Candidate–aware Graph Contrastive Learning for Recommendation

Wei He*
Guohao Sun*
2212483@mail.dhu.edu.cn
ghsun@dhu.edu.cn
Donghua University

Jinhu Lu
Donghua University
1209126@mail.dhu.edu.cn

Xiu Susie Fang†
Donghua University
xiu.fang@dhu.edu.cn

ABSTRACT

Recently, Graph Neural Networks (GNNs) have become a mainstream recommender system method, where it captures high-order collaborative signals between nodes by performing convolution operations on the user-item interaction graph to predict user preferences for different items. However, in real scenarios, the user-item interaction graph is extremely sparse, which means numerous users only interact with a small number of items, resulting in the inability of GNN in learning high-quality node embeddings. To alleviate this problem, the Graph Contrastive Learning (GCL)-based recommender system method is proposed. GCL improves embedding quality by maximizing the similarity of the positive pair and minimizing the similarity of the negative pair. However, most GCL-based methods use heuristic data augmentation methods, i.e., random node/edge drop and attribute masking, to construct contrastive pairs, resulting in the loss of important information. To solve the problems in GCL-based methods, we propose a novel method, **C**andidate-aware **G**raph **C**ontrastive **L**earning for Recommendation, called **CGCL**. In CGCL, we explore the relationship between the user and the candidate item in the embedding at different layers and use similar semantic embeddings to construct contrastive pairs. By our proposed CGCL, we construct structural neighbor contrastive learning objects, candidate contrastive learning objects, and candidate structural neighbor contrastive learning objects to obtain high-quality node embeddings. To validate the proposed model, we conducted extensive experiments on three publicly available datasets. Compared with various state-of-the-art **DNN**-, **GNN**- and **GCL**-based methods, our proposed **CGCL** achieved significant improvements in all indicators[1].

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

*Both authors contributed equally to this research.
†Corresponding author
[1]https://github.com/WeiHeCnSH/CGCL-Pytorch-master

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591647

KEYWORDS

Recommendation System, Graph Neural Network, Contrastive Learning, Candidate.

ACM Reference Format:
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 (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, Taipei, Taiwan, 10 pages. https://doi.org/10.1145/3539618.3591647

1 INTRODUCTION

As an efficient method of information filtering, the recommender system can discover favorite items for users from massive data according to their requirements [23] and is widely applicable in news [38], advertising [14], e-commerce [49, 50], and other fields. The main task of recommender systems is predicting whether the user will interact with the item. In graph-based recommendation scenario, it can be regarded as a kind of link prediction problem [29]. As a famous method in graph-based tasks, Graph neural networks (GNN) have made significant progress with their powerful representation learning capabilities, such as node classification [19] / link prediction [43] / graph classification [42], etc. By modeling the interaction data as a graph (user-item interaction graph [2, 3, 9, 16, 34], heterogeneous graph [13], and knowledge graph [28, 33]), these GNN-based methods stack multilayer convolution to iteratively aggregate information from the local neighbor nodes to enrich the embedding of nodes. However, these GNN-based methods heavily rely on explicit supervision signals during training, which requires numerous labeled data [40, 47]. , and real-world user-item interaction records are often implicit (e.g., clicking, browsing, etc.), sparse, and noisy, resulting in a lack of explicit supervision signals [15, 26, 39]. Due to these limitations, GNN-based methods are unable to learn high-quality node embeddings.

In the past few years, contrastive learning (CL) has been applied in many scenarios, such as Computer Vision (CV) [7], and graph data mining [5]. CL can extract features of data from unlabeled data for the improvement of downstream tasks [21]. To solve the data sparsity problem in recommender systems, some researchers [15, 37, 39, 46] realize that combining GNN and CL can generate higher-quality recommendations for users. Therefore, Graph Contrastive Learning (GCL)-based recommender system methods are proposed. The key to GCL is generating appropriate anchors, and positive and negative instances through the view generator. GCL minimizes mutual information, such that, in the embedding space, anchors are close to positive instances and far away from negative instances to improve the quality of node embeddings. At

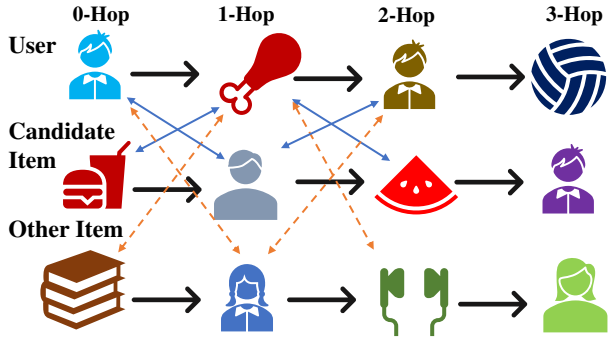


Figure 1: Illustration of the relationship between the user and the candidate item at different layers, where the blue line indicates the distance that needs to be pulled in the embedding space and vice versa in orange.

present, most GCL-based methods use heuristic-based view generators to generate instances. For example, SGL [39] generates positive instances through node/edge drop and random walk. But these methods still face the following two main problems: (1) Data augmentation with random perturbations, such as random node/edge drop, random walk, and diffusion, etc., changes the structure of the graph and cannot preserve semantic information well [6, 15, 37, 46], which makes it easy to discard important information or introduce noise and thus, resulting in poor results. (2) The heuristic-based data augmentation strategy is not universal, it requires manual intervention, and selects the appropriate augmentation strategy according to the specific application scenario, which limits the adaptation range of the model [5, 45, 46, 51]. To solve the above-mentioned problems, some scholars [15, 46] try to study view generators to avoid structural perturbation. They propose to construct the contrastive pairs from the semantic space and the structure and use the supervision signal provided by the context of the nodes.

