniform}} = \log \mathbb{E}_{(x,x^-)\sim p_{data}} \left[\exp(-2 \left\| f(x) - f(x^-) \right\|^2) \right], \tag{21}$$

where $p_{data}(\cdot,\cdot)$ is the distribution of the input data. We then further show the relationship between contrastive loss $\mathcal{L}_{\text{infoNCE}}$ and alignment and uniformity:

$$\mathcal{L}_{\text{infoNCE}} = \mathbb{E}_{\substack{(x,x^+)\sim p_{pos}, \\ (x,x^-)\sim p_{data}}} \left[-\frac{f(x)^\top f(x^+)}{\tau} + \log \left(\exp\left(\frac{f(x)^\top f(x^+)}{\tau} \right) + \sum_{x^-} \exp\left(\frac{(f(x)^\top f(x^-)}{\tau} \right) \right) \right] \tag{22}$$

$$\geq \mathbb{E}_{\substack{(x,x^+)\sim p_{pos}, \\ (x,x^-)\sim p_{data}}} \left[-\frac{1}{\tau} + \log \left(\exp\left(\frac{1}{\tau} \right) + \sum_{x^-} \exp\left(\frac{f(x)^\top f(x^-)}{\tau} \right) \right) \right]. \tag{23}$$

For Equation (22), its lower bound (Equation (23)) holds only if the positive pair (x, x^+) is perfectly aligned. For the lower bound, since the temperature coefficient τ is a fixed constant, the optimization process for perfect alignment is reduced to:

$$\min \mathbb{E}_{\substack{(x,x^+)\sim p_{pos}, \\ (x,x^-)\sim p_{data}}} \left[\log \left(\sum_{x^-} \exp\left(\frac{f(x)^\top f(x^-)}{\tau} \right) \right) \right]. \tag{24}$$

Based on the above analysis, we review the three contrastive losses proposed in embedding-less GCL:

- $\mathcal{L}^{\text{inter}}_{\text{InfoNCE}}$: This loss treats the user u and the interacted item i as positively, and other item j within the batch $\mathcal{B}$ is considered negative sample, so we have $(x, x^+) \leftarrow (u, i)$ and $(x, x^-) \leftarrow (u, j)$. Since the user and the item come from different embeddings, its conforms to the case of Equation (22).
- $\mathcal{L}^{\text{user}}_{\text{InfoNCE}}$: This loss treats the user u himself as positively, and other user v within the batch $\mathcal{B}$ is considered negative sample, so we have $(x, x^+) \leftarrow (u, u)$ and $(x, x^-) \leftarrow (u, v)$. Since users themselves fall into perfect alignment, its conforms to the case of Equation (23).
- $\mathcal{L}^{\text{item}}_{\text{InfoNCE}}$: This loss treats the item i itself as positively, and other item j within the batch $\mathcal{B}$ is considered negative sample, so we have $(x, x^+) \leftarrow (i, i)$ and $(x, x^-) \leftarrow (i, j)$. Since items themselves fall into perfect alignment, its conforms to the case of Equation (23).

The above analysis shows that embedding-less GCL aims to achieve alignment and uniformity between user pairs, item pairs, and interaction pairs. In keeping with the core purpose of recommendation, we force users and positive item to align as much as possible, while keeping the user and item distributions uniformly distributed. It should be noted that some novel works attempt to use alignment (Equation (20)) and uniformity (Equation (21)) directly for recommendation tasks, such as DirectAU [28] and GraphAU [43]. We point out that EGCF is different from them: (1) DirectAU only considers the alignment of interaction pairs and not the uniformity between different interactions;

Table 3. Statistics of the Datasets

Dataset	#Users	#Items	#Interactions	Sparsity
Yelp2018	31,668	38,048	1,561,406	99.87%
Amazon-Book	52,643	91,599	2,984,108	99.94%
Alibaba-iFashion	300,000	81,614	1,607,813	99.99%

(2) EGCF follows the design of conventional GCL-based models, with the temperature coefficient τ to adjust the uniformity across the feature space.

4 Experiments

4.1 Datasets

To verify the validity and generalizability of EGCF, we select three classical real-world large-scale datasets: Yelp2018[2] [10, 38], Amazon-Book[3] [10, 38], and Alibaba-iFashion[4] [38, 47], all of which have been widely used and studied in previous works. The statistical information for three datasets is shown in Table 3. All datasets are divided into training, validation, and testing sets with a ratio of 7:1:2 [10, 47]. Items that the user has interacted with are considered as corresponding positive samples and vice versa will be considered as negative samples.

4.2 Baselines

we choose the following state-of-the-art recommendation methods for comparison experiments:

—*MF* [25]: The method factorizes the interaction matrix into independent user and item embeddings, and uses the inner product directly to obtain the user's prediction score for the item.
—*FISM* [16]: The method is a classical item-based CF approach, which contains two types of item embeddings, one being used to construct user embeddings, which can be viewed as performing a neighborhood information aggregation.
—*Mult-VAE*[5] [19]: The method is a classical item-based CF model, which models the mean and variance of user preferences. The model structure of Mult-VAE is set to [200, 600] as suggested in the source paper. The dropout strategy is used to prevent over-fitting early in training.
—*CVGA* [52]: The method approximates user behavioral preferences as Gaussian distributions through GCN and reconstructs the complete interaction graph through a multi-layer perceptron. The decoder of CVGA uses one layer of neural network with a consistent dimension of the embedding size d.
—*DiffRec*[6] [32]: The method is a novel item-based recommendation model that enhances the robustness of the model by gradually adding Gaussian noise to the interaction information. The model structure of DiffRec is chosen from {[200,600], [1,000]} with a step embedding size of 10. The diffusion and inference steps are {5,40,100} and 0, respectively.

