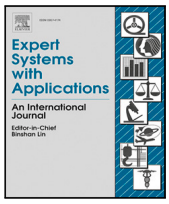
Expert Systems With Applications 284 (2025) 127855

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



GLSCL: Graph local similarity contrastive learning for recommendation

Zhou Zhou, Zheng Hu, Shi-Min Cai *, Tao Zhou

Big Data Research Center, University of Electronic Science and Technology, Chengdu, 611731, China

ARTICLE INFO

Keywords:
Recommender system
Contrastive learning
Graph neural network

ABSTRACT

Graph Contrastive Learning (GCL) has recently gained widespread adoption in recommendation systems owing to its outstanding performance and capability to alleviate data sparsity issues. GCL mitigates data sparsity issues by learning more uniformly distributed user and item representations. However, current recommendation approaches based on GCL have a significant drawback, which is to overlook the relationships between homogeneous nodes (i.e., users to users and items to items). The potential of contrastive learning in these contexts remains largely untapped because contrastive learning often focuses only on the user–item interaction space, missing the fine-grained, contextual similarities that exist within homogeneous nodes. These overlooked relationships can significantly improve recommendation accuracy by providing richer, more context-sensitive embeddings. To address this problem, we propose a Graph Local Similarity Contrastive Learning (GLSCL) framework, which enhances embedding uniformity and constructs contrast pairs based on local similarity. Specifically, to avoid the loss of original information, we employ a random perturbation contrastive task to enhance embedding uniformity and improve recommendation performance by exploring the inherent correlations among users (or items). GLSCL treats users (or items) with their local batch of most similar users (or items) as positive contrastive pairs during training, which can capture the homogenous relationships in user–user and item–item similarity relationships while maintaining embedding uniformity. To validate the proposed model, extensive experiments were conducted on three real-world datasets, including Douban-Book, Yelp, and Amazon-Book. On the three datasets, our model outperforms the suboptimal model with an average improvement of 3.23%, 3.28%, 3.55%, and 3.41% in four metrics, respectively. Extensive ablation experiments and visual analyses were conducted, providing conclusive evidence for the effectiveness of the proposed core modules.

1. Introduction

In the era of information explosion, recommendation systems play a crucial role in alleviating information overload (Lü et al., 2012; Sharma & Gera, 2013). Recommender systems have become an integral part of user engagement in online services, including product recommendations (McAuley et al., 2015), video recommendations (Cheng et al., 2025; Covington et al., 2016), social networks (Cheng et al., 2024; Hu et al., 2023; Li et al., 2025), online shopping (Zhou et al., 2018), and more. Recommendation systems effectively filter and screen information, allowing users to retrieve information. Collaborative Filtering (CF) methods like (Adomavicius & Tuzhilin, 2005; He et al., 2017), relying on the idea that similar users often share similar preferences, are widely used for personalized recommendations. Recently, Graph Neural Networks (GNNs) (Feng et al., 2020; He, Deng et al., 2020; Wang, He, Wang et al., 2019) have further enhanced CF by modelling user preferences and intentions for personalized recommendations. GNNs learn node representations by aggregating local collaborative signals through neighbourhood representations. For instance, NGCF (Wang, He, Wang et al., 2019) employs Graph Convolutional Networks (GCNs) to propagate information between neighbours, capturing high-order interactions between users and items. LightGCN (He, Deng et al., 2020) simplifies the message-passing process by omitting non-linear transformations and using average pooling, as it found that these contribute limitedly to CF. Despite significant successes of these methods, existing methods based on GNNs still face data sparsity challenges (Wu et al., 2021). In real-world recommendation scenarios, most users occasionally interact with a minimal number of items, resulting in sparse user–item interaction data. Many GNN-based models that rely entirely on the supervised learning paradigm cannot learn reliable user/item representations in cases of scarce interaction labels (Zheng et al., 2019). Recent applications of contrastive learning in various domains of deep learning (Chen et al., 2020; He, Fan et al., 2020; van den Oord et al.,

* Corresponding author.
E-mail addresses: 2018081303022@std.uestc.edu.cn (Z. Zhou), huzheng@std.uestc.edu.cn (Z. Hu), shimincai@uestc.edu.cn (S.-M. Cai), zhutou@ustc.edu (T. Zhou).

https://doi.org/10.1016/j.eswa.2025.127855
Received 11 March 2024; Received in revised form 16 April 2025; Accepted 21 April 2025
Available online 30 April 2025

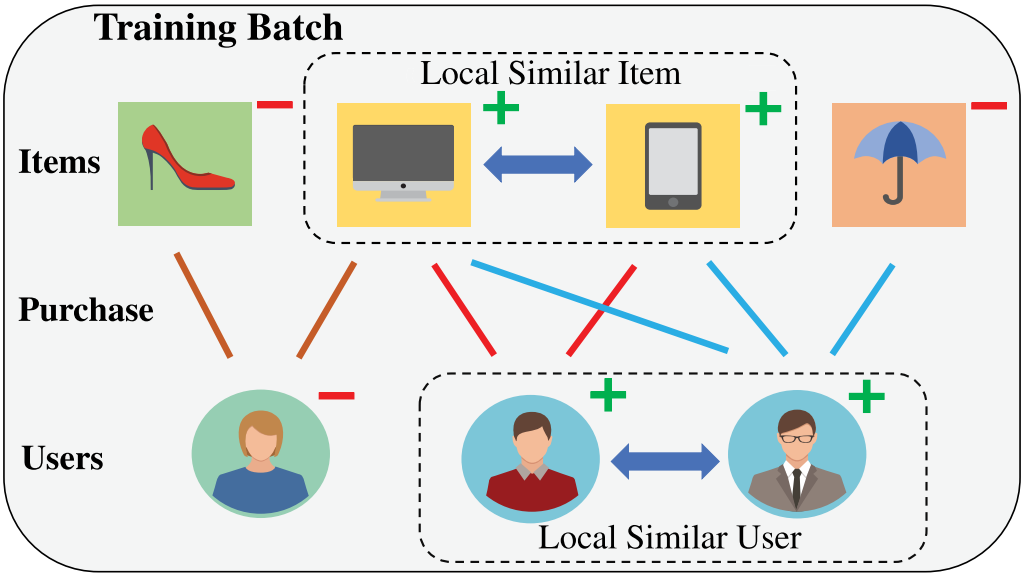


Fig. 1. An illustration of the proposed Graph Local Similarity Contrastive Learning Module, where the green "+" sign indicates a contrast positive pair and The red "–" sign indicate pairs of items in the training batch that are not similar and do not participate in our core comparison task. Different coloured lines represent different user interactions.

2018) have shown remarkable results and breakthroughs. Inspired by the progress of contrastive learning in these domains, Graph Contrastive Learning (GCL) has been proposed. GCL has recently garnered significant attention in graph representation learning due to its ability to alleviate data sparsity, mitigate noise interference, and enhance the robustness of recommendations through self-supervised learning (Qin et al., 2021; Xie et al., 2021). Contrastive learning is a form of self-supervised learning that aims to bring similar samples closer and push different pieces farther apart. The capacity of contrastive learning to learn general features from unlabelled data can effectively address data sparsity issues.

However, existing contrastive learning approaches still have several limitations. Many GCL-based recommendation models, such as SGL (Wu et al., 2021), use heuristic graph data augmentation methods, like node or edge dropout, to construct contrastive views. This often leads to the loss of original information. SimGCL (Yu et al., 2022) found that graph data augmentation is not necessary and proposed a simple yet effective graph-less contrastive learning recommendation method, which can smoothly adjust uniformity. However, SimGCL only focuses on embedding distribution uniformity and does not explore various potential relationships between nodes. CGCL (He et al., 2023) constructs various contrastive learning objectives by leveraging relations between users and candidate items at different embedding layers to improve node embedding quality. However, it indirectly and intricately explores semantic similarities from different layers without explicitly proposing a practical and direct method that utilizes node similarity semantics. Existing methods perform neighbourhood information aggregation between heterogeneous nodes in the user–item bipartite graph. Two users, even with similar consumption habits, are at least at a distance of two hops (user–item–user), indirectly hindering information propagation between homogeneous neighbour nodes. This overlooks the direct adjacent relevance between homogeneous nodes (users or items). Beyond user–item interactions, there exist various latent relationships beneficial for recommendation tasks, such as user similarity and item similarity. We aim to design more effective contrastive learning methods to leverage these valuable relationships in neural graph filtering. By leveraging rich node relationships and contrastive learning to improve node representation learning, the potential for recommendations can be enhanced.

Given the abovementioned challenges, we propose a novel contrastive learning framework called Graph Local Similarity Contrastive Learning (GLSCL). Specifically, we design a contrastive learning strategy that leverages user similarity and item similarity semantics to contrast nodes. We also introduce random perturbations to the node encoding process for contrastive learning to obtain a more uniform embedding distribution. Instead of contrasting nodes by augmenting two views through data augmentation, we contrast them using locally similar nodes from the dataset. As shown in Fig. 1, positive contrastive pairs are formed by selecting the most similar users in terms of purchase preferences within a local batch for users, and similarly, positive contrastive pairs for items are constructed by selecting the most similar items within the local batch. The positive comparison pairs for each training batch usually change. By incorporating these additional relationships, we can capture and model the homogenous relationships between users and between items. Our experimental results demonstrate that it significantly improves the performance of recommendation methods based on GNNs. Our approach establishes contrastive learning tasks tailored to data distribution through uniformity and local similarity modelling, enhancing the graph neural collaborative filtering paradigm. Our contributions are highlighted as follows:

- We propose a simple yet effective contrastive learning framework, GLSCL, which combines random perturbation contrast tasks and local collaborative relationships between users and items to improve neural graph collaborative filtering, further optimizing contrast signals.
- We propose a novel homogenous-relationship-aware sampling method, which samples the most similar users and items in the training batch to create contrast views, avoiding the issue of single contrast targets during training and capturing the homogenous relationships between users and items.
- Extensive experiments on three public datasets demonstrate that our proposed GLSCL significantly outperforms strong baselines and recent contrastive learning-based recommendation methods. Multiple ablation experiments validate the rationality and effectiveness of GLSCL. Furthermore, our method effectively addresses the challenge of data sparsity.