Although some improvements have been made by GCL-based methods, they ignore the relationship between the user and the candidate item at different layers, which limits their performance. In specific, these GCL-based methods only construct contrastive pairs from the perspective of structural perturbations [21, 37, 39, 41, 45] , node similarity [5, 15, 51], or graph structure [20] and ignore the relationship between the different layer embeddings of the user and candidate item. in the embedding space, the embeddings with similar semantic information should be closer to each other than embeddings without similar semantic information. Take Figure 1 as an example, drumstick should be close to cola and watermelon and away from books and headphones in the embedding space because drumsticks, cola, and watermelon belong to the same class of the food. Establishing the connection between the user and the candidate item at different layers can better model the interest of the user and the characteristics of the item, and can better explore the intention of user interaction with the item at a finer granularity. By doing so, the collaborative signal between nodes can be effectively extracted. Therefore, to model a better relationship between the user and the candidate item, we propose a novel graph contrastive learning method, called Candidate-aware Graph contrastive Learning for Recommendation (CGCL). In summary, the main contributions can be summarized as follows:

- To model the better contrastive pairs, we propose to select the anchors, positive instances, and negative instances for contrastive learning objects from the embeddings of *different layer* of the user and the candidate item based on semantic similitude. To the best of our knowledge, we are the first to explore the relationship between the user and candidate item at different embedding layers for graph contrastive learning.
- In CGCL, we propose structured neighbor contrastive learning loss objects, the candidate contrastive learning loss objects, and the candidate structure neighbor contrastive learning loss objects to extract the low-order and high-order relationship between the user and the candidate item. The quality of the node embeddings effectively improved.
- We conduct extensive experiments on three publicly available datasets, and the results reveal that the propose CGCL can significantly improve the performance of recommendation, surpassing the state-of-the-art DNN-, GNN-, and GCL-based methods.

2 RELATED WORK

In this section, we briefly review graph neural networks and graph contrastive learning for the recommendation.

2.1 Graph neural networks

Recently, graph neural networks have been introduced into recommender systems and have made significant progress. GCMC [29] extracts the information of first-order neighbor nodes through the autoencoder. In contrast, NGCF [34] extracts the high-order connection between the user and the item by propagating the embedding of nodes in the graph. Compared with NGCF, LightGCN [9] simplifies NGCF by removing nonlinearity activation and feature transformation, which can overcome the problems faced by GNN, e.g., over-smoothing, etc. To accurately model user intent, DGCF [35] disentangles the representation of the item and user through maximum mutual information. MCCF [36] identifies potential components by decomposing edges into multiple latent spaces at the node level and uses the combiner to automatically determine the importance of different components to get the unified user and item embedding. In addition, IMP-GCN [16] further enhances LighGCN by clustering nodes with similar interests to avoid negative information propagation. To avoid the information bottleneck of depth GNN, GOTNet [2] establishes long-distance dependence between the node and neighbor nodes through non-local attention. MIXGCF [31] injects positive sample information into randomly sample negative samples through Mixup technology to synthesize hard negative samples, which effectively improves the quality of training samples. UltraGCN [18] uses constraints to estimate embeddings for nodes after infinite convolution operations. GFCF [25] proves that the neighborhood-based methods, linear auto-encoders, and low-rank matrix factorization are special cases of various classic low-pass filters. Moreover, some works[10, 28, 33, 49, 50] try to take into account information about candidates. But the relationship between the user's embedding and the candidate item's embeddings

at different layers is overlooked. Therefore, due to the data sparsity problem, these GNN-based methods cannot learn high-quality node embeddings, resulting in suboptimal recommendations.