[2]https://github.com/wujcan/SGL-Torch/tree/main/dataset/yelp2018
[3]https://github.com/wujcan/SGL-Torch/tree/main/dataset/amazon-book
[4]https://github.com/wujcan/SGL-Torch/tree/main/dataset/ifashion
[5]https://github.com/dawenl/vae_cf
[6]https://github.com/YiyanXu/DiffRec

—*NGCF*[7] [33]: The method employs a standard GCN design to encode embedding representations of users and items, with additional consideration of user–item affinities. We introduce the message dropout mechanism for NGCF.
—*LightGCN*[8] [10]: The method is a classical GCN-based recommendation model with a series of simplifications of original GCN for CF tasks. The weight decay coefficient is set to $1e^{-4}$ by default.
—*IMP-GCN*[9] [21]: The method is an improved GCN-based recommendation method that divides the users into different sub-graphs based on their interests. The number of GCN layers for IMP-GCN is set to 6, the number of user groups g is set to 3 (2 on the Amazon-Book dataset due to the OOM issues).
—*SGL-WA* [47]: The method is a variant of the SGL method [38], which does not construct data augmentation, i.e., it only measures the uniformity between nodes.
—*SGL-ED*[10] [38]: The method is classical GCL-based recommendation method, which constructs different views of the same node by randomly dropping edges. The setting ranges for the contrastive loss coefficient, temperature, and edge mask probability of SGL-ED are {0.005, 0.01, 0.05, 0.1, 0.5, 1.0}, {0.1, 0.2, 0.5, 1.0}, and {0.1, 0.2, 0.3, 0.4, 0.5}, respectively. The SGL-WA settings are consistent with the SGL-ED.
—*MixGCF*[11] [13]: The method introduces an additional hop mixing technique to LightGCN to synthesize negative samples used to compute the BPR loss. The size of the candidate set is 64.
—*NCL*[12] [20]: The method provides additional supervision signals for CF task by constructing structural and semantic neighbors. The number of structural neighbor layers for NCL is set to 1, the weights for structural- and prototype-contrastive loss are set in the range of $\{1e^{-8}, 1e^{-7}, 1e^{-6}\}$. The weight of structure-contrastive loss on the item side are set in the range of {0.1, 2.0} (step size is 0.1), the temperature coefficients are set in the range of {0.05, 1.0} (step size is 0.025), and the number of clusters is set in the range of {500, 800, 1,000, 1,500, 2,000}.
—*DirectAU*[13] [28]: The method remeasures the CF task via alignment and uniformity to optimize the distribution of users and items in the feature space. The range of adjustments for the weights controlling uniformity loss in DirectAU is {0.2, 0.5, 1.0, 2.0, 5.0}.
—*SimGCL*[14] [47]: The method is a GCL-based recommendation method that achieves data augmentation by adding random noise to user and item embedding representations. According to the authors' suggestions, the temperature coefficient of SimGCL is set to 0.2 by default. The weight of contrastive loss and noise strength are both set in the range of {0.05, 0.1, 0.2, 0.5, 1.0}.
—*GraphAU*[15] [43]: The method tries to measure user–item alignment in each graph convolutional layer and proposes a layer-wise alignment pooling to take into account the semantic distinctions among different layers. According to the description of the article, we manually adjust the weight vector α and the tradeoff parameter γ from 0.1 to 1.0.

[7]https://github.com/xiangwang1223/neural_graph_collaborative_filtering
[8]https://github.com/kuandeng/LightGCN
[9]https://github.com/liufancs/IMP_GCN
[10]https://github.com/wujcan/SGL-Torch
[11]https://github.com/huangtinglin/MixGCF
[12]https://github.com/RUCAIBox/NCL
[13]https://github.com/THUwangcy/DirectAU
[14]https://github.com/Coder-Yu/SELFRec
[15]https://github.com/YangLiangwei/GraphAU

— *CGCL*[16] [9]: This is a novel GCL-based recommendation method that considers intermediate embeddings of different GCN layers to act as contrastive views. The three contrastive loss weights of the CGCL are searched in $\{1e^{-8}, 1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$, and we set the temperature coefficient $\tau = 0.1$ by default.
— *DCCF*[17] [24]: The method is an intent- and contrastive learning-based recommendation method which learns the intent prototypes of users and items and constructs different perspectives of the original interaction graph by learnable GA. The settings for the dimensions of the intent embedding are in the range of [32, 64, 128, 256], the temperature coefficient is adjusted in the range of [0.1, 0.2, 0.5, 1.0], and the weights of the intent regularization loss and contrastive loss are adjusted in the ranges of $[2.5e^{-5}, 5e^{-4}, 5e^{-3}]$ and [0.001, 0.025, 0.1, 0.2] respectively.
— *VGCL*[18] [44]: The method is a novel GCL-based recommendation method that learns user distributions through graph-based variational inference and constructs contrastive views using reparameterization trick and clustering. The weight of contrastive loss is set in the range of {0.01, 0.05, 0.1, 0.2, 0.5, 1.0}. The weight of cluster-level contrastive loss is set in the range of {0.0, 1.0} (step size is 0.1). The temperature coefficients are set in the range of {0.10, 0.25} (step size is 0.01). For the number of clusters of users and items, we use grid search to determine the optimal pair of values within the range of {200, 400, 600, 800, 1,000}.

4.3 Hyper-Parameters

We implement EGCF by PyTorch.[19] For fair comparisons, the embedding size is set to 64 for all models (except Mult-VAE [19] and DiffRec [32]). The default learning rate and regularization strength are set to 0.001 and 0.0001, respectively. All models are initialized by Xavier [7] and optimized by Adam [17]. The batch size for all datasets is fixed at 2,048. For the GCN- and GCL-based recommendation methods, the number of GCN layers is set to 3, and the GCN encoder is selected as LightGCN [10] by default. For all comparison methods, all experimental results are obtained by running the source code published by the authors. We determine the range and value of each hyper-parameter based on suggestions of original papers and the grid search, which can be referred to Section 4.2. The default settings of EGCF (learning rate, regularization strength, number of GCN layers) are all consistent with LightGCN. The weight of contrastive loss λ_1 and temperature coefficient τ used for embedding-less GCL are both set in the ranges of {0.05, 0.1, 0.2, 0.3, 0.4}.

4.4 Evaluation Protocols

To validate the performance of EGCF in top-N recommendation task and to provide a fair comparison with existing works, we adopt the widely used all-ranking strategy [10, 33]. Specifically, for each training period, the recommender system needs to calculate the prediction scores of each user for all items. Before testing, the scores of interacted items need to be filtered out first, and subsequently, the top-N items with the highest scores among the remaining uninteracted items are selected for recommendation. We measure the performance of all recommendation models using Recall@20 and NDCG@20 [33, 52]. Recall indicates the proportion of recommended items in the testing set, while NDCG additionally takes into account the position of correctly predicted items in the recommendation list, with the higher ranked items having higher scores [49].