2. Related work

In this section, we briefly review relevant work in two aspects: GNN-based recommendations and GCL-based recommendations.

2.1. GNN-based recommendations

In recent years, GNN-based recommendation has garnered significant attention due to their ability to model complex relationships between users and items in recommendations effectively (Fan et al., 2019; Wu et al., 2023). Since most information can be organized as a graph, interactions between users and items are often viewed as a bipartite graph, and GNNs are employed to refine the representations of each node. A significant advantage of GNN architectures is their ability to capture multi-hop relations between users and items, enabling the extraction of high-order structural information. Under the framework of GNNs, many recommendation systems based on GNNs have been proposed to capture various graph structural relationships in recommendations (Hamilton et al., 2017; He, Deng et al., 2020; Wang, He, Wang et al., 2019). NGCF (Wang, He, Wang et al., 2019) demonstrated the importance of high-order relations between users and items in collaborative filtering. GC-MC (van den Berg et al., 2017) modelled each node using only neighbouring nodes, and neighbours are aggregated through pooling modules for rating prediction tasks. LightGCN (He, Deng et al., 2020) proposed to omit non-linear transformations in the propagation process and perform neighbourhood aggregation using a simple weighted summation operation. Due to its simplicity and excellent performance, LightGCN has become one of the most popular GCN-based recommendation models. Some prior works have integrated side information, such as social networks and knowledge graphs, and multiple behaviours into recommendation models. For instance, KGAT (Wang, He, Cao et al., 2019) incorporates items into a knowledge graph and considered user–item interactions as a type of relation. DNNRec (Kiran et al., 2020) proposes a hybrid recommender system that integrates auxiliary information about users and items into very deep neural network to alleviate the cold start problem. IGT (Wang, Zeng et al., 2023) is based on session sequences, constructing an interval-enhanced session graph and utilizing a graph transformer with embedded time intervals to learn interaction information among items. Satori (Chen et al., 2022) is a next-item recommendation model that constructs a user interaction graph and utilizes a graph attention network to combine user intention and preference. By integrating self-attention mechanisms and hybrid user representations, Satori effectively addresses the data sparsity issue. GANRec (Yang et al., 2023) integrates negative sampling and Generative Adversarial Networks (GANs) on a bipartite graph to improve recommendation effectiveness. TDSRec (Wang, Shi et al., 2023) introduces a temporal Density-aware sequence recommendation network with contrastive learning, eliminating handcrafted data augmentation strategies on the raw sequence data. HDRSR (Hu et al., 2025) introduces a hierarchical denoising robust social recommendation model that addresses intra-domain and inter-domain noise by refining social ties and applying a user interest-aware cross-domain denoising gate, improving the model's diversity, robustness, and performance. SR-HetGNN (Chen, Li et al., 2024) is a session-based recommendation method that leverages heterogeneous graph neural networks to capture both item transitions and user-specific preferences, addressing the limitations of current methods by incorporating user profile information into session embeddings. SIGFormer (Chen, Chen et al., 2024) combines positive and negative feedback, using the Transformer architecture for recommendation. CIKGRec (Hu et al., 2025) introduces a GNN-based knowledge-aware recommendation model based on large language model enhanced knowledge graphs. These GNN-based models encode interactions between users and items, allowing the model to capture multi-hop relations between users and items, thus generating more effective recommendations. However, many GNN-based models that rely entirely on supervised learning encounter data sparsity issues, especially when interaction data between users and items is scarce, which significantly limits the model's performance.

2.2. GCL-based recommendations

Recent promising research has integrated contrastive learning into graph-based recommendation systems to address the label sparsity problem. In the recommendations domain, contrastive learning has emerged as a practical self-supervised framework to capture consistency in feature representations across different views. For example, SGL (Wu et al., 2021) performs data augmentation on the graph structure through random node dropout, edge dropout, and random walk operations to obtain two augmented views, and representations of the same node in different views are contrasted. NCL (Lin et al., 2022) designed neighbourhood enhancement schemes, utilizing a clustering algorithm based on expectation maximum to contrast nodes with cluster centres in the representation space. SimGCL (Yu et al., 2022) achieved outstanding performance by introducing random noise to node embedding representations for data augmentation. XSimGCL (Yu et al., 2024) simplifies the architecture by combining the encoding process of SimGCL, reducing the number of forward and backward passes in small-batch computations. In HCCF (Xia et al., 2022), local–global contrastive learning was designed for self-supervised enhancement based on parameterized hypergraph structures. LightGCL (Cai et al., 2023) employed singular value decomposition and reconstruction for data augmentation, enhancing the ability to handle adversarial data sparsity and popularity bias. CGCL (He et al., 2023) explored the relationships between users and candidate items in different layer embeddings and constructed contrastive pairs using similar semantic embeddings. These contrastive learning methods leverage auxiliary signals for specific data or task-specific enhancement. Furthermore, contrastive learning has been applied to various recommendation scenarios, including social recommendations, session-based recommendations, and multi-behaviour recommendations. MHCN (Yu, Yin, Li et al., 2021) proposed a multi-channel hypergraph convolutional network that leveraged high-order user relationships to enhance social recommendations. DHCN (Xia et al., 2021) modelled session data as hypergraphs and introduced a dual-channel hypergraph convolutional network to improve recommendations. CCGL (Xu et al., 2023) improves the information cascade model by using contrast, self-supervised, and task-agnostic information cascade graph learning. Many GCL-based approaches rely on heuristic data enhancement techniques such as random node/edge dropping and attribute masking, which can result in the loss of critical information, affecting the quality of the learned embeddings. In addition, many methods tend to ignore relationships between isomorphic nodes, and these neglected relationships limit the model's ability to take full advantage of the supplementary information available in user-project interactions, thus hindering the overall performance of the model.

3. Methodology and preliminaries

In this section, we aim to provide a detailed description of the proposed GLSCL framework. As shown in Fig. 2, GLSCL is a lightweight GCL paradigm consisting of three components: (1) The first component employs a graph-based message-passing module as an encoder to capture local collaboration relationships between users and items. This module serves to encode the local cooperation dynamics within the graph structure. (2) The second component introduces a strategy for local similarity node contrastive learning. It is designed to explore similar nodes from local training batches, constructing positive contrastive pairs to mine semantic similarity between nodes and improve the recommendation effectiveness. (3) The third component introduces a random perturbation task to enhance the uniformity of node embeddings. This task ensures the embeddings are robust and well-distributed across the feature space. In summary, GLSCL integrates these three components to form a holistic framework for lightweight GCL, addressing local collaboration, uniformity, and similarity aspects to enhance the quality of learned embeddings for recommendation tasks. Next, we will introduce preliminaries and give details on the three components.

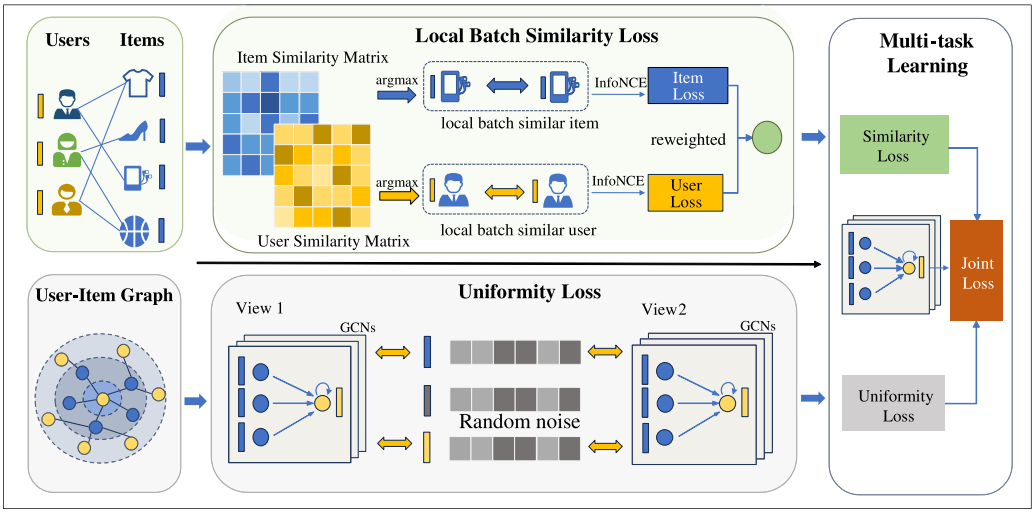


Fig. 2. Overall framework of the proposed GLSCL model.