2.2 Graph Contrastive learning

Contrastive learning (CL) has attracted wide attention in CV [7], and other fields. It effectively improves the embedding quality by minimizing the mutual information of positive pairs. As a typical self-supervise learning method, contrastive learning does not require manual labeling and can mine the intrinsic characteristics of data [14, 32] to improve the downstream task. Combining the advantages of GNN and CL, Graph contrastive learning, GCL-based recommender system methods are proposed, which can effectively alleviate the data sparsity problem and better model the interests of users. For example, to design a better loss function, SimpleX [17] proposes the cosine similarity contrastive learning loss to optimize the user and the item, but only the first-order neighbor nodes of the user are utilized. SGL [39] is a heuristic-based data augmentation method for generating contrastive instances. Specifically, SGL uses node/edge drop, and random walk to generate positive instances. AutoGCL[45] designs a view generation generator for each data augmentation method that adaptively selects node drop and attribute masking at the node level, which overcomes the adaptability problem of data augmentation. GCA [51] proposes a joint adaptive data augmentation method in topology and node attribute level by removing edge and mask features to provide different context information for nodes in different views, which effectively explores the impact of different data augmentation methods. However, random edge/node drop and perturbation may lead to the loss of important information. Therefore, CGI [37] proposes to construct the more practical contrastive instances by selectively dropping edge/node. Differing from SGL, NCL [15] takes similar semantic prototypes in the embedding space and the structure neighbors as positive instances. Moreover, SimGCL [46] experimentally found that the contribution of data augmentation to SGL was very small and then propose to add noise to each layer embedding to generate positive instances. In order to avoid manually constructing contrastive instances, SimGRACE [41] uses different graph encoders as the view generator and compares the semantic similarity between the instances obtained after perturbing two different encoders, while LightGCL [1] uses matrix factorization to reconstruct the user-item interaction graph as the contrastive instances.

However, the current methods are limited by the quality of the instances generated, thus they are often complex, cumbersome, inefficient, and difficult to train [1, 39, 40, 46, 47]. These disadvantages limit them to obtain high-quality node embeddings.

3 PROBLEM AND BACKGROUND

In this section, we will give the definition of the problem and a brief introduction to GNN. Meanwhile, we also introduce the LightGCN [9] backbone.

3.1 Preliminary

Given the set of users $\mathcal{U} = \{u\}$ and items $\mathcal{I} = \{i\}$. Let $\mathcal{I}^+$ and $\mathcal{I}^-$ (i.e., $\mathcal{I} - \mathcal{I}^+$) denote items that users have interacted with and items that have not observed interaction records, respectively. The user-item interaction matrix $\mathbb{A} \in \mathbb{R}^{M \times N}$ can be constructed according to historical interaction records, where M and N indicate the number of users and items, respectively. Set $A_{ui} \in \{0, 1\} = 1$ (observed item) if user u has an interaction with item i. Otherwise, Set $A_{ui} \in \{0, 1\} = 0$ (unobserved). 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 $\mathcal{E} = \{A_{ui} | A_{ui} = 1\}$ represents observed interactions. CGCL is to predict whether the user u will interact with the unobserved item i by the given input graph $\mathcal{G}$.

3.2 Graph neural network in recommendation

We will briefly introduce GNN. Specifically, GNN has several key components: embedding, aggregation, propagation, readout, prediction, and optimization.

3.2.1 Embedding. A vector with dimension d is used to represent the initialization embedding of the user (item) $e_u^{(0)} \in \mathbb{R}^d$ ($e_i^{(0)} \in \mathbb{R}^d$) , where d is the size of the embedding, which can be obtained via embedding lookup tables and can be represented as follows:

$$e_u^{(0)} = lookup\,(u)\,, \quad e_i^{(0)} = lookup\,(i)\,. \tag{1}$$

where u and i represent the IDs of the user and item, respectively.

3.2.2 Aggregation. GNN aggregates the information of neighboring nodes by performing convolution operations. The embedding of neighboring nodes in the l layer can be represented as follows:

$$\begin{aligned} e_{u,N}^{(l)} &= Agg\left(e_i^{(l-1)}, i \in N_u\right), \\ e_{i,N}^{(l)} &= Agg\left(e_u^{(l-1)}, u \in N_i\right). \end{aligned} \tag{2}$$

where $e_u^{(l)}$ and $e_i^{(l)}$ indicate the l-layer embedding redefined by the user u and item i, respectively, and N_u and N_i indicate the neighbors that directly interact with the user and item, respectively.

3.2.3 Propagate. GNN performs information propagation that uses the embedding of aggregate neighbor nodes to update its own embedding, which can be represented as follows:

$$\begin{aligned} e_u^{(l)} &= Prop\left(e_u^{(l-1)}, e_{u,N}^{(l)}\right), \\ e_i^{(l)} &= Prop\left(e_i^{(l-1)}, e_{i,N}^{(l)}\right). \end{aligned} \tag{3}$$

3.2.4 Readout. After L time convolutional operations, GNN adopts the pooling techniques to aggregate the embeddings of nodes at different depths to generate the final representation of the user and item, which can be represented as follows:

$$\begin{aligned} e_u &= Readout\left(e_u^{(0)}, \cdots, e_u^{(L)}\right), \\ e_i &= Readout\left(e_i^{(0)}, \cdots, e_i^{(L)}\right). \end{aligned} \tag{4}$$

The readout function can be weight summation [9, 34], etc.

3.2.5 Prediction. The likelihood of interaction between the user u and the item i can be represented as follows:

$$\hat{y}_{ui} = f\,(e_u, e_i)\,. \tag{5}$$

where $f\,(\cdot)$ is a scoring prediction function be used to calculate the probability of interaction between the user u and the item i, which can be the inner product [9, 16, 34], MLP[27], etc.