[16] https://github.com/WeiHeCnSH/CGCL-Pytorch-master
[17] https://github.com/HKUDS/DCCF
[18] https://github.com/yimutianyang/SIGIR23-VGCL
[19] https://github.com/BlueGhostYi/ID-GRec

Table 4. Overall Performance Comparisons on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

	Yelp2018		Amazon-Book		Alibaba-iFashion	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF [25]	0.0539	0.0439	0.0308	0.0239	0.0884	0.0412
FISM [16]	0.0559	0.0459	0.0401	0.0309	0.0925	0.0436
Mult-VAE [21]	0.0584	0.0450	0.0407	0.0315	0.1041	0.0497
CVGA [52]	0.0694	0.0571	0.0492	0.0379	0.1132	0.0541
DiffRec [32]	0.0665	0.0556	0.0514	0.0418	0.0767	0.0398
NGCF [33]	0.0560	0.0456	0.0342	0.0261	0.1043	0.0486
LightGCN [10]	0.0639	0.0525	0.0411	0.0315	0.1078	0.0507
IMP-GCN [21]	0.0653	0.0531	0.0460	0.0357	N/A	N/A
MixGCF [13]	0.0713	0.0589	0.0485	0.0378	0.1085	0.0520
SGL-WA [47]	0.0671	0.0550	0.0466	0.0373	0.1065	0.0519
SGL-ED [38]	0.0675	0.0555	0.0478	0.0379	0.1126	0.0538
NCL [20]	0.0685	0.0577	0.0481	0.0373	0.1131	0.0547
DirectAU [28]	0.0703	0.0583	0.0506	0.0406	0.1182	0.0571
SimGCL [47]	0.0721	0.0601	0.0515	0.0414	0.1151	0.0567
GraphAU [43]	0.0691	0.0574	0.0508	0.0403	0.1175	0.0563
DCCF [24]	0.0661	0.0540	N/A	N/A	N/A	N/A
CGCL [9]	0.0694	0.0561	0.0482	0.0375	0.1101	0.0523
VGCL [44]	0.0715	0.0587	0.0506	0.0401	0.1137	0.0541
EGCF (ours)	**0.0748***	**0.0617***	**0.0540***	**0.0431***	**0.1222***	**0.0593***
p-Values	6.06e-9	3.74e-8	1.61e-7	1.39e-6	1.71e-9	2.07e-7

The results of EGCF are bolded, whereas the best baseline is underlined. "N/A" stands for failure to converge within an acceptable time.
*Denotes that the improvement is significant with a p-value < 0.001 based on a two-tailed paired t-test.

4.5 Performance Comparisons

4.5.1 Overall Comparisons. Table 4 shows the performance of EGCF and all comparison methods on three datasets, and we have the following observations.

EGCF achieves the best performance over all baselines on all datasets. Quantitatively, EGCF improves over the best baselines w.r.t. Recall@20 by 3.75%, 4.85%, and 3.38% on Yelp2018, Amazon-Book, and iFashion datasets, respectively. The results are indicative that EGCF can achieve competitive performance while exhibiting strong generalizability.

Focusing on all item-based methods, FISM significantly outperforms MF, which demonstrates the advantages of item-based CF in modeling user preferences, but both FISM and Mult-VAE underperform LightGCN, which demonstrates the necessity for graph structure modeling. Although DiffRec achieves better performance than Mult-VAE and LightGCN by introducing diffusion model, it is still limited by sparse interaction data.

Compared with traditional GCN-based recommender methods, GCL-based recommender methods all show significant performance improvements, mainly due to the fact that contrast learning can provide additional supervised signals for the recommendation task, which is especially critical for recommender systems with extremely sparse interactions. Although novel approaches such as SimGCL, CGCL, and VGCL achieve better performance, they are still lower than EGCF. We argue that the core reason for this is that they only consider the contrastive process between users (items), ignoring the fact that the core of the recommendation task is CF, which measures the similarity between users and items.

Table 5. Performance Comparisons between DCCF and EGCF on Gowalla, Amazon-Book, and Tmall Datasets

	Gowalla		Amazon-Book		Tmall	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
DCCF [24]	0.1876	0.1123	0.0889	0.0680	0.0668	0.0469
EGCF	**0.1977**	**0.1208**	**0.0998**	**0.0780**	**0.0768**	**0.0546**

The results of EGCF are bolded.

Table 6. Comparison of the Number of Trainable Parameters (in Millions) on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

	MF	FISM	Mult-VAE	CVGA	DiffRec	NGCF	**EGCF**
Yelp2018	4.26M	4.64M	43.89M	6.97M	72.62M	4.28M	**2.32M**
Amazon-Book	8.80M	11.18M	105.20M	16.77M	174.81M	9.07M	**5.59M**
Alibaba-iFashion	23.29M	9.96M	68.82M	14.94M	155.76M	23.30M	**4.98M**
	LightGCN	IMP-GCN	MixGCF	SGL-WA	SGL-ED	NCL	**EGCF**
Yelp2018	4.26M	4.26M	4.26M	4.26M	4.26M	4.26M	**2.32M**
Amazon-Book	8.80M	8.81M	8.80M	8.80M	8.80M	8.80M	**5.59M**
Alibaba-iFashion	23.29M	-	23.29M	23.29M	23.29M	23.29M	**4.98M**
	DirectAU	SimGCL	GraphAU	CGCL	VGCL	DCCF	**EGCF**
Yelp2018	4.26M	4.26M	4.26M	4.26M	4.26M	4.27M	**2.32M**
Amazon-Book	8.80M	8.80M	8.80M	8.80M	8.81M	-	**5.59M**
Alibaba-iFashion	23.29M	23.29M	23.29M	23.29M	23.30M	-	**4.98M**

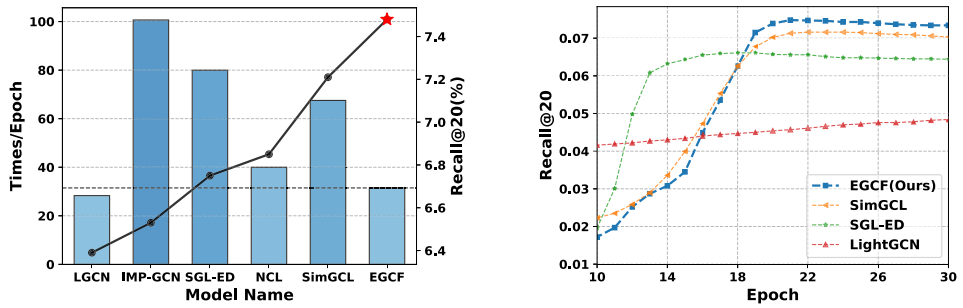
The results of EGCF are bolded.

Compared with SGL-WA, directAU, and GraphAU that also do not use data augmentation, EGCF shows strong competitiveness, which demonstrates the necessity of measuring the alignment and uniformity of users, items, and user–item pairs. The performance of EGCF compared with traditional item-based CF methods (e.g., Mult-VAE and DiffRec) demonstrates the effectiveness of graph structure modeling with SSL.