3.1. Preliminaries

In most common settings, a standard recommender system comprises two sets of entities: a user set $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ containing m users ($|\mathcal{U}| = m$) and an item set $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ containing n items ($|\mathcal{I}| = n$). User–item interactions are recorded in a rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, where $\mathbf{R}_{ij} = 1$ indicates an interaction (e.g., purchase or like) between user i and item j, and $\mathbf{R}_{ij} = 0$ otherwise. Construct the user–item interactions as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where the node set $\mathcal{V} = \{\mathcal{U} \cup \mathcal{I}\}$ involves all users and items, and the edge set $\mathcal{E} = \{\mathbf{R}_{ij} | \mathbf{R}_{ij} = 1\}$ represents observed interactions. The goal of GLSCL is to predict unobserved interactions between users and items by the given input graph $\mathcal{G}$. The important notations in this paper can refer to Table 1.

3.2. Basic embedding computation

Following LightGCN (He, Deng et al., 2020), we omit non-linear activation functions and feature transformations to model interactions observed between users and items. The computation for each user and item embedding representation is as follows:

$$\begin{aligned} \mathbf{e}_u^{(l+1)} &= \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \mathbf{e}_i^{(l)}, \\ \mathbf{e}_i^{(l+1)} &= \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_u|}} \mathbf{e}_u^{(l)}, \end{aligned} \tag{1}$$

where $\mathbf{e}_u^{(l)}$ and $\mathbf{e}_i^{(l)}$ represent the embedding of user u and item i in the l-layer, respectively. When $l = 0$, it represents the initial embedding of the nodes initialized by Xavier initializer (Glorot & Bengio, 2010). N_u and N_i denote the neighbours of user u and item i in the interaction graph $\mathcal{G}$, respectively. After L layers of message propagation, average weighted sum pooling is applied to generate the ultimate user/item representations, and the initial embedding is removed in our work:

$$\begin{aligned} \mathbf{e}_u &= \frac{1}{L}\left(\sum_{l=0}^{L} \mathbf{e}_u^{(l)} - \mathbf{e}_u^{(0)}\right), \\ \mathbf{e}_i &= \frac{1}{L}\left(\sum_{l=0}^{L} \mathbf{e}_i^{(l)} - \mathbf{e}_i^{(0)}\right), \end{aligned} \tag{2}$$

where $\mathbf{e}_u$ and $\mathbf{e}_i$ represent the final representations of user u and item i, a classic approach to predicting the likelihood of u adopting i is to use the dot product.

$$\hat{y}_{u,i} = \mathbf{e}_u^{\top} \mathbf{e}_i, \tag{3}$$

Table 1
The major notations.

Notations	Description
$\mathcal{U}$	User set
$\mathcal{I}$	Item set
$\mathcal{V}$	Node set (union of $\mathcal{U}$ and $\mathcal{I}$)
$\mathcal{E}$	Edge set (observed user–item interactions)
$\mathcal{G}$	User–item bipartite graph
$\mathcal{B}$	Model training batch
$\mathcal{N}_i$	Neighbours of node i
$\mathbf{R}$	User–item rating matrix
$\mathbf{L}$	Layers of message propagation
$\mathbf{S}_{user}$	User similarity matrix
$\mathbf{S}_{item}$	Item similarity matrix
$\mathbf{e}_u$	Embedding of user u
$\mathbf{e}_i$	Embedding of item i
$\mathbf{z}_i$	Normalized embedding of node i
$\hat{y}_{u,i}$	Predicted score for user u and item i
ϵ	Constant for noise constraint
τ	Temperature for contrastive learning
$\mathbf{h}'_i, \mathbf{h}''_i$	Random noise vectors
α	Hyperparameter for balancing user loss and item loss
$\lambda_1, \lambda_2, \lambda_3$	Hyperparameter for balancing multi-task learning losses
T	Similarity matrix store threshold
σ	Sigmoid function
$\mathcal{L}$	Model loss function
Θ	Model trainable parameters

where $\hat{y}_{u,i}$ is the predicted score for user u and item i. To optimize model parameters, existing work often formulates the task as one of supervised learning, where the supervisory signal comes from observed interactions. We adopt the Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009). The objective function of the BPR loss is as follows:

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in \mathcal{O}} -\log\left(\sigma\left(\hat{y}_{u,i} - \hat{y}_{u,j}\right)\right), \tag{4}$$

where $\mathcal{O}$ indicates the training set, u denotes the user, i stands for the positive sample, j signifies a randomly selected negative sample from items with which the user u has not interacted, and $\sigma(\cdot)$ is the sigmoid function.

3.3. Contrastive learning with graph local similar node

Many models ignore the effect of homogeneous neighbours and do not fully utilize the potential of contrast learning. To tackle this issue, we propose another contrastive task: Contrastive Learning with Graph Local Similar Node. This method is a novel homogenous-relationship-aware sampling method for contrastive learning, and the mechanism

facilitates the contrast between locally similar nodes within the training batch. Specifically, GLSCL samples the most similar users and items in the local training batch to create contrast views, avoiding the issue of single contrast targets during training and capturing the homogenous relationships in user–user and item–item similarity relationships. GLSCL allows users to be compared with multiple similar neighbours during training, allowing them to explore similar semantics while reducing the risk of over-smooth embeddings around popular users or items. During the training process, the local most similar user or item for each user or item usually changes according to the range of batch selection, and the user and item are compared with themselves when the similarity of the positive pair does not reach the threshold. In order to solve the sparsity problem in the user–item interaction matrix, we adopt the normalized cosine similarity calculation. Specifically, all interactions of each user with the item are treated as a vector. The numerator is the dot product of two vectors (representing the number of items you like together), while the denominator normalizes the vector length using L2 normalization. This ensures that cosine similarities are calculated with unit lengths, thus eliminating the effect of differences in interaction frequencies. Mathematically, the cosine similarity computation can be expressed in vector form as follows:

For user similarity, let $\mathbf{u}_i$ and $\mathbf{u}_j$ represent the interaction vectors of users i and j, where each element corresponds to the interaction score (0 or 1) with an item. The similarity $S_{user}(i,j)$ is given by:

$$S_{user}(i,j) = \frac{\mathbf{u}_i \cdot \mathbf{u}_j}{\|\mathbf{u}_i\|_2 \cdot \|\mathbf{u}_j\|_2}, \tag{5}$$

where $\mathbf{u}_i \cdot \mathbf{u}_j$ denotes the dot product of the two vectors, and $\|\mathbf{u}_i\|_2 = \sqrt{\sum_{k=1}^{n} u_{i,k}^2}$ represents the L2 norm of vector $\mathbf{u}_i$. Similarly, for item similarity, let $\mathbf{v}_i$ and $\mathbf{v}_j$ represent the interaction vectors of items i and j, where each element corresponds to the interaction score with a user. The similarity $S_{item}(i,j)$ is given by:

$$S_{item}(i,j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\|_2 \cdot \|\mathbf{v}_j\|_2}, \tag{6}$$

where $\mathbf{v}_i \cdot \mathbf{v}_j$ denotes the dot product of the two vectors, and $\|\mathbf{v}_i\|_2 = \sqrt{\sum_{k=1}^{m} v_{k,i}^2}$ represents the L2 norm of vector $\mathbf{v}_i$. The expanded form of the similarity matrix calculation is as follows:

$$\begin{aligned} \mathbf{S}_{user}(i,j) &= \frac{\sum_{k=1}^{n} \mathbf{R}_{i,k} \cdot \mathbf{R}_{j,k}}{\sqrt{\sum_{k=1}^{n} \mathbf{R}_{i,k}^2} \cdot \sqrt{\sum_{k=1}^{n} \mathbf{R}_{j,k}^2}}, \\ \mathbf{S}_{item}(i,j) &= \frac{\sum_{k=1}^{m} \mathbf{R}_{k,i} \cdot \mathbf{R}_{k,j}}{\sqrt{\sum_{k=1}^{m} \mathbf{R}_{k,i}^2} \cdot \sqrt{\sum_{k=1}^{m} \mathbf{R}_{k,j}^2}}, \end{aligned} \tag{7}$$

In the equation, $\mathbf{R}$ represents the user–item interaction matrix, while $\mathbf{S}_{user}$ and $\mathbf{S}_{item}$ denote the user and item similarity matrices, respectively. This normalization step ensures that users or items with fewer interactions are not disproportionately affected during similarity computation. By normalizing the embeddings, we mitigate the risk of popular users or items dominating the similarity computation, thereby improving the robustness of our method in handling sparse data.

Subsequently, we embark on a quest to identify the most similar nodes to both user u and item i within the training batch, as opposed to considering the most similar users or items globally. This approach allows users to contrast with multiple similar neighbours within the training process, enabling the exploration of similarity semantics while mitigating the risk of excessive differentiation of embeddings around popular users or items. Even in extremely sparse cases with few user interactions, we can build contrast pairs in the local similarity contrast task by using cosine similarity to find another user with few interactions but similar preferences. The method of obtaining the positive contrast pair index i^* for node index i is as follows:

$$i^* = \underset{j \neq i, j \in \mathcal{B}}{\arg\max} \mathbf{S}(i,j), \tag{8}$$

in the training batch $\mathcal{B}$, it means finding the global index of the local most similar node for the node of index i. The process of obtaining a local similar node for user u is as follows: Similarly, the process for obtaining local similar nodes for items is as follows:

$$\mathbf{e}_i^{sim} = \begin{cases} \mathbf{e}_i, & \text{if } \mathbf{S}_{item}(i,i^*) < T \\ \mathbf{e}_{i^*}, & \text{otherwise.} \end{cases} \tag{9}$$

In cases where user or item interactions are extremely sparse, local similarity computation may yield low-quality contrastive pairs with insufficient similarity. To address this issue, we introduce a similarity threshold T to filter out pairs with insufficient similarity. Specifically, if no sufficiently similar neighbour is found, the node is contrasted with itself. This mechanism reduces the influence of noisy or irrelevant pairs during training. In the actual experiment, we only need to focus on the most similar part of the data set of users and items (such as 10% proportion) to get good experimental results. If the overall similarity in the dataset is low, the threshold can be adjusted to include users or items with relatively high similarity for local similarity contrast learning.