3.2.6 Optimize. The parameters of the model are optimized by minimizing the loss function, which can be represented as follows:

$$\mathcal{L}_{Rec} = g\left(y_{ui}, \hat{y}_{ui}\right). \tag{6}$$

For the recommendation task, the loss function can be Bayesian Personalized Ranking (BPR) loss [22], Triplet loss [24], etc.

3.3 LightGCN Backbone

In CGCL, we use LightGCN [9] as the backbone of GNN. As the GCN architecture introduce earlier, LighGCN uses IDs mapping to obtain the embedding of nodes. However, complex nonlinear activation and feature transformation are not valid for one-hot encode IDs, they are omitted in LightGCN. The aggregation process of LightGCN can be represented as follows:

$$\begin{aligned} e_{u,N}^{(l+1)} &= \textstyle\sum_{i\in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^{(l)}, \\ e_{i,N}^{(l+1)} &= \textstyle\sum_{u\in N_i} \frac{1}{\sqrt{|N_i||N_u|}} e_u^{(l)}. \end{aligned} \tag{7}$$

In LightGCN, since the self-node already appears in the graph, the self-loop is further omitted. The update node embedding can be directly represented using the embedding of neighboring nodes. The process of propagating information can be represented as follows:

$$e_u^{(l+1)} = e_{u,N}^{(l+1)}, \quad e_i^{(l+1)} = e_{i,N}^{(l+1)}. \tag{8}$$

After obtaining the node embeddings at each layer, LightGCN leverages average pooling to generate the final embedding of the nodes. The readout function of LightGCN can be represented as follows:

$$e_u = \tfrac{1}{L+1} \textstyle\sum_{l=0}^{L} e_u^{(l)}, \quad e_i = \tfrac{1}{L+1} \textstyle\sum_{l=0}^{L} e_i^{(l)}. \tag{9}$$

The likelihood of interaction between the user u and the item i can be represented as follows:

$$\hat{y}_{ui} = e_u^T e_i. \tag{10}$$

The widely used BPR [22] loss function is adopted to optimize the model, which assumes that positive samples score higher than negative samples, which can be represented as follows:

$$\mathcal{L}_{Rec} = \frac{1}{|O|} \sum_{(u,i,j)\in O} -\log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right). \tag{11}$$

where O indicates the training set, u indicates the user, i indicates the positive sample, j indicates the randomly select the negative sample from items not interacting with the user u, and σ indicates the sigmoid activation function.

4 THE PROPOSED METHOD

In this work, we aim to explore the relationship between the user and the candidate item to improve the quality of node embeddings learned by GNN. The proposed CGCL model is shown in Figure 2, which consists of three components: Firstly, to establish the relationship between structural neighbors and the center node, we propose the structural neighbor contrastive learning objects in section 4.1. Secondly, to establish the relationship between users and candidate items, we propose the candidate contrastive learning objects in section 4.2. Thirdly, to establish the relationship between the structural neighbors of users and candidate items, we propose the candidate structural neighbor contrastive learning objects in section 4.3. The details of each component are described below.

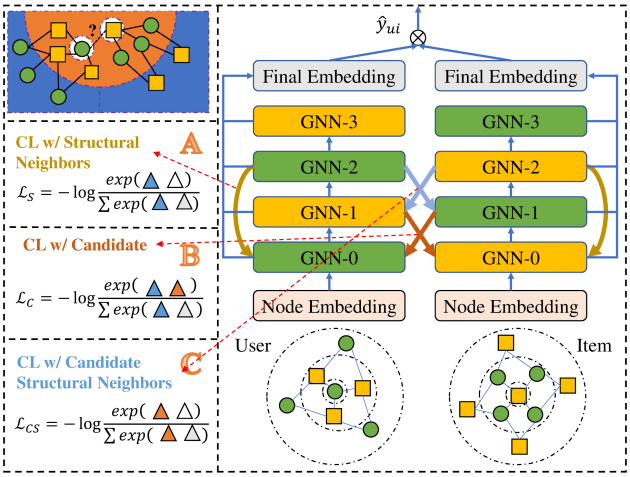


Figure 2: The proposed CGCL method, where the circle represents the user, the square represents the item, and the triangle represents the embedding at each layer. The subgraph on the left is the proposed three contrastive learning objects on the user side, and the subgraph on the right is the GNN backbone.

4.1 Contrastive Learning with Structural Neighbors

In this subsection, we will introduce structural neighbor contrastive learning objects, which correspond to part A in Figure 2. The user-item interaction graph is a typical bipartite graph that contains two types of nodes (the user node and the item node). The types of neighbor nodes in each layer are the same. As shown in Figure 1, on the user side, nodes at the even layer $(0, 2, \cdots)$ are user types, and nodes at the odd layer $(1, 3, \cdots)$ are item types. These homogeneous user (item) nodes are connected through convolution propagation. They can be considered to have similar semantic information and are closer to each other in the embedding space.