In addition, due to the high complexity of DCCF, DCCF suffers from the out-of-memory problem on Amazon-Book and Alibaba-iFashion datasets when embedding size d is set to 64 on a NVIDIA A100 40 G. Therefore, we apply EGCF to the three datasets (Gowalla, Amazon-Book, and Tmall) used in the DCCF paper [24], and keep the same settings for EGCF (embedding size $d = 32$ and number of GCN layers $K = 2$), and the experimental results are shown in Table 5. In this comparison, EGCF still demonstrates performance improvement, which further shows the effectiveness of EGCF.

4.5.2 Comparisons w.r.t. Simplicity. Another aspect to point out is the simplicity of EGCF, which includes the space complexity (Section 3.4) and time complexity (Section 3.5) that we analyzed previously.

For space complexity, Table 6 presents the comparison of the number of trainable parameters (in millions) of EGCF with all comparison methods on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets. The number of parameters of EGCF is much lower than those of all baseline methods on three datasets. All of item-based CF methods achieve better performance than MF, and subsequent studies (e.g., CVGA and DiffRec) even outperform graph recommendation methods such as LightGCN [10]. These studies show the rationality and feasibility of removing user embedding.



(a) training speed (b) convergence speed

Fig. 6. (a) Comparisons of training speed (left y-axis) and best Recall@20 (right y-axis) for EGCF and other methods. (b) Training curves of EGCF and other comparison methods (10–30 epochs). LGCN, LightGCN.

Unfortunately, these item-based CF methods need to consider more dimensions for item modeling, which significantly increases the number of model parameters. It should be noted, however, that EGCF has a completely different design philosophy from these approaches. Embedding-less GCN utilizes the available interaction graph information so that user embeddings can be constructed with alternate information, which is one of the advantages of the minimalist nature of EGCF. In addition, fewer parameters are seen as a parameter reuse trick, which significantly reduces the training burden and does not require the introduction of additional techniques to prevent over-fitting.

For time complexity, we also present comparisons of the training and convergence speeds of EGCF on Yelp2018 dataset, as shown in Figure 6. Benefiting from less-embedding GCN and less-embedding GCL, the training time of EGCF is much lower than those of traditional GCL-based recommendation methods, while only a little higher than that of LightGCN (requires additional calculation of contrastive losses), and model convergence can be accomplished with a number of iterations on par with the level of methods such as SimGCL [47].

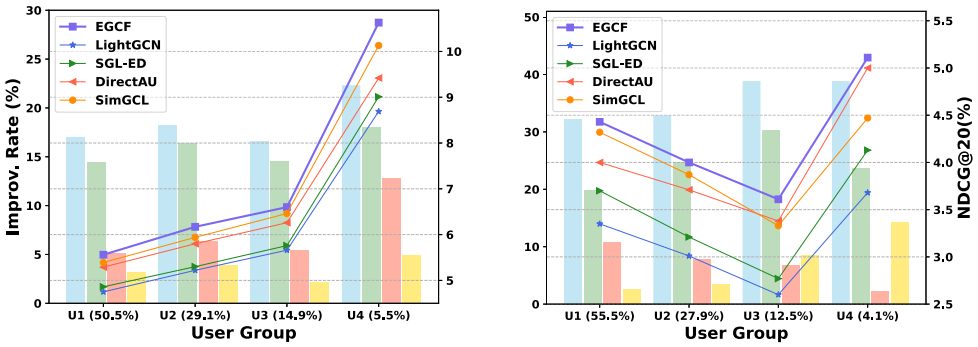
4.5.3 Comparisons w.r.t. GCN Layers. For graph structure-based recommendation methods, studying the number of stacked graph convolution layers is one way to verify whether the graph structure can be effectively utilized. Based on this, Table 7 provides comparisons of the effectiveness of EGCF, LightGCN, and SimGCL with various GCN layer settings on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets. As the number of layers increases, the performance of both EGCF and the other compared methods begins to rise, suggesting that EGCF can effectively model graph structure. Overall, EGCF achieves excellent performance in all settings. And it should be noted that EGCF with only one layer outperforms even SGL-ED and SimGCL with three layers, which shows that EGCF can more effectively mine user–item relationships without over-reliance on high-order graph structures. As shown in Table 2, stacking more graph convolutional layers significantly increases the time complexity, while EGCF can achieve comparable performance with fewer layers [21, 52]. Therefore, in the same situation, EGCF can achieve the same performance with less training time consumption.

4.5.4 Comparisons w.r.t. Data Sparsity. To verify the recommendation performance of EGCF in different sparse situations, we divide the datasets into four equal-sized user groups based on the number of user interactions and test them separately [33, 52]. Specifically, all users are categorized into four user groups U1, U2, U3, and U4 based on sparsity (i.e., number of user interactions). Let

Table 7. Performance Comparisons between EGCF and Other Baseline Methods w.r.t. the Number of GCN Layer K on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

#Layers	Method	Yelp2018		Amazon-Book		Alibaba-iFashion	
		Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
$K = 1$	LightGCN [10]	0.0631	0.0515	0.0384	0.0298	0.0990	0.0454
	SGL-ED [38]	0.0637	0.0526	0.0451	0.0353	0.1125	0.0536
	SimGCL [47]	0.0689	0.0572	0.0453	0.0358	0.1036	0.0505
	EGCF	**0.0738**	**0.0608**	**0.0529**	**0.0419**	**0.1208**	**0.0588**
$K = 2$	LightGCN [10]	0.0622	0.0504	0.0411	0.0315	0.1066	0.0505
	SGL-ED [38]	0.0668	0.0549	0.0468	0.0371	0.1091	0.0520
	SimGCL [47]	0.0719	0.0601	0.0507	0.0405	0.1119	0.0548
	EGCF	**0.0743**	**0.0612**	**0.0534**	**0.0423**	**0.1222**	**0.0592**
$K = 3$	LightGCN [10]	0.0639	0.0525	0.0410	0.0318	0.1078	0.0507
	SGL-ED [38]	0.0675	0.0555	0.0478	0.0379	0.1139	0.0539
	SimGCL [47]	0.0721	0.0601	0.0515	0.0414	0.1151	0.0567
	EGCF	**0.0748**	**0.0617**	**0.0540**	**0.0431**	**0.1222**	**0.0593**

The results of EGCF are bolded.