Next, we design the contrastive learning loss for users:

$$\mathcal{L}_{sim}^{U} = \sum_{u \in U} -\log \frac{\exp\left(\mathbf{e}_u \cdot \mathbf{e}_u^{sim}/\tau\right)}{\sum_{j \in \mathcal{B}_U} \exp\left(\mathbf{e}_u \cdot \mathbf{e}_j/\tau\right)}, \tag{10}$$

where $\mathbf{e}_u^{sim}$ represents the local similar user embedding to user u in the current training batch. The local similar node contrastive learning loss on the item side can be represented as follows:

$$\mathcal{L}_{sim}^{I} = \sum_{i \in I} -\log \frac{\exp\left(\mathbf{e}_i \cdot \mathbf{e}_i^{sim}/\tau\right)}{\sum_{j \in \mathcal{B}_I} \exp\left(\mathbf{e}_i \cdot \mathbf{e}_j/\tau\right)}. \tag{11}$$

Combining the local similar user contrastive learning loss and local similar item contrastive learning loss, we obtain the contrastive learning loss for the graph local similar node contrastive learning task:

$$\mathcal{L}_{sim} = \mathcal{L}_{sim}^{U} + \alpha \mathcal{L}_{sim}^{I}, \tag{12}$$

where α is a hyperparameter that balances the weights of the two losses in graph local similar node contrastive learning task.

3.4. Contrastive learning with random noise perturbation

In addition to the basic message passing, we integrate a random perturbation loss task similar to SimGCL (Yu et al., 2022). This process aims to adjust the uniformity of learned representations within a specific range to enhance recommendation performance. In generating embeddings, we introduce random noise to the node embeddings and use the final embeddings for contrastive learning. Formally, given a node i and its representation $\mathbf{e}_i$, two new representations are generated by applying random perturbations to the $\mathbf{e}_i$ and passing them through the layers of LightGCN:

$$\mathbf{e}_i' = \mathbf{e}_i + \mathbf{h}_i', \ \mathbf{e}_i'' = \mathbf{e}_i + \mathbf{h}_i'', \tag{13}$$

$\mathbf{h}_i'$ and $\mathbf{h}_i''$ are two independently generated noise vectors, and the added noise vectors $\mathbf{h}_\mathbf{i}'$ are generated as follows: First, random values are sampled from a uniform distribution $U(0,1)$ to create tensors of the same shape as the input embedding matrix. Formally, let $\mathbf{r}_\mathbf{i} \in \mathbb{R}^d$ denote the random tensor sampled from $U(0,1)$, where d is the dimensionality of the embedding space. Next, these random tensors are normalized to unit L2 norm to ensure that the magnitude of the perturbation vectors is consistent across all dimensions. The normalization process can be expressed as:

$$\bar{\mathbf{h}}_\mathbf{i} = \frac{\mathbf{r}_\mathbf{i}}{\|\mathbf{r}_\mathbf{i}\|_2}, \tag{14}$$

where $\|\mathbf{r}_\mathbf{i}\|_2 = \sqrt{\sum_{i=1}^{d} r_i^2}$ denotes the L2 norm of the random tensor $\mathbf{r}_\mathbf{i}$. This hypersphere constraint ensures that the perturbation vectors have a consistent magnitude of 1, which helps maintain the structural

Table 2
The comparison of time complexity.

Stage	LightGCN	SGL	SimGCL	**GLSCL**
Pre-processing	$O(2E)$	$O(2E+4\rho E)$	$O(2E)$	$O(2E+U^2+I^2)$
Graph encoding	$O(2ELd)$	$(2+4\rho)ELd$	$O(6ELd)$	$O(6ELd)$
Prediction	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$
Contrastive learning	–	$O(Bd+BMd)$	$O(BMd)$	$O(2B^2+2Bd+2BMd)$

integrity of the embedding space. Without this constraint, unbounded perturbations could lead to excessive divergence of the embeddings during training, disrupting the local similarity structure.

To preserve the semantic meaning of the embeddings, the noise vectors are restricted to remain within the same hyperoctant as the original embeddings. This is achieved by multiplying the normalized noise vector $\bar{\mathbf{h}}$ with the sign of the corresponding embedding components:

$$\mathbf{h_i} = \bar{\mathbf{h}_i} \odot \text{sign}(\mathbf{e_i}), \tag{15}$$

where $\odot$ denotes the element-wise product, and $\text{sign}(\mathbf{e_i})$ is a vector whose elements are the signs of the corresponding components of the embedding $\mathbf{e_i}$. Restricting noise vectors to the same hyperoctant prevents drastic changes to the embeddings' semantic meaning. This constraint allows exploration of possible future embedding changes. By maintaining the same hyperoctant, perturbations do not reverse learned representations, preserving the embedded semantics.

Finally, the perturbation strength is scaled by a hyperparameter ϵ to adapt to the learned embedding magnitudes. The final perturbed embedding is computed as:

$$\mathbf{e}'_\mathbf{i} = \mathbf{e_i} + \epsilon \cdot \mathbf{h_i}. \tag{16}$$

The uniformity loss function is as follows:

$$\mathcal{L}_{uniform} = \sum_{i\in\mathcal{B}} -\log \frac{\exp\left(sim\left(\mathbf{z}'_i, \mathbf{z}''_i\right)/\tau\right)}{\sum_{j\in\mathcal{B}} \exp\left(sim\left(\mathbf{z}'_i, \mathbf{z}''_j\right)/\tau\right)}, \tag{17}$$

where $\mathcal{B}$ represents the training batch, τ is the temperature for contrastive learning, $\text{sim}(\cdot)$ stands for computation function, here we use dot product and $\mathbf{z}'_i$ is the normalized representation of node i where $\mathbf{z}'_i = \frac{\mathbf{e}'_i}{\|\mathbf{e}'_i\|_2}$. This formula is in the style of InfoNCE (van den Oord et al., 2018), which aims to learn more uniformly distributed user and item representations.

3.5. Optimization

The overall optimization objective treats the BPR loss as the primary objective for modelling interactions between users and items while considering the random perturbation contrastive learning loss and local similar node contrastive learning loss as supplementary objectives. We employ a multi-task learning strategy to jointly train the traditional ranking loss and the proposed contrastive learning losses.

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_{uniform} + \lambda_2 \mathcal{L}_{sim} + \lambda_3 \|\Theta\|_2, \tag{18}$$

where λ_1, λ_2, and λ_3 control the two proposed contrastive learning losses and regularization terms, Θ indicates the trainable parameters. The algorithmic pseudocode is shown in Algorithm 1.

3.6. Time complexity

In this section, we analyse the time complexity of our proposed GLSCL framework and compare it with LightGCN, SGL, and SimGCL. We summarize the results in Table 2. Here, E represents the number of edges, L denotes the number of users and items, d is the embedding dimension, B is the batch size, and M is the number of nodes in a batch. $\rho \in (0,1)$ represents the retention rate for edge pruning in SGL. From the table, we can observe the following:

Algorithm 1 Graph Local Similarity Contrastive Learning (GLSCL).

Input: user-item interactions data $\mathbf{R}$, training dataset $\mathcal{X}$, model parameters Θ
Output: user and item representations $\{\mathbf{e_u}, \mathbf{e_i}\}$

```
// Preprocess: Calculate similarity matrices S_user and S_item
while GLSCL Not Convergence do
  for x ∈ Dataloader(X) do
    // Forward propagation
    // Get user embedding e_u, positive item embeddings e_i+, and negative item embeddings e_i-
    // Calculate BPR Loss L_main
    L_main = cal_BPR_loss(e_u, e_i+, e_i-)
    // Calculate uniformity loss L_uniform
    L_uniform = cal_uniformity_loss(z'_u, z''_u, z'_i, z''_i)
    // Calculate the local batch of the most similar embeddings
    e_u^sim, e_i^sim = find_most_similar(e_u, e_i)
    // Calculate local batch similarity loss L_sim
    L_sim = cal_local_batch_similarity_loss(e_u, e_u^sim, e_i, e_i^sim)
    // Calculate total loss L
    L = L_main + λ1 L_uniform + λ2 L_sim + λ3 ||Θ||_2
    // Back propagation
    backward(L)
    Model parameters: Θ = Θ − η∇_Θ L
    Update user and item representations {e_u, e_i}
  end for
end while
return {e_u, e_i}
```

- All models require $O(2E)$ time to normalize the adjacency matrix. For SGL, there are two enhanced views, each augmented with $2\rho E$ non-zero elements. Our model also requires additional preprocessing to compute the user–user and item–item similarity matrices, which takes $O(|U|^2+|I|^2)$ time. Since this preprocessing step occurs only once at the start of training, its computational cost is negligible compared to the training stage.
- In the graph encoding stage, traditional methods like LightGCN require $O(2ELd)$ for graph convolution, whereas SimGCL and GLSCL employ a three-encoder architecture, resulting in a time complexity of $O(6ELd)$.
- In the prediction stage, all models use the BPR loss function, so the time complexity for prediction is the same across all methods, specifically $O(2Bd)$, where B is the batch size.
- For GLSCL, the time complexity for contrastive learning is $O(2B^2+2Bd+2BMd)$. This includes the cost of precalculating local batch similarities, which takes $O(2B^2)$ time, as well as the time complexity of both contrastive tasks, which is $O(Bd+BMd)$. In contrast, the complexity for SGL is $O(Bd+BMd)$, and SimGCL's cost is $O(BMd)$.