Therefore, similar to [15], we take the embedding of the central node itself as the anchor, the homogeneous neighbor node embedding of the central node as the positive instances, and the homogeneous neighbor node embedding of other central nodes as the negative instances. Based on the InfoNCE [30] loss, we propose the structural contrastive learning objects to minimize the distance between homogeneous neighbor nodes. The structural neighbor contrastive learning loss function on the user side can be represented as follows:

$$\mathcal{L}_S^U = \sum_{u\in\mathcal{U}} -\log \frac{\exp\left(sim\left(e_u^{(k)}, e_u^{(0)}\right)/\tau\right)}{\sum_{v\in\mathcal{U}} \exp\left(sim\left(e_u^{(k)}, e_v^{(0)}\right)/\tau\right)}. \tag{12}$$

where $e_u^{(k)}$ is the embedding of the k-layer convolution on the user side, k is the even number, sim indicates cosine similarity, and τ is the temperature hyperparameter of softmax.

Similarly, the structural neighbor contrastive learning loss function on the item side can be represented as follows:

$$\mathcal{L}_S^I = \sum_{i \in I} -\log \frac{\exp\left(sim\left(e_i^{(k)}, e_i^{(0)}\right)/\tau\right)}{\sum_{j \in I} \exp\left(sim\left(e_i^{(k)}, e_j^{(0)}\right)/\tau\right)}. \tag{13}$$

where $e_i^{(k)}$ represents the embedding of the k-layer convolution on the item side. Combining these two losses, we get the objective function of structural neighbor contrastive learning loss as follows:

$$\mathcal{L}_S = \alpha \mathcal{L}_S^U + (1-\alpha)\,\mathcal{L}_S^I. \tag{14}$$

where α is the hyperparameter to control the strengths of two different structural neighbor contrastive learning objects.

4.2 Contrastive Learning with Candidate

In this subsection, we will introduce candidate contrastive learning objects, which correspond to part B of Figure 2. For the given user u and candidate item i, the goal of the recommender system is to predict whether there will be an interaction between them. According to the basic assumption of the recommender system, Similar users may interact with the same item. If user u and candidate item i has a higher interaction possibility, then the user should be similar to the historical interaction user of the candidate item in the embedding space. In addition, The user u shouldn't be similar to historical interaction users of other items in the embedding space.

Therefore, on the user side, we take the embedding of user u as the anchor, the first-order neighbor node embedding of the candidate item i as the positive instances, and the first-order neighbor node embedding of the other items as the negative instances. The candidate contrastive learning loss on the user side can be represented as follows:

$$\mathcal{L}_C^U = \sum_{i \in I} -\log \frac{\exp\left(sim\left(e_i^{\left(k'\right)}, e_u^{(0)}\right)/\tau\right)}{\sum_{v \in \mathcal{U}} \exp\left(sim\left(e_i^{(k')}, e_v^{(0)}\right)/\tau\right)}. \tag{15}$$

where $e_i^{\left(k'\right)}$ represents the embedding of the item-side at the $k^{'}$-layer, and $k^{'}$ is the odd number.

Similarly, on the candidate item side, the historical interaction items of user u are closer to the candidate item i in embedding space, while the historical interaction items of other users are far away from the candidate item i.

Therefore, we use the embedding of the candidate item i as the anchors, the first-order neighbor node embedding of user u as the positive instances, and the first-order neighbor node embedding of any users different from u as the negative instances. The candidate contrastive learning loss on the item side can be represented as follows:

$$\mathcal{L}_C^I = \sum_{v \in \mathcal{U}} -\log \frac{\exp\left(sim\left(e_u^{\left(k'\right)}, e_i^{(0)}\right)/\tau\right)}{\sum_{j \in I} \exp\left(sim\left(e_u^{(k')}, e_j^{(0)}\right)/\tau\right)}. \tag{16}$$

where $e_u^{\left(k'\right)}$ represents the embedding of the user-side at the $k^{'}$-layer. Combining these two losses above, we get the objective function of candidate contrastive learning loss as follows:

$$\mathcal{L}_C = \alpha \mathcal{L}_C^U + (1-\alpha)\,\mathcal{L}_C^I. \tag{17}$$

4.3 Contrastive Learning with Candidate Structure Neighbors

In this subsection, we will introduce candidate structure neighbor contrastive learning objects, which corresponds to part C of Figure 2. The structural neighbors of users (items) contain items (users) with similar interests. The second-order neighbors of the user and the first-order neighbors of the item are the same type of nodes (type of user). They will be connected when performing multi-layer convolution, and there is a long-distance dependence relationship between them. If user u may interact with candidate item i, then the user-type neighbor nodes of the user u are closer to the user-type neighbor nodes of the candidate item i in the embedding space. Meanwhile, the item-type neighbor nodes of the user u are also close to the item-type neighbor node of the item i in the embedding space. Therefore, we choose the homogeneity neighbor node of the user (item) as the positive instances of the item (user). In addition, the relationship between the center node and the neighbor nodes that are farther away from the center node is weak, because this kind of relationship contains more noise and irrelevant information.