(a) Yelp2018 (b) Amazon-Book

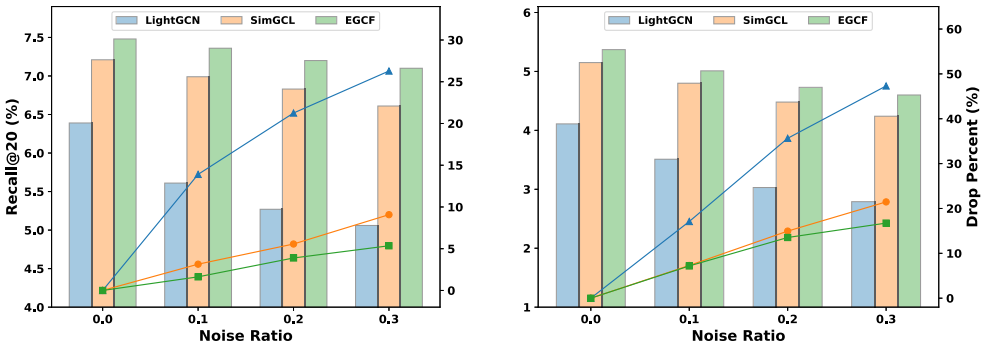
Fig. 7. Sparsity tests on (a) Yelp2018 and (b) Amazon-Book datasets. Relative improvement rates (left y-axis, bar graph) between EGCF and each of baselines. Performance curves (right y-axis, line graph) for each model with different sparsities. The x-axis shows the different user groups and the corresponding proportions. The color of the bar of any model in the bar graph corresponds to the color of the same model in the line graph.

$s(\mathrm{U})$ be the sparsity of the user group U, then we have:

$$s(\mathrm{U1}) > s(\mathrm{U2}) > s(\mathrm{U3}) > s(\mathrm{U4}). \tag{25}$$

The proportion of user groups segmented indicates the sparsity of the dataset. On Yelp2018 dataset, for example, 50.5% of users are classified into the sparsest group U1, while only 5.5% are classified into U4, which indicates the extreme skewness and sparseness of the data distribution. During the training phase, all users are trained using the complete training set, whereas they are tested in groups based on sparsity during testing.

The experimental results are shown in Figure 7. EGCF achieves the best performance in all sparse groups across all datasets, demonstrating the effectiveness of embedding-less GCN and embedding-less GCL for improving user and item distributions in the feature space. By aligning



(a) Yelp2018 (b) Amazon-Book

Fig. 8. Performance comparisons (left y-axis, bar graph) w.r.t. Recall@20 and percentage of performance degradation (right y-axis, line graph) for different noise ratios on (a) Yelp2018 and (b) Amazon-Book datasets. The color of the bar of any model in the bar graph corresponds to the color of the same model in the line graph.

and uniforming users and items within the same feature space, items with different sparsity levels have the opportunity to be recommended to users, thus effectively alleviating the popularity bias problem [36]. Note that performance degradation occurred for all methods on groups U2 and U3 on the Amazon-Book dataset, the possible reason for this is that these clusters contain too many false-positive samples [31], which interferes with model training. Although performance degradation occurred for all models, EGCF still achieves the best performance, which demonstrates the stability and consistency of EGCF's performance improvement.

4.5.5 Comparisons w.r.t. Noise Interactions. Contrastive learning can effectively extract useful CF patterns through the contrast process between different views, which makes it less susceptible to noisy data [38]. The proposed embedding-less GCL in EGCF does not include data augmentation with MV construction, which raises concerns about the performance of EGCF, i.e., whether the proposed non-augmentation strategy is equally resistant to the effects of noise. To investigate the robustness of EGCF, we inject a certain percentage of random noise (10%, 20%, and 30%) into the training set on Yelp2018 and Amazon-book datasets while keeping the testing set unchanged. To ensure fair comparisons, all comparison methods are trained using the same training set with random noise and subsequently tested on the clean testing set.

Figure 8 presents the experimental results of LightGCN, SimGCL, and EGCF on Yelp2018 and Amazon-Book datasets. Obviously, by injecting more noisy data, the performance of all models starts to degrade. However, the performance of EGCF decreases at a lower rate than those of LightGCN and SimGCL, and the gap among the decrease curves is more obvious at larger noise (30%), which demonstrates the excellent ability of EGCF to resist noise. This shows that the non-augmented embedding-less GCL employed by EGCF also has good noise resistance. As research from previous work [47] suggests, the essence of the improvement of GCL for recommendation performance lies in regulating the uniform distribution of the feature space, rather than relying on manually designed data augmentation. Therefore, EGCF adopts the same idea and directly measures the alignment and uniformity between user pairs, item pairs and user–item pairs. Experimental results show that this contrastive strategy is equally effective and more efficient in resisting noise.

Table 8. Ablation Studies of EGCF on Yelp2018 and Amazon-Book Datasets

Type	Model	Yelp2018		Amazon-Book	
		Recall@20	NDCG@20	Recall@20	NDCG@20
Embedding-less GCN	MF [25]	0.0539	0.0439	0.0308	0.0239
	w/o user embedding	**0.0642**	**0.0532**	**0.0380**	**0.0302**
	NGCF [33]	0.0560	0.0456	0.0342	0.0261
	w/o user embedding	**0.0576**	**0.0470**	**0.0355**	**0.0271**
	LightGCN	0.0639	0.0525	0.0411	0.0315
	w/o user embedding	**0.0671**	**0.0552**	**0.0420**	**0.0325**
Embedding-less GCL	w/ MF	0.0628	0.0516	0.0497	0.0388
	w/ NGCF	0.0583	0.0475	0.0423	0.0328
	w/ LightGCN	0.0728	0.0602	0.0511	0.0409
	w/o $\mathcal{L}_{\text{InfoNCE}}^{\text{align}}$	0.0724	0.0600	0.0510	0.0404
	w/o $\mathcal{L}_{\text{InfoNCE}}^{\text{user}}$	0.0741	0.0610	0.0521	0.0411
	w/o $\mathcal{L}_{\text{InfoNCE}}^{\text{item}}$	0.0717	0.0593	0.0441	0.0351
	EGCF	**0.0748**	**0.0617**	**0.0540**	**0.0431**

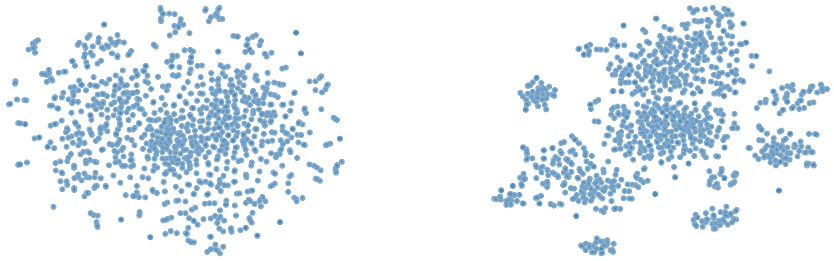
The best-performing models are shown in bold.