Although GLSCL incorporates multiple modules, including local similarity contrastive tasks, embedding perturbations, and multi-task learning strategies, its computational cost remains competitive due to the efficient design of these components. Specifically, the substantial improvements in recommendation performance and embedding consistency achieved by our model justify the added complexity.

Table 3
Statistics of the datasets.

DataSet	# User	# Item	# Interaction	Density
Douban-Book	13,024	22,347	792,062	0.272%
Yelp	31,668	38,048	1,561,406	0.118%
Amazon-Book	52,463	91,599	2,984,108	0.062%

4. Experiments

4.1. Experimental setup

4.1.1. Datasets

To evaluate the performance of the proposed GLSCL, we conducted experiments on three public datasets: Douban-Book, Yelp, and Amazon-Book. The statistics of these datasets are shown in Table 3.

- **Douban-Book** (Yu, Yin, Gao et al., 2021): Douban is a popular book and movie social platform. Douban-Book has ratings from 1 to 5. Following previous work (Yu et al., 2022), we discarded ratings lower than 4 in the Douban-Book dataset and set others to 1.
- **Yelp** (He, Deng et al., 2020): This is a business recommendation dataset that has been widely used to evaluate recommendation systems with the task of recommending businesses.
- **Amazon-Book** (Wu et al., 2021): Amazon is one of the world's largest e-commerce companies. This dataset records user ratings for book-related products on Amazon.

4.1.2. Evaluation metrics

To evaluate the performance of top-N recommendations, we used two widely used evaluation metrics: Recall@K and NDCG@K. Where K is set to 10 and 20, and other baseline methods were compared using these values. For users in the test set, we evaluated the top-K recommendation performance and reported the average Recall@K and NDCG@K. We adopted a full-ranking strategy (He et al., 2015) to rank items that users have not interacted with. We split the interactions into training, validation, and test sets in a 7:1:2 ratio.

4.1.3. Baseline models

We compared the proposed GLSCL model with the following baseline methods:

- **MF** (Koren et al., 2009) is a naive matrix factorization method that directly decomposes the user–item rating matrix.
- **LightGCN** (He, Deng et al., 2020) is an advanced CF method. It simplifies the process by omitting non-linear transformations and reducing the number of parameters, resulting in better performance.
- **SGL** (Wu et al., 2021) introduces self-supervised learning to enhance recommendations by performing data augmentation through node dropout, edge dropout, and random walks. We used SGL-ED as an instantiation of SGL.
- **NCL** (Lin et al., 2022) is a contrastive learning method that enriches collaborative filtering by merging potential neighbours into contrastive pairs. NCL introduces structural and semantic neighbours of users and items, developing structural contrast and clustering prototype contrast objectives.
- **DirectAU** (Wang et al., 2022) proposes a novel learning objective for collaborative filtering based on alignment and uniformity of hyper-spheres to assess the quality of representations. It directly optimizes both properties to improve recommendation performance.
- **SimGCL** (Yu et al., 2022) revealed that contrastive learning takes effect by learning more uniformly distributed embeddings, and the graph augmentation deemed necessary played only a trivial role. It optimizes the uniformity of the embedding distribution through random perturbation to obtain better recommendation results.
- **CGCL**: He et al. (2023) explores the relationships between users and candidate items in different layer embeddings and constructs contrast pairs using similar semantic embeddings.
- **AdaGCL**: Jiang et al. (2023) is a novel framework that enhances recommender systems through adaptive graph contrastive learning. It uses two trainable view generators, a graph generation model and a graph denoising model to create adaptive contrast views. These views address model collapse and provide high-quality training signals, improving the robustness of graph neural collaborative filtering.
- **XSimGCL** (Yu et al., 2024) introduces a novel and simplified approach to graph contrastive learning by discarding the graph augmentation techniques commonly used in previous methods.
- **SIGformer** (Chen, Chen et al., 2024) addresses the limitations of existing methods that separately handle positive and negative feedback in recommender systems. By employing the transformer architecture, SIGformer fully leverages both types of feedback through signed graphs.dgsdg

4.1.4. Hyperparameters settings

We implemented the proposed model and all baselines in Pytorch, and the replication of baseline results was doubly verified against the original code framework. To ensure a fair comparison, we adjusted the hyperparameters of all baselines within the ranges recommended in the original papers. For our GLSCL, we tuned the batch size in {2000, 4000, 6000, 8000, 10 000}, set the embedding size to 64, and initialized all parameters with the default Xavier distribution. In GLSCL, we empirically let the temperature $\tau = 0.2$ and λ_1 in the range [0, 3.0]. We searched for the weight λ_2 parameter in {1e−4, 1e−5, 1e−6, 1e−7, 1e−8}, similarity threshold T in the range [0, 0.2] and the α parameter in the range [0, 3.0]. We used the Adam optimizer with a learning rate of 0.001. For the number of GNN layers, we tried {2, 3, 4} and finally chose two layers based on performance. For the other baselines' hyperparameters, we select the optimal parameters reported in the original paper and adjust the datasets to select the best hyperparameters.

4.2. Overall performance comparison

As shown in Table 4, the effectiveness of GLSCL was validated by conducting a comprehensive performance evaluation of the three publicly available datasets and comparing GLSCL to various baselines. From the overall performance results table, we have the following key observations:

- Our proposed GLSCL consistently outperformed other baselines in top-10 and top-20 performance. This advantage is attributed to combining the proposed local batch similarity loss and random perturbation uniformity loss. Furthermore, our method exhibited more significant improvements on datasets with a higher ratio of items to users, such as the Douban-Book and Amazon-Book datasets, where item similarity is crucial for performance enhancement due to a richer set of items relative to the number of users.
- In self-supervised methods, the suboptimal model XSimGCL consistently outperformed other supervised methods on the three datasets, highlighting the effectiveness of uniformity in enhancing recommendation performance. However, XSimGCL focused only on uniformity and overlooked other potential relationships in the recommendation system, such as user similarity, failing to optimize node embeddings accordingly.

Table 4
Performance comparison with baselines on three datasets.

Setting		Baseline methods										Ours		
DataSet	Metric	MF	LightGCN	SGL	NCL	DirecAU	SimGCL	CGCL	AdaGCL	XsimGCL	SIGFormer	GLSCL	Improv.	p-val
Douban-Book	R@10	0.0864	0.1004	0.1235	0.1136	0.1137	0.1254	0.1228	0.1235	0.1257	0.1244	**0.1308**	+4.06%	1.3e−5
	N@10	0.0980	0.1170	0.1482	0.1317	0.1266	0.1499	0.1446	0.1468	0.1503	0.1478	**0.1561**	+3.85%	6.8e−5
	R@20	0.1290	0.1495	0.1761	0.1643	0.1576	0.1777	0.1760	0.1765	0.1784	0.1767	**0.1865**	+4.55%	2.4e−6
	N@20	0.1062	0.1261	0.1559	0.1407	0.1348	0.1575	0.1537	0.1551	0.1581	0.1553	**0.1646**	+4.12%	7.7e−4
Yelp	R@10	0.0282	0.0350	0.0397	0.0398	0.0414	0.0427	0.0419	0.0422	0.0428	0.0422	**0.0433**	+1.17%	3.2e−2
	N@10	0.0317	0.0403	0.0452	0.0457	0.0481	0.0490	0.0478	0.0485	0.0491	0.0481	**0.0497**	+1.22%	1.7e−2
	R@20	0.0491	0.0602	0.0678	0.0676	0.0707	0.0724	0.0715	0.0719	0.0725	0.0717	**0.0734**	+1.24%	8.4e−3
	N@20	0.0395	0.0495	0.0556	0.0557	0.0587	0.0597	0.0586	0.0591	0.0599	0.0587	**0.0605**	+1.00%	6.3e−2
Amazon-Book	R@10	0.0178	0.0227	0.0275	0.0256	0.0285	0.0312	0.0301	0.0291	0.0314	0.0304	**0.0328**	+4.46%	5.6e−6
	N@10	0.0187	0.0242	0.0291	0.0267	0.0315	0.0332	0.0314	0.0307	0.0335	0.0322	**0.0351**	+4.78%	8.1e−5
	R@20	0.0310	0.0392	0.0468	0.0444	0.0483	0.0514	0.0506	0.0497	0.0515	0.0509	**0.0540**	+4.85%	8.9e−7
	N@20	0.0239	0.0307	0.0368	0.0342	0.0390	0.0410	0.0395	0.0403	0.0411	0.0401	**0.0432**	+5.11%	6.1e−6

- Evaluation results indicate that self-supervised learning enhances existing Collaborative Filtering (CF) frameworks, such as SGL, NCL, SimGCL, XSimGCL, CGCL, and our proposed GLSCL model. This improvement is attributed to the enhancement of embedding uniformity through the contrastive learning task, providing beneficial regularization based on the input data. However, SGL's use of random data augmentation to generate multiple views may lose useful signals capturing important user–item interaction patterns. SimGCL and XSimGCL focus only on uniformity and neglect the homogeneous relationships of user–user and item–item similarity. GLSCL has two main advantages: it retains valuable information in random data augmentation, mining critical patterns in the original data, and ensures that the learned representations are diverse and information-rich. While maintaining uniformity, it captures the homogeneous relationships between user–user and item–item similarity.
- Compared to recent methods such as CGCL and AdaGCL, GLSCL offers distinct advantages. CGCL focuses on leveraging candidate-aware sampling to enhance the discriminative power of embeddings, which can lead to over-smoothing in dense regions. In contrast, GLSCL leverages local similarity-based contrastive pairs within the training batch, avoiding excessive smoothing while capturing fine-grained user and item preferences. Additionally, AdaGCL dynamically adjusts the contrastive loss using two trainable view generators, which introduces additional complexity. GLSCL achieves similar robustness and generalization performance with a simpler random perturbation strategy, reducing computational overhead while maintaining high-quality embeddings.