Therefore, we use the embeddings of neighbor nodes that are closer to the center node as the anchors, the embeddings of neighbor nodes that are farther away from the center node as the positive instances, and the embeddings of structural neighbors of other interaction pairs that are farther away from the center node as the negative instances. The candidate structure neighbor contrastive learning loss on the user side can be represented as follows:

$$\mathcal{L}_{CS}^U = \sum_{i \in I} -\log \frac{\exp\left(sim\left(e_i^{(k)}, e_u^{\left(k'\right)}\right)/\tau\right)}{\sum_{v \in \mathcal{U}} \exp\left(sim\left(e_i^{(k)}, e_v^{(k')}\right)/\tau\right)}. \tag{18}$$

Similarly, on the item side, we use the embeddings of structure neighbor nodes (user types) of the candidate item as the anchors, the embeddings of neighbor nodes (item types) of the user as the positive instances, and the embeddings of neighbor nodes (item types) of other users as the negative instances. The candidate structure neighbor contrastive learning loss on the item side can be represented as follows:

$$\mathcal{L}_{CS}^I = \sum_{v \in \mathcal{U}} -\log \frac{\exp\left(sim\left(e_u^{(k)}, e_i^{\left(k'\right)}\right)/\tau\right)}{\sum_{j \in I} \exp\left(sim\left(e_u^{(k)}, e_j^{(k')}\right)/\tau\right)}. \tag{19}$$

Combining these two losses above, we get the objective function of candidate structure neighbor contrastive learning loss as follows:

$$\mathcal{L}_{CS} = \alpha \mathcal{L}_{CS}^U + (1-\alpha)\,\mathcal{L}_{CS}^I. \tag{20}$$

4.4 Optimization with CGCL

Same as the other methods [15, 37, 39, 46], we treat three contrastive learning losses as auxiliary losses and use the multi-task learning strategy to jointly optimize the BPR [22] loss and the contrastive learning loss function. So the total loss function can be represented as follows:

$$\mathcal{L}_{CGCL} = \mathcal{L}_{Rec} + \lambda_1 \mathcal{L}_S + \lambda_2 \mathcal{L}_C + \lambda_3 \mathcal{L}_{CS} + \lambda_4 \|\Theta\|_2^2 . \quad (21)$$

where λ_1, λ_2, and λ_3 are hyperparameters to control the weights of three contrastive learning losses, respectively. λ_4 is the regularization coefficient, and Θ indicates the trainable parameters.

Algorithm 1 Candidate–aware Graph Contrastive Learning

Input: User-Item bipartite graph $\mathcal{G} = \{\mathcal{U} \cup \mathcal{I}, \mathcal{E}\}$, training dataset $\mathcal{X}$;
Parameter: Trainable parameters $\{e\}_{u \in \mathcal{U}}, \{e\}_{i \in \mathcal{I}}$, Hyperparameter $\alpha, \lambda_1, \lambda_2, \lambda_3, \lambda_4$;
Output: Trained Model $\mathcal{F}(e_u, e_i | \Theta, \mathcal{G})$;

while CGCL Not Convergence **do**
  for x in $Dataloader(\mathcal{X})$ **do**
    // Forward propagation
    $e_u, e_i = GNN\left(\mathcal{G}, e_u^{(0)}, e_i^{(0)}\right)$;
    // Calculate Loss
    Calculate BPR loss $\mathcal{L}_{Rec}$;
    Calculate contrastive learning with structural neighbors loss $\mathcal{L}_S$;
    Calculate contrastive learning with candidate loss $\mathcal{L}_C$;
    Calculate contrastive learning with candidate structure neighbors loss $\mathcal{L}_{CS}$;
    Calculate total loss $\mathcal{L}_{CGCL}$;
    // Back propagation
    $e_u = e_u - \alpha \frac{\partial \mathcal{L}}{\partial e_u}$;
    $e_i = e_i - \alpha \frac{\partial \mathcal{L}}{\partial e_i}$;
  end for
end while
return $\mathcal{F}(e_u, e_i | \Theta, \mathcal{G})$;

5 EXPERIMENTS

In this section, we will conduct extensive experiments on three publicly available datasets and compare the performance with state-of-the-art DNN-, GNN- and GCL-based methods to validate our proposed model, and our goal is to answer the following questions:

- Q1: How does the proposed CGCL perform compared to state-of-the-art **DNN**-, **GNN**- and **GCL**-based methods?
- Q2: How do different parameter settings affect the performance of CGCL?
- Q3: How do data of different sparsity affect the performance of CGCL?
- Q4: How do different parts of CGCL contribute to the final performance?
- Q5: How does CGCL perform when applied to other GNN backbones?
- Q6: What is the effect of CGCL on improving the embedding distribution of nodes?