4.6 In-Depth Studies of EGCF

4.6.1 Ablation Studies. We first construct a series of ablation experiments to investigate whether each part of EGCF is effective or not, and the experimental results are shown in Table 8.

Focusing on embedding-less GCN, we investigate the effectiveness of removing user embeddings for GCN encoding. Specifically, we apply the design idea of embedding-less GCN to the classical MF, NGCF, and LightGCN methods, respectively. For MF, we first construct user embeddings via Equation (9) and subsequently compute the prediction scores directly via inner product; for NGCF and LightGCN, we update the user and item embeddings by parallel and alternating iterations proposed in Section 3.1 and obtain the final user and item embeddings by concatenation and mean operations. It can be clearly observed that removing the user-side embedding by embedding-less GCN improves the performance of all variants to some extent over the base model. This shows the rationality and generalization of embedding-less GCN. We argue that the core of the performance improvement lies in the fact that after removing the embedding update on the user side, the gradient update based on the BPR loss will only take into account the similarity between the item embeddings (in particular, the set of neighboring items $\{\mathbf{e}_{i'}|i' \in \mathcal{N}_u\}$ of user u), which makes the embedding-less GCN easier to be trained. In addition, we replace the embedding-less GCN with LightGCN while keeping the embedding-less GCL unchanged, named $\text{EGCF}_{\text{w/ LightGCN}}$, and we find that there is a certain degree of performance degradation in $\text{EGCF}_{\text{w/ LightGCN}}$, which also shows the effectiveness of embedding-less GCN.

Focusing on embedding-less GCL, we also apply the design idea of embedding-less GCL to the classical MF, NGCF, and LightGCN methods. Specifically, we keep the original MF, NGCF, and LightGCN unchanged, add embedding-less GCL to their original training objectives, and control the strength of embedding-less GCL by parameter λ_1. Obviously, the introduction of embedding-less GCL resulted in a significant improvement in the recommendation performance for all methods, which validates the effectiveness of embedding-less GCL. We attribute the performance improvement to the power of contrastive learning, which not only mitigates the data sparsity problem of recommendation by additional supervised signals but also solves the over-aggregation



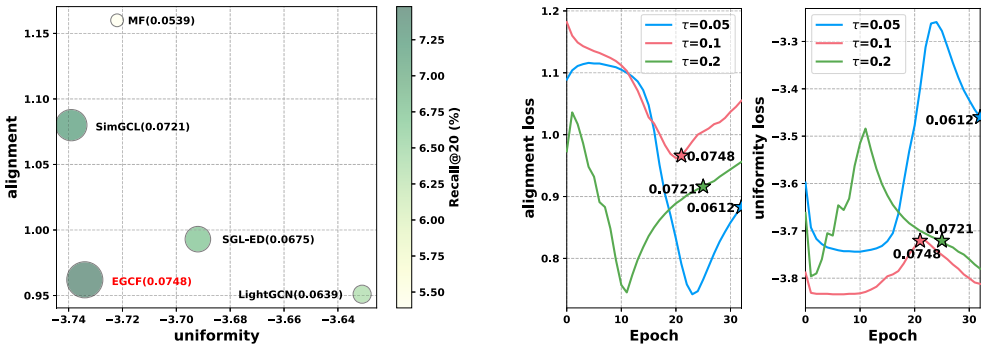
(a) User embeddings (b) Item embeddings

Fig. 9. t-SNE visualization results of randomly sampled (a) user and (b) item embeddings from well-trained EGCF on Yelp2018 dataset.

problem in recommendation by regulating uniformity. Besides, the removal of each of three self-supervised losses results in varying degrees of degradation in the performance of EGCF, indicating the necessity of modeling the alignment and uniformity for users, items, and user–item pairs. We also find that the effect of item-side loss $\mathcal{L}^{\text{item}}_{\text{InfoNCE}}$ on performance is even more significant, suggesting that EGCF is sensitive to the uniformity among items. This is reasonable because the trainable parameters for EGCF are only item embeddings $\mathbf{E}^{(0)}_I$, and thus uniformity of item embeddings will be more important than ever.

4.6.2 Impact of Alignment and Uniformity. As analyzed in Section 3.6, the core of contrastive learning lies in alignment and uniformity. Therefore, this section investigates whether EGCF has well-alignment and uniformity. First, uniformity is crucial to mitigate the popularity bias in recommender systems, which can provide users with more opportunities to focus on those cold-start items. Figure 9 presents the visualization results of EGCF on the Yelp2018 dataset. As a counterpoint to Figure 4, we similarly randomly select 1,000 embeddings of users and items and project them to the 2D surface using t-SNE. Obviously, the clustering effect of users and items is significantly mitigated, explicitly more evident on the user side. We attribute this to the proposed embedding-less GCN and embedding-less GCL. On the one hand, after removing the redundant initial user embedding, the user relies more on the information of the multi-order neighboring items and thus exhibits the extreme uniformity property; on the other hand, the item embedding, as the only trainable parameter, maintains a well-balanced between uniformity and clustering.

In addition, we further analyze the alignment and uniformity of EGCF as a counterpart to the analysis in Section 3.6. Figure 10(a) presents a comparison of the uniformity and alignment of EGCF with other classical methods. As pointed out in previous work, LightGCN further exacerbates the over-aggregation effect through neighbor aggregation, which causes LightGCN to have the smallest alignment loss as well as the largest uniformity loss. MF, on the other hand, leads to the largest alignment loss because of its inability to effectively model user–item correlations. SGL-ED and SimGCL mitigate the problems existing in MF and LightGCN to some extent by introducing comparative learning. However, as we have pointed out, the core of recommendation lies in measuring the user–item similarity, which is the part that is neglected by GCL-based methods such as SGL and SimGCL. In EGCF, we further reduce the alignment loss through direct contrast between users and items as well as perfect alignment on the user and item side, while achieving better tradeoff between alignment and uniformity. Finally, we present the curves of alignment and uniformity loss with training iterations for EGCF with different temperature coefficient τ on Yelp2018 dataset, as shown in Figure 10(b). It can be intuitively seen that the alignment and uniformity losses have opposite trends and are always in check with each other. The temperature



(a) Comparison of uniformity-alignment

(b) Analysis of uniformity and alignment

Fig. 10. (a) Comparisons of alignment (left y-axis), uniformity (x-axis), and Recall@20 (right y-axis) for EGCF and other methods; (b) Impact of temperature for alignment and uniformity on Yelp2018 dataset. Colored stars indicate the epoch of optimal performance (Recall@20).