4.3. Ablation experiment

In this section, we delve deeper into a comprehensive analysis of the proposed GLSCL to confirm its effectiveness. We conducted a series of experiments on the Douban-book, Yelp, and Amazon-book datasets for this purpose. To gain a better understanding of the effectiveness of the components within GLSCL, we conducted extensive experiments. We performed ablation studies by individually removing the three applied techniques in GLSCL to analyse their contributions. These four variants are defined as follows:

- W/O – S&U&I: Removing the random perturbation contrastive task, the local similar user contrastive task and the local similar item contrastive task
- W/O – S: Removing the random perturbation contrastive task.
- W/O – U: Removing the local similar user contrastive task.
- W/O – I: Removing the local similar item contrastive task.

We observe that removing or retaining only a portion of the model's structure can achieve better performance than W/O – S&U&I across all datasets. This indicates that the three proposed contrastive learning objectives are effective. From Table 5, we can observe that the models with removed modules exhibit decreased performance compared to those without removal.

- Removing the embedding random perturbation performs better than the baseline W/O – S&U&I, indicating that both tasks outperform the baseline W/O – S&U&I. This suggests that explicitly modelling these two relationships is beneficial for the performance of graph-based collaborative filtering.
- Removing the local similar user contrastive task performs better than removing the local similar item contrastive task, and item similarity has a more significant impact on recommendation performance, especially in datasets where items outnumber users, such as Douban-Book and Amazon-Book. Users tend to have diverse interests, while items show clustering patterns. This makes item similarity more useful for capturing grouping patterns. The observation is also consistent with the early knowledge that item-based collaborative filtering is usually more accurate than user-based collaborative filtering (Zhou et al., 2009). The three relationships (local user similarity, local item similarity, and random perturbation) complement each other, improving the model's performance.

To further evaluate the advantages of random noise perturbation, we compare it with dropout-based noise techniques. In terms of time complexity, the time complexity of methods based on Dropout-edge and Dropout-node, as shown in Table 2, is $(2 + 4\rho)ELd$, where ρ represents the retention rate ($\rho \approx 1$). Our random perturbation method has a time complexity of $6ELd$ and does not impose a significant computational burden compared to dropout. Dropout randomly removes nodes or edges during training, which can lead to information loss, especially in sparse user–item interaction matrices. In contrast, our method introduces controlled perturbations constrained to the unit hypersphere, ensuring bounded perturbations that avoid drastic changes to the semantic meaning of the embeddings. This leads to more stable training and improved generalization. Table 6 shows the comparison results on three datasets (Douban-book, Yelp, and Amazon-book). Random noise perturbation strategy consistently outperforms dropout-based approaches in terms of Recall@20 and NDCG@20.

4.4. Impact of the hyperparameters

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_{uniform} + \lambda_2 \mathcal{L}_{sim} + \lambda_3 \|\Theta\|_2,$$
$$\mathcal{L}_{sim} = \mathcal{L}^{U}_{sim} + \alpha \mathcal{L}^{I}_{sim}$$

- **Impact of the Balance Coefficient α.** In the structure of the defined equation for the local contrastive loss, the coefficient alpha balances the two losses modelling local similarity node pairs in the contrastive loss: user similarity loss and item similarity loss. To analyse the impact of the coefficient α, we varied α within the set {0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0} and reported the results as shown in Fig. 3. The results indicate that an appropriate α can effectively enhance the performance of GLSCL. The following observations can be made:

Table 5
Ablation study on key components of GLSCL.

DataSet	Douban-Book		Yelp		Amazon-Book	
Metric	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
W/O - S&U&I	0.1495	0.1261	0.0602	0.0495	0.0392	0.0307
W/O - S	0.1548	0.1271	0.0626	0.0514	0.0409	0.0313
W/O - I	0.1750	0.1538	0.0728	0.0599	0.0504	0.0401
W/O - U	0.1862	0.1645	0.0730	0.0602	0.0533	0.0425
GLSCL	**0.1864**	**0.1646**	**0.0734**	**0.0605**	**0.0540**	**0.0432**

Table 6
Comparison with dropout-based noise techniques.

Method	Douban-Book		Yelp		Amazon-Book	
Metric	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
Dropout-node	0.1621	0.1449	0.0650	0.0533	0.0445	0.0347
Dropout-edge	0.1719	0.1514	0.0664	0.0545	0.0476	0.0380
Random noise perturbation	**0.1770**	**0.1584**	**0.0673**	**0.0568**	**0.0489**	**0.0387**

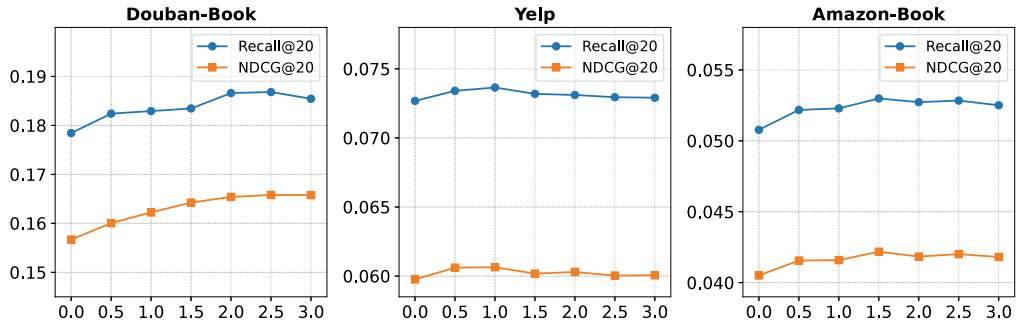


Fig. 3. The results of Recall@20 and NDCG@20 on the three datasets for balance coefficient α.

In most cases, when the alpha coefficient is set between 1.0 and 3.0, GLSCL achieves optimal performance, such as around 2.5 on the Douban-Book dataset, about 1.0 on the Yelp dataset, and about 0.5 on the Amazon-Book dataset. This is because an appropriate alpha weight aids in better embedding learning for users and items.

The alpha coefficient is positively correlated with the ratio of items to users. When the number of items in the dataset is higher than the number of users, a higher weight alpha is preferable, as seen in datasets like Douban-Book and Amazon-Book, where the number of items surpasses the number of users. In the Yelp dataset, where the number of users and items is roughly equal, an alpha coefficient of 1.0 implies that users and items are equally important in the local similarity loss. Specifically, when the hyperparameter is set around 1.0, it indicates that user and item similarity are of equal value. A value higher than 1.0 suggests that item weight is more significant, and vice versa.

Therefore, the appropriate selection of alpha coefficient plays a crucial role in influencing the performance of GLSCL, ensuring a balanced consideration of user and item embeddings in the local similarity loss.

- **Impact of the module weight λ_2.** The weight assigned to the local similarity contrastive loss in the equation also plays a crucial role in the contrastive learning task. To analyse the impact of λ_2 on GLSCL, we conducted a search for λ_2 within the range {1e−8, 1e−7, 1e−6, 1e−5, 1e−4}, and the results are depicted in Fig. 4. We observed that huge weight values on the three datasets led to a rapid performance decline. The gradual improvement in performance as parameter λ_2 increases from 1e−8 suggests that an appropriately weighted local similarity contrastive task can enhance overall recommendation performance. Optimal values for λ_2 were 1e−6 for the Douban-Book and Yelp datasets, while 1e−5 was preferable for the Amazon-Book dataset. Therefore, selecting an appropriate λ_2 weight can facilitate better coordination in multi-task learning, leading to improved recommendation effectiveness.
- **Impact of the Batch Size $\mathcal{B}$.** To investigate the impact of batch size on the local similarity contrastive task, we conducted a search with batch sizes in the range {2000, 4000, 6000, 8000, 10 000}, and the results are illustrated in Fig. 5. The findings indicate that an appropriate batch size can effectively enhance the performance of GLSCL.

  Specifically, in the local similarity contrastive task, different batch sizes determine the range for finding the most similar nodes. A too-large batch size tends to converge towards continually comparing nodes with their globally most similar nodes, resulting in a singular contrastive learning target. Conversely, a too-small batch size may select contrastive positives that lack sufficient similarity. For the Douban-Book dataset, an optimal batch size is around 6000, while for Yelp, it is about 4000. In the case of the Amazon-Book dataset, a batch size of approximately 10,000 achieves better performance, and further increases show diminishing returns while increasing training time.

  In summary, selecting an appropriate batch size is crucial for achieving optimal performance in the local similarity contrastive task, ensuring a balanced trade-off between capturing local similarities and computational efficiency.