DataSets	#Users	#Items	#Interactions	#Density
Yelp	45,478	30,709	1,777,765	0.00127
Gowalla	29,859	40,989	1,027,464	0.00084
Books	58,145	58,052	2,517,437	0.00075

Table 1: Statistics of the datasets

5.1 Experimental Setup

5.1.1 Datasets. To fairly evaluate the proposed CGCL, we experiment on the following three publicly available datasets, **Yelp** [2], **Gowalla** [3], **Amazon Books** [4]. These datasets are provided by NCL [15], and to ensure a fair comparison, we follow the same treatment as NCL. The statistics of the dataset are summarized in Table 1. Similar to NCL, the training, validation, and test sets are divided into 8:1:1, respectively. The normal distribution is used to randomly sample one negative sample for each positive sample to form a training pair.

5.1.2 Baseline Models. We compare CGCL with the state-of-the-art **DNN** (NeuMF, DMF), **GNN** (GCMC, NGCF, LightGCN), and **GCL** (SGL, Simplex, NCL) methods. These methods are described as follows:

NeuMF [11]: NeuMF integrates the matrix decomposition and multilayer perceptron (MLP) to extract the low-order and high-order features of the nodes to establish a complex relationship between the user and the item.
DMF [44]: DMF simulates matrix decomposition directly on the user-item interaction matrix using multi-layer neural networks.
GCMC [29]: GCMC treats matrix completion as a link prediction problem on the graph, and the user-item interaction graph was reconstructed using an auto-encoder.
NGCF [34]: NGCF propagates embeddings in the user-item interaction graph to establish the high-order connection between nodes.
LighGCN [9]: LightGCN simplifies the NGCF by removing nonlinearity activation, feature transformations, and self-loop.
SimpleX [17]: SimpleX aggregates the embedding of the user and the first-order neighbor node by weight summation and uses cosine contrastive loss to optimize the embedding of the user and the item.
SGL [39]: SGL constructs different contrastive learning views by node drop, edge drop, and random walk in the user-item interaction graph to improve the performance of the recommender system. As with NCL, we use the best SGL-ED as the implementation of SGL.
NCL [15]: NCL uses LightGCN as the backbone and considers the homogeneous structure neighbors and semantic prototypes of the nodes as the positive instances of contrastive learning.

5.1.3 Metrics. Same as other GNN-based methods [9, 15, 16, 34, 39], we use two widely used metrics to evaluate the performance of the TOP-N recommendation: Recall@N and NDCG@N, where N is set to 20 and 50, respectively. We use the all-ranking protocol to calculate all metrics [8], and treat all items that the user has not interacted with as the negative sample.

[2] https://www.yelp.com/dataset
[3] https://snap.stanford.edu/data/loc-gowalla.html
[4] http://jmcauley.ucsd.edu/data/amazon/

DataSet	Yelp				Gowalla				Amazon-Books			
Metric	Recall		NDCG		Recall		NDCG		Recall		NDCG	
Method	20	50	20	50	20	50	20	50	20	50	20	50
NeuMF	0.0572	0.1457	0.0328	0.0510	0.1167	0.2255	0.0753	0.0969	0.0530	0.1216	0.0339	0.0486
DMF	0.0660	0.1253	0.0353	0.0507	0.1025	0.1706	0.0591	0.0759	0.0542	0.1007	0.0293	0.0414
GCMC	0.0908	0.1690	0.0494	0.0696	0.1472	0.2410	0.0810	0.1038	0.0868	0.1553	0.0481	0.0658
NGCF	0.0953	0.1764	0.0519	0.0729	0.1576	0.2546	0.0893	0.1130	0.0902	0.1598	0.0495	0.0676
LightGCN	0.1218	0.2123	0.0690	0.0741	0.1920	0.2988	0.1133	0.1192	0.1201	0.2002	0.0691	0.0737
SGL-ED	0.1335	0.2177	0.0808	0.1028	0.2153	0.3277	0.1281	0.1558	0.1418	0.2172	0.0866	0.1065
SimpleX	0.1221	0.2044	0.0728	0.0941	0.1555	0.2601	0.0784	0.1040	0.1339	0.2143	0.0790	0.1001
NCL	0.1394	0.2258	0.0816	0.1044	0.2147	0.3299	0.1260	0.1543	0.1389	0.2209	0.0818	0.1034
CGCL	**0.1457**	**0.2404**	**0.0849**	**0.1097**	**0.2205**	**0.3391**	**0.1292**	**0.1583**	**0.1538**	**0.2400**	**0.0920**	**0.1147**
Imp (%).	4.52	6.47	4.04	5.08	2.42	2.79	0.86	1.60	8.46	8.65	6.24	7.70

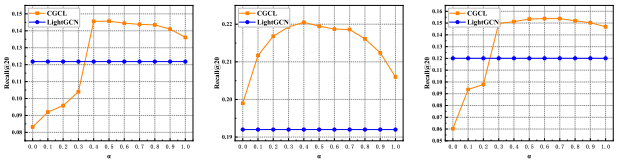
Table 2: The overall performance comparison of the proposed CGCL model with the state-of-the-art DNN-, GNN-, and GCL-based baselines. The optimal value is bolded and the suboptimal value is underlined.