Table 9. Performance Comparisons between EGCF-User and EGCF-Item on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

	Yelp2018		Amazon-Book		Alibaba-iFashion	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
EGCF-user	0.0729	0.0603	0.0516	0.0410	0.1203	0.0580
EGCF-item	**0.0748**	**0.0617**	**0.0540**	**0.0431**	**0.1222**	**0.0593**

The best-performing model is shown in bold.

coefficient τ is crucial for regulating the alignment and uniformity. It is further shown through experimental results that too low alignment and uniformity losses do not necessarily indicate better recommendation performance.

4.6.3 Impact of Initial Embedding. For embedding-less GCN, we remove the initial user embedding through neighbor information aggregation. Without considering the actual meaning, user and item embeddings can be considered as equivalent entities. Both initializing user embeddings and initializing item embeddings have the same impact on recommendation performance. This view seems intuitive since the interaction graph involved in information propagation is fixed and the user and item embeddings will converge gradually given the smoothing properties of GCNs. Specifically, we refer to the model that uses only the initial user embedding as *EGCF-user*, and the model that uses only the initial item embedding as *EGCF-item* (i.e., the model given in the paper). All parameter settings remain the same for both models except for the different initial embeddings. Table 9 presents the experimental results of the two models on the Yelp2018, Amazon-Book, and iFashion datasets. Surprisingly, the experimental results are completely opposite to our previous assumptions, and initializing items seems to be a better choice. For the above results, we have the following analysis.

On both Yelp2018 and Amazon-book datasets, the number of items exceeds the number of users, which allows for initializing item embeddings to encode more valid information. For the Alibaba-iFashion dataset, although the number of users far exceeds the number of items, we argue that the extremely sparse nature of Alibaba-iFashion (99.99%) severely affects the learning process for these users. Initializing too many user embeddings may make it difficult to learn effective collaborative

Table 10. Performance Comparisons between EGCF-Alternate and EGCF-Parallel on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

	Yelp2018		Amazon-Book		Alibaba-iFashion	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
EGCF-alternate	0.0744	0.0616	0.0525	0.0417	**0.1222**	**0.0593**
EGCF-parallel	**0.0748**	**0.0617**	**0.0540**	**0.0431**	0.1220	0.0585

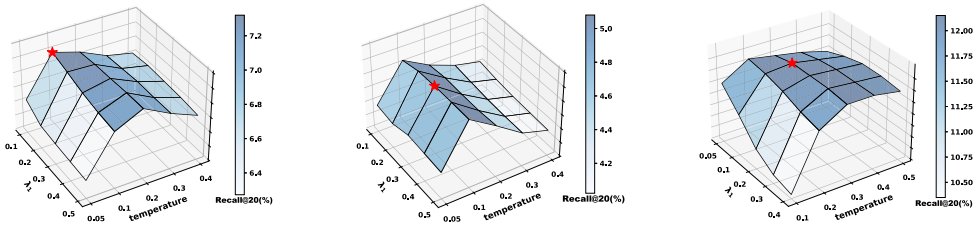
The best-performing model is shown in bold.

signals, resulting in sub-optimal performance. Besides, initialized item embeddings align better with the real world if realistic semantics are considered. Items are objective entities with no subjectivity, whereas human behavior can be reflected through items. Finally, the essence of a recommender system is that the user and the items he interacts with are as close as possible in the feature space. As shown in Figure 1, through model training and loss optimization, most of the items that the user has interacted with are clustered near the user node, showing local clustering. Based on this, we believe that it is feasible to construct an embedding representation of the user by neighboring item aggregation because this process is accelerated to force the interacted items to be closer to the central user in the feature space, thus significantly improving the training speed.

4.6.4 Impact of Iteration Method. Many previous works have demonstrated that stacking multiple graph convolutional layers can be effective in exploring high-order connectivity and thus significantly improve recommendation performance. However, directly performing graph convolution like LightGCN is not feasible for EGCF as there is no initial user embedding. Based on this, we propose two solutions, parallel iteration (Equation (12)) and alternating iteration (Equation (14)), respectively. In order to specifically investigate the impact of these two iteration methods on the performance of EGCF, we use them to conduct experiments on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets, with the corresponding models named EGCF-parallel and EGCF-alternate, respectively.

The experimental results are shown in Table 10. EGCF-alternate and EGCF-parallel exhibit similar performance on the three datasets, which is also consistent with our hypothesis. Specifically, the two methods are largely consistent on the Yelp2018 and Alibaba-iFashion datasets, while there are minor differences on the Amazon-Book dataset, which we attribute to the structural variability of the datasets themselves. In addition, combined with the complete performance comparison results, we observe that both methods outperform all compared baseline methods, which demonstrates the rationality and effectiveness of the EGCF as well.

4.6.5 Hyper-Parameter Sensitivities. Compared with LightGCN, EGCF introduces only two additional parameters: contrastive loss weight λ_1 and temperature coefficient τ, which makes the number of hyper-parameters in EGCF lower than most of the current leading GCL-based methods (e.g., SGL, SimGCL, and VGCL). To show the effect of these two parameters on EGCF in detail, we present the grid search results on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets, as shown in Figure 11. Consistent with the studies in previous works [38, 47], the temperature coefficient is very sensitive to GCL, which can effectively regulate the uniformity of the feature space [28, 29]. Besides, too large weights will interfere with the optimization process of the main task, which will lead to an increase in the number of training iterations and affect performance [38]. For EGCF, the optimal settings on Yelp2018, Amazon-Book, and Alibaba-iFashion datasets are $\{\lambda_1 = 0.1, \tau = 0.1\}$, $\{\lambda_1 = 0.3, \tau = 0.1\}$, and $\{\lambda_1 = 0.05, \tau = 0.2\}$, respectively. Table 11 shows the optimal parameter settings for EGCF on three datasets.



(a) Yelp2018 (b) Amazon-Book (c) Alibaba-iFashion

Fig. 11. Grid searches of impact on contrastive loss weight λ_1 and temperature τ of EGCF on (a) Yelp2018, (b) Amazon-Book, and (c) Alibaba-iFashion datasets. Red star represents optimal performance.