4.5. Impact of data sparsity levels

In this section, we evaluated the performance of different user groups based on varying sparsity levels. Specifically, we divided all users into five groups based on their interaction frequencies while keeping each group's total number of interactions constant. We then

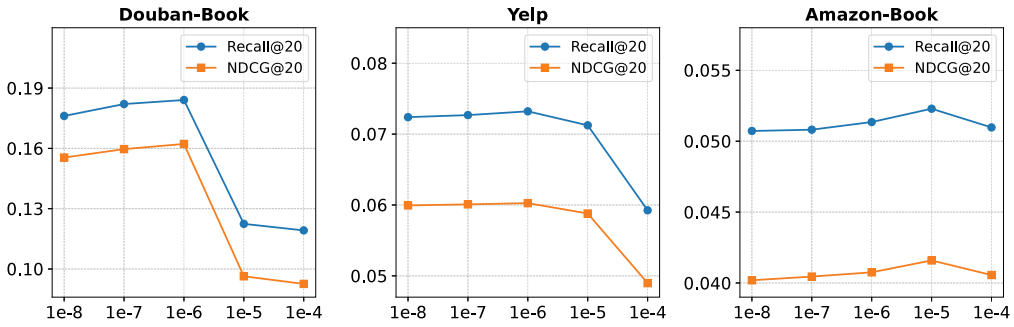


Fig. 4. The results of Recall@20 and NDCG@20 on the three datasets for module weight λ_2.

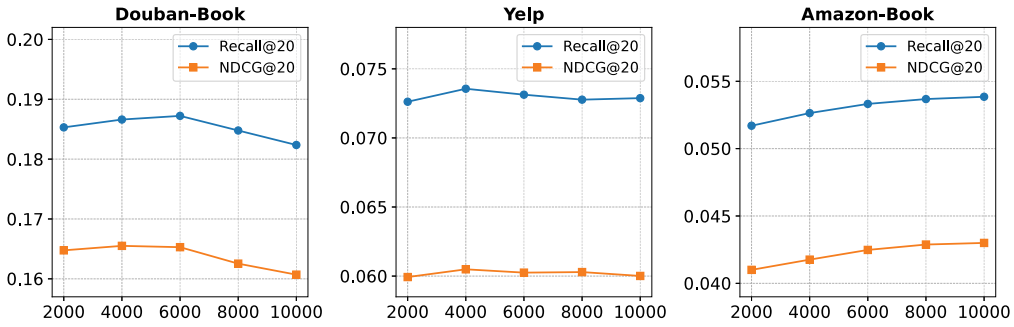


Fig. 5. The results of Recall@20 and NDCG@20 on the three datasets for batch size $\mathcal{B}$.

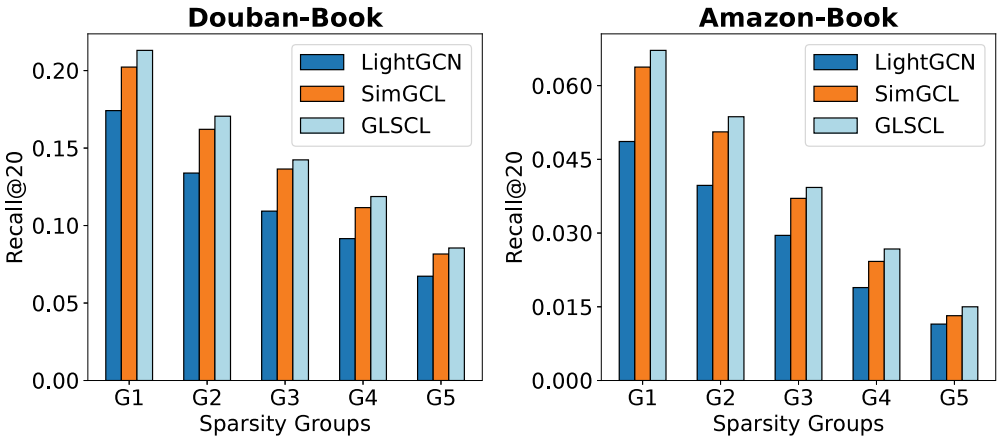


Fig. 6. Performance comparison of user groups with different sparsity levels on Douban-Book and Amazon-Book datasets (G1 indicates the group with the least number of interactions. The number of interactions from G1 to G5 increases gradually).

compared the recommendation performance across different sparsity levels. The testing set was partitioned into five user groups based on their interaction frequencies while maintaining an equal number of user interactions in each group. The minimum and maximum limits of the number of interactions with users in each group (G1 to G5) for different datasets are shown in the Table 7. The results for LightGCN, SimGCL, and GLSCL on these five user groups are illustrated in Fig. 6.

From Fig. 6, it can be observed that GLSCL outperforms LightGCN and SimGCL in all user groups. This suggests that the proposed GLSCL can mitigate the sparsity of interaction data to a greater extent as it explores local collaborative relationships and enforces global uniformity within the framework.

However, when users are grouped by sparsity in the same data set, users with fewer interactions tend to have higher recall scores. That is because users who interact less often have more focused interests, which makes it easier for the model to predict what they are likely to engage with. In contrast, users with a large number of interactions tend to have a wider range of interests, and the increased volume of interactions can include noisy or irrelevant interactions that negatively impact performance.

We attribute this advantage to the collaborative supervision capabilities between local cooperative relationship encoding and global

Table 7
Statistics of user interactions in different groups for each dataset.

DataSet	G1	G2	G3	G4	G5
Douban-Book	[1, 9]	[9, 17]	[17, 27]	[27, 47]	[47, 348]
Yelp	[1, 6]	[6, 9]	[9, 15]	[15, 29]	[29, 463]
Amazon-Book	[1, 6]	[6, 11]	[11, 20]	[20, 42]	[42, 2631]

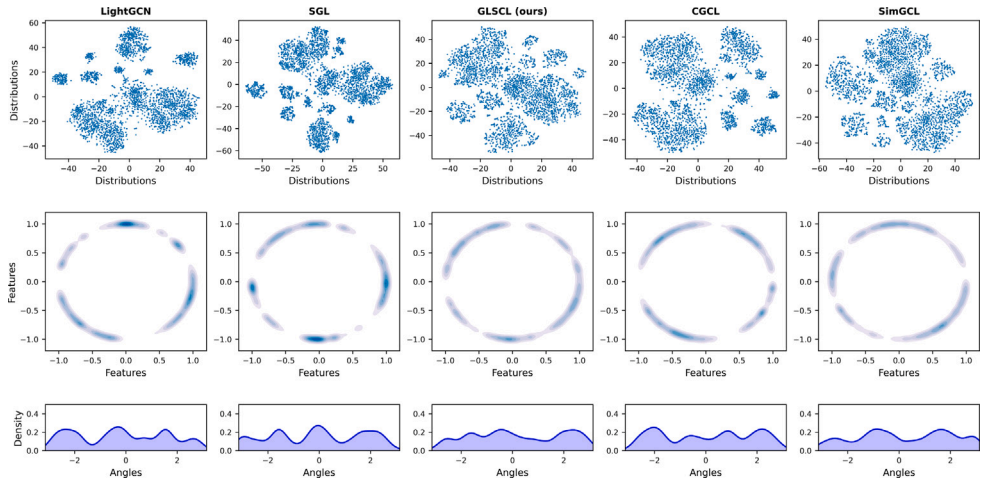


Fig. 7. The distribution of user embeddings learned from the Yelp dataset. The top of each graph is the embedded distribution visualized using t-SNE, and the middle of each graph is plotted using Gaussian kernel density estimation (KDE) of the feature distribution; the darker the colour indicates, the more points in the region. The bottom of each graph is plotted with an estimate of the Gaussian kernel density from the Angle of each point. (i.e., atan2(y,x) for each point (x,y)).

uniformity in GLSCL. It exposes the potential limitation of GCN-based methods like LightGCN, which may struggle to learn high-quality representations of user preferences solely from sparse interaction data.

4.6. Embedding visualization analysis

To provide interpretability for the outstanding performance of our model and conduct further analysis, in this section, we aim to offer more in-depth insights from the perspective of the spatial distribution of learned embedding. Specifically, we visualize the embeddings learned by our model and some baseline models using t-SNE (Van der Maaten & Hinton, 2008). We randomly sample 3000 nodes, and the visualizations of these embeddings are shown in Fig. 7. According to Fig. 7, we have several important observations:

- Our method has a more uniform embedding distribution. Yu et al. (2022) point out that a more uniform distribution is a primary reason for the performance improvement of GCL. From the KDE analysis in the visual analysis Fig. 7 (i.e. the middle and bottom figures of Fig. 7), GLSCL, CGCL, and SimGCL show greater uniformity compared to LightGCN and SGL and these performances are also better than LightGCN and SGL.
- Compared to SimGCL, our model has a clear community structure. From the top row of Fig. 7, although our distribution is overall uniform compared to SimGCL, our method can identify a clearer community structure. Our approach effectively selects positive samples within a batch, enabling the model to learn more accurate and fine-grained user and item similarity information. Our model performance is better than SimGCL, and we attribute the success of our model to this factor.
- Our model avoids small clusters that are highly differentiated and clustered. From the distribution graph results(i.e. the top figures in Fig. 7), LightGCN, SGL, and CGCL exhibit more dispersed and concentrated particle clusters, where internal embeddings are highly similar, indicating an issue of excessive smoothing. Our method ensures more reasonable dispersion within each community, better reflecting the characteristics of user-specific preferences.