Table 11. Optimal Hyper-Parameter Settings of EGCF on Yelp2018, Amazon-Book, and Alibaba-iFashion Datasets

Dataset	$\|B\|$	d	η	K	τ	λ_1	λ_2
Yelp2018	2,048	64	0.001	3	0.1	0.1	0.0001
Amazon-Book	2,048	64	0.001	3	0.1	0.3	0.0001
Alibaba-iFashion	2,048	64	0.001	3	0.2	0.05	0.0001

$|B|$, batch size; d, embedding size; η, learning rate; K, the number of GCN layers; τ, temperature coefficient; λ_1, weight of contrastive loss; λ_2, weight of regularization.

5 Conclusion and Future Work

In this article, we revisited GCN- and GCL-based recommendation paradigms and proposed a novel end-to-end recommendation framework called EGCF, which contains the embedding-less GCN and the embedding-less GCL. To better adapt to the CF task, we removed redundant user embeddings via graph convolutional aggregation and proposed three non-data-augmented graph contrastive strategies to measure the alignment and uniformity of users, items, and user–item pairs. Compared with previous works, EGCF further streamlines the model design and does not require additional computational cost or parameter settings, which makes it perform well and easier to be trained. We conducted extensive experiments on three public datasets and validated the advantages of EGCF in terms of performance, training efficiency, number of parameters, sparsity resistance, and robustness. We believe that the extremely streamlined nature of the EGCF is well suited for many downstream tasks. In the future, we would like to further explore the design idea of EGCF and expect to introduce it into highly complex application scenarios such as social network, knowledge graph or multi-modal recommendations.

References

[1] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Transactions on Information Systems* 38, 2 (2020), 1–28.

[2] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, 27–34.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning*. PMLR, 1597–1607.

[4] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten De Rijke. 2019. Joint neural collaborative filtering for recommender systems. *ACM Transactions on Information Systems* 37, 4 (2019), 1–30.

[5] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S. Sheng. 2023. Frequency enhanced hybrid attention network for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 78–88.

[6] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 112–121.

[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[8] Yongjing Hao, Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Guanfeng Liu, and Xiaofang Zhou. 2023. Feature-level deeper self-attention network with contrastive learning for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2023), 10112–10124.

[9] Wei He, Guohao Sun, Jinhu Lu, and Xiu Susie Fang. 2023. Candidate-aware graph contrastive learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1670–1679.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web,* 173–182.

[12] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*, 193–201.

[13] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 665–674.

[14] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4252–4261.

[15] Xinzhou Jin, Jintang Li, Yuanzhen Xie, Liang Chen, Beibei Kong, Lei Cheng, Bo Hu, Zang Li, and Zibin Zheng. 2023. Enhancing graph collaborative filtering via neighborhood structure embedding. In *Proceedings of the 2023 IEEE International Conference on Data Mining (ICDM '23)*. IEEE, 190–199.

[16] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-N recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 659–667.

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980

[18] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations*.

[19] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference,* 689–698.

[20] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022,* 2320–2329.

[21] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware message-passing GCN for recommendation. In *Proceedings of the Web Conference 2021,* 1296–1305.

[22] Meng Liu, Jianjun Li, Guohui Li, and Peng Pan. 2020. Cross domain recommendation via bi-directional transfer graph collaborative filtering networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management,* 885–894.

[23] Xiuyuan Qin, Huanhuan Yuan, Pengpeng Zhao, Junhua Fang, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor Sheng. 2023. Meta-optimized contrastive learning for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 89–98.

[24] Xubin Ren, Lianghao Xia, Jiashu Zhao, Dawei Yin, and Chao Huang. 2023. Disentangled contrastive collaborative filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 1137–1146.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence,* 452–461.

[26] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor interaction aware graph convolution networks for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* 1289–1298.

[27] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.

[28] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* 1816–1825.

[29] Feng Wang and Huaping Liu. 2021. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2495–2504.

[30] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of International Conference on Machine Learning*. PMLR, 9929–9939.

[31] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 373–381.

[32] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 832–841.

[33] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval,* 165–174.

[34] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*, 878–887.

[35] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR Conference on Research and Development in Information Retrieval*, 1001–1010.

[36] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1791–1800.

[37] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. 2023. Tree-rings watermarks: Invisible fingerprints for diffusion images. In *Advances in Neural Information Processing Systems*, Vol. 36, 58047–58063.

[38] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 726–735.

[39] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. 2023. Graph-less collaborative filtering. In *Proceedings of the ACM Web Conference 2023,* 17–27.

[40] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 70–79.

[41] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 2 (2022), 2412–2429.

[42] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-N recommendation. *ACM Transactions on Information Systems* 37, 3 (2019), 1–25.

[43] Liangwei Yang, Zhiwei Liu, Chen Wang, Mingdai Yang, Xiaolong Liu, Jing Ma, and Philip S. Yu. 2023. Graph-based alignment and uniformity for recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management,* 4395–4399.

[44] Yonghui Yang, Zhengwei Wu, Le Wu, Kun Zhang, Richang Hong, Zhiqiang Zhang, Jun Zhou, and Meng Wang. 2023. Generative-contrastive graph learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 1117–1126.

[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* 974–983.

[46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[47] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 1294–1303.

[48] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 1 (2023), 335–355.

[49] Huanhuan Yuan, Pengpeng Zhao, Xuefeng Xian, Guanfeng Liu, Yanchi Liu, Victor S. Sheng, and Lei Zhao. 2023. Sequential recommendation with probabilistic logical reasoning. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence,* 2432–2440.

[50] An Zhang, Leheng Sheng, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. 2023. Empowering collaborative filtering with principled adversarial contrastive loss. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, 6242–6266.

[51] Yiding Zhang, Chaozhuo Li, Xing Xie, Xiao Wang, Chuan Shi, Yuming Liu, Hao Sun, Liangjie Zhang, Weiwei Deng, and Qi Zhang. 2022. Geometric disentangled collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval,* 80–90.

[52] Yi Zhang, Yiwen Zhang, Dengcheng Yan, Shuiguang Deng, and Yun Yang. 2023. Revisiting graph-based recommender systems from the perspective of variational auto-encoder. *ACM Transactions on Information Systems* 41, 3 (2023), 1–28.

[53] Yi Zhang, Yiwen Zhang, Dengcheng Yan, Qiang He, and Yun Yang. 2024. NIE-GCN: Neighbor item embedding-aware graph convolutional network for recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54, 5 (2024), 2810–2821.

[54] Yi Zhang, Yiwen Zhang, Yuchuan Zhao, Shuiguang Deng, and Yun Yang. 2024. Dual variational graph reconstruction learning for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* 36, 11 (2024), 6002–6015.

Received 14 April 2024; revised 4 August 2024; accepted 9 October 2024