5. Conclusion and future work

This paper introduced a novel contrastive learning approach called Graph Local Similarity Contrastive Learning (GLSCL). In GLSCL, we treated user–user similarity and item–item relevance as homogeneous relationships to capture unique node information. Firstly, to achieve uniformity, we adopted a random perturbation strategy to obtain uniformity in embedding distribution. Simultaneously, we calculated similarity matrices by computing similarity between nodes based on the interaction matrix to mitigate the bias introduced by random perturbation and obtain similarity relationships. This allowed us to find similar nodes within training batches for constructing positive contrastive pairs for learning. Compared to previous contrastive learning recommendation models, our extensive experiments on three public datasets demonstrated the effectiveness of the proposed GLSCL. Our research findings indicate that our approach also performs well in handling data sparsity challenges. In future work, finding similar nodes in the proposed local similarity contrastive learning module can be improved, and our model can be optimized in memory.

CRediT authorship contribution statement

Zhou Zhou: Conceptualization, Methodology, Software, Writing – original draft. **Zheng Hu:** Validation, Supervision, Writing – reviewing & editing. **Shi-Min Cai:** Resources, Supervision, Writing – reviewing & editing. **Tao Zhou:** Supervision, Writing – reviewing & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. T2293771, 42361144718, 61673086), and the Ministry of Education of Humanities and Social Science Project (Grant No. 21JZD055), and Sichuan Science and Technology Program (Grant No. 2023NSFSC1919). The funders had no role in the study design, data collection, analysis, decision to publish, or preparation of the manuscript.

Data availability

Data will be made available on request.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.

Cai, X., Huang, C., Xia, L., & Ren, X. (2023). LightGCL: Simple yet effective graph contrastive learning for recommendation. In *ICLR*. OpenReview.net.

Chen, J., Cao, Y., Zhang, F., Sun, P., & Wei, K. (2022). Sequential intention-aware recommender based on user interaction graph. In *ICMR* (pp. 118–126). ACM.

Chen, S., Chen, J., Zhou, S., Wang, B., Han, S., Su, C., Yuan, Y., & Wang, C. (2024). SIGformer: Sign-aware graph transformer for recommendation. In *SIGIR* (pp. 1274–1284). ACM.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. E. (2020). A simple framework for contrastive learning of visual representations. In *ICML* (pp. 1597–1607). IEEE.

Chen, J., Li, H., Zhang, X., Zhang, F., Wang, S., Wei, K., & Ji, J. (2024). SR-HetGNN: session-based recommendation with heterogeneous graph neural network. *Knowledge and Information Systems*, *66*(2), 1111–1134.

Cheng, Z., Lang, J., Zhong, T., & Zhou, F. (2025). Seeing the unseen in micro-video popularity prediction: Self-correlation retrieval for missing modality generation. In *KDD (1)* (pp. 142–152). ACM.

Cheng, Z., Zhang, J., Xu, X., Trajcevski, G., Zhong, T., & Zhou, F. (2024). Retrieval-augmented hypergraph for multimodal social media popularity prediction. In *KDD* (pp. 445–455). ACM.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. In *RecSys* (pp. 191–198). ACM.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, Y. E., Tang, J., & Yin, D. (2019). Graph neural networks for social recommendation. In *WWW* (pp. 417–426). ACM.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., & Tang, J. (2020). Graph random neural networks for semi-supervised learning on graphs. In *NeurIPS* (pp. 22092–22103).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS* (pp. 249–256).

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NIPS* (pp. 1024–1034).

He, X., Chen, T., Kan, M., & Chen, X. (2015). TriRank: Review-aware explainable recommendation by modeling aspects. In *CIKM* (pp. 1661–1670). ACM.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR* (pp. 639–648). ACM.

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. B. (2020). Momentum contrast for unsupervised visual representation learning. In *CVPR* (pp. 9726–9735). IEEE.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural collaborative filtering. In *WWW* (pp. 173–182). ACM.

He, W., Sun, G., Lu, J., & Fang, X. S. (2023). Candidate-aware graph contrastive learning for recommendation. In *SIGIR* (pp. 1670–1679). ACM.

Hu, Z., Li, Z., Jiao, Z., Nakagawa, S., Deng, J., Cai, S., Zhou, T., & Ren, F. (2025). Bridging the user-side knowledge gap in knowledge-aware recommendations with large language models. In *AAAI* (pp. 11799–11807). AAAI.

Hu, Z., Nakagawa, S., Luo, L., Gu, Y., & Ren, F. (2023). Celebrity-aware graph contrastive learning framework for social recommendation. In *CIKM* (pp. 793–802). ACM.

Hu, Z., Nakagawa, S., Zhuang, Y., Deng, J., Cai, S., Zhou, T., & Ren, F. (2025). Hierarchical denoising for robust social recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *37*(2), 739–753.

Jiang, Y., Huang, C., & Huang, L. (2023). Adaptive graph contrastive learning for recommendation. In *KDD* (pp. 4252–4261). ACM.

Kiran, R., Kumar, P., & Bhasker, B. (2020). DNNRec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, *144*, Article 113054.

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Li, W., Deng, J., You, J., He, Y., Zhuang, Y., & Ren, F. (2025). ETS-MM: A multi-modal social bot detection model based on enhanced textual semantic representation. In *THE web conference 2025*.

Lin, Z., Tian, C., Hou, Y., & Zhao, W. X. (2022). Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW* (pp. 2320–2329). ACM.

Lü, L., Medo, M., Yeung, C. H., Zhang, Y., Zhang, Z., & Zhou, T. (2012). Recommender systems. *Physics Reports*, *519*, 1–49.

McAuley, J. J., Targett, C., Shi, Q., & van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *SIGIR* (pp. 43–52). ACM.

Qin, Y., Wang, P., & Li, C. (2021). The world is binary: Contrastive learning for denoising next basket recommendation. In *SIGIR* (pp. 859–868). ACM.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *UAI* (pp. 452–461). AUAI Press.

Sharma, L., & Gera, A. (2013). A survey of recommendation system: Research challenges. *International Journal of Engineering Trends and Technology*, *4*(5), 1989–1992.

van den Berg, R., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. CoRR, abs/1706.02263.

van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. CoRR, abs/1807.03748.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11), 2579–2605.

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T. (2019). KGAT: knowledge graph attention network for recommendation. In *KDD* (pp. 950–958). ACM.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T. (2019). Neural graph collaborative filtering. In *SIGIR* (pp. 165–174). ACM.

Wang, J., Shi, Y., Yu, H., Zhang, K., Wang, X., Yan, Z., & Li, H. (2023). Temporal density-aware sequential recommendation networks with contrastive learning. *Expert Systems with Applications*, *211*, Article 118563.

Wang, C., Yu, Y., Ma, W., Zhang, M., Chen, C., Liu, Y., & Ma, S. (2022). Towards representation alignment and uniformity in collaborative filtering. In *KDD* (pp. 1816–1825). ACM.

Wang, H., Zeng, Y., Chen, J., Han, N., & Chen, H. (2023). Interval-enhanced Graph Transformer solution for session-based recommendation. *Expert Systems with Applications*, *213*, Article 118970.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2023). Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, *55*(5), 97:1–97:37.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021). Self-supervised graph learning for recommendation. In *SIGIR* (pp. 726–735). ACM.

Xia, L., Huang, C., Xu, Y., Zhao, J., Yin, D., & Huang, J. X. (2022). Hypergraph contrastive collaborative filtering. In *SIGIR* (pp. 70–79). ACM.

Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., & Zhang, X. (2021). Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI* (pp. 4503–4511). AAAI Press.

Xie, Z., Liu, C., Zhang, Y., Lu, H., Wang, D., & Ding, Y. (2021). Adversarial and contrastive variational autoencoder for sequential recommendation. In *WWW* (pp. 449–459). ACM / IW3C2.

Xu, X., Zhou, F., Zhang, K., & Liu, S. (2023). CCGL: contrastive cascade graph learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*(5), 4539–4554.

Yang, Z., Qin, J., Lin, C., Chen, Y., Huang, R., & Qin, Y. (2023). GANRec: A negative sampling model with generative adversarial network for recommendation. *Expert Systems with Applications*, *214*, Article 119155.

Yu, J., Xia, X., Chen, T., Cui, L., Hung, N. Q. V., & Yin, H. (2024). XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *36*(2), 913–926.

Yu, J., Yin, H., Gao, M., Xia, X., Zhang, X., & Hung, N. Q. V. (2021). Socially-aware self-supervised tri-training for recommendation. In *KDD* (pp. 2084–2092). ACM.

Yu, J., Yin, H., Li, J., Wang, Q., Hung, N. Q. V., & Zhang, X. (2021). Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *WWW* (pp. 413–424). ACM / IW3C2.

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022). Are graph augmentations necessary?: Simple graph contrastive learning for recommendation. In *SIGIR* (pp. 1294–1303). ACM.

Zheng, L., Li, C., Lu, C., Zhang, J., & Yu, P. S. (2019). Deep distribution network: Addressing the data sparsity issue for top-n recommendation. In *SIGIR* (pp. 1081–1084). ACM.

Zhou, T., Su, R.-Q., Liu, R.-R., Jiang, L.-L., Wang, B.-H., & Zhang, Y.-C. (2009). Accurate and diverse recommendations via eliminating redundant correlations. *New Journal of Physics*, *11*(12), Article 123008.

Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., & Gai, K. (2018). Deep interest network for click-through rate prediction. In *KDD* (pp. 1059–1068). ACM.