5.1.4 Implementation Details. All baselines and our proposed CGCL are implemented using Recbole [48]. In Recbole, the loss function of GCMC [29] is replaced by the cross-entropy loss function. To ensure a fair comparison, all parameters are initialized with the default Xavier [4], and the embedding size and mini-batch are set to 64 and 2048, respectively. The GNN backbone of the GCL-based method defaults to LightGCN, and model depths are set to 3. The default learning rate is 1e-3, all methods are optimized using Adam [12], and hyperparameters are carefully searched for all baselines. Similar to [9, 15, 34], we carefully adjust the parameters λ_1, λ_2, and λ_3 in $[1e-4, 1e-8]$, λ_4 is set to 1e-4, and τ in $[0.05, 0.2]$. The embedding farther away from the center contains more noise and irrelevant information, which can introduce false learning signals. Therefore, The $k^{'}$ and k are set to 1 and 2, respectively. The early stop strategy is used to avoid overfitting. Where, we use Recall@20 as an indicator and stop training after the indicator drops 10 consecutive rounds below the optimal value on the validation set. The average results of five experiments on the test set are reported.

5.2 Performance Comparison with State-of-the-Arts (Q1)

The detailed performance comparison of the proposed CGCL and other state-of-the-art DNN-, GNN-, and GCL-based methods on three datasets is reported in Table 2. Some conclusions can be drawn from these results.

Our proposed CGCL achieves the best performance on all datasets, significantly exceeding state-of-the-art DNN-, GNN-, and GCL-based methods. For example, CGCL achieves an absolute improvement of more than 1% in Recall@50 compared to the two strongest baselines (SGL and NCL) on all datasets. We think these baselines are limited by the following shortcomings: (1) The DNN-based methods only use the information of the node self (NeuMF) or the first-order neighbor node (DMF), which cannot capture the high-order connection between the nodes, resulting in difficulty extracting enough information from the user-item interaction. (2) The GNN-based methods are limited by the data sparsity problem and cannot learn high-quality node embeddings. (3) The existing



(a) Yelp. (b) Gowalla. (c) Amazon-Books.

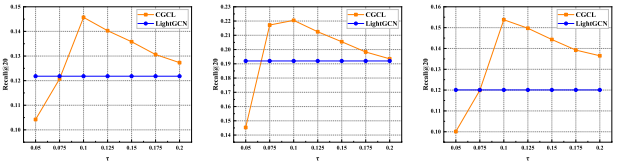
Figure 3: Performance Comparison w.r.t. Different Balance Coefficient α.

GCL-based methods don't explore the relationship between the user and the candidate item at different layers. In addition, they construct inappropriate contrastive pairs, such that they are unable to reasonably optimize the embedding of nodes. These experimental results sufficiently prove the superiority of CGCL. We speculate that CGCL can optimize the quality of node embeddings more reasonably by establishing the relationship between the embeddings of the user and candidate item at different layers, such that the candidate item with high interaction possibilities is close to the user and the item with low interaction probabilities is far away.

5.3 Parameter Analyses (Q2)

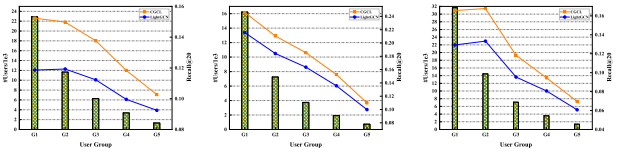
In this section, we will research the effects of different parameters (Balance Coefficient α and temperature τ) on CGCL.

5.3.1 Impact of the Balance Coefficient α. To study the effect of the balance coefficient α, we adjust the range of α, and the results are shown in Figure 3. We can observe the following phenomena: (1) In most cases, when the balance coefficient α is around 0.5, CGCL achieves the best performance because the appropriate balance coefficient can help the user and the item learn better embeddings. (2) When reducing the balance coefficient α, that is, decreasing the importance of user-side contrastive learning loss, we find that the performance on the Yelp and Amazon-Books datasets decreased significantly, which indicates that the user-side contrastive learning loss is more important for GCCL than the item-side contrastive



(a) Yelp. (b) Gowalla. (c) Amazon-Books.

Figure 4: Performance Comparison w.r.t. Different Temperature τ.



(a) Yelp. (b) Gowalla. (c) Amazon-Books.

Figure 5: Performance Comparison (Recall@20) w.r.t. Different sparsity-level user groups on three different datasets (The average number of interactions from the G1 group to the G5 group are gradually increasing).

learning loss. (3) When increasing the balance coefficient α, the performance on the Yelp and Amazon-Books datasets stays stable, which can be explained by the following two aspects: (a) We calculate evaluation metrics using the all-ranking protocol, which searches for the most likely interact item from all unobserved items. In this case, the embedding of the optimization items has less impact on the results. (b) The first-order neighbor of the user is the item-type node. Therefore, the item-type nodes are optimized in the candidate contrastive learning objects, and success compensates for the impact of the decreasing importance of the item-side contrastive learning loss. To conclude, CGCL can achieve good results even under the influence of a large balance coefficient.

5.3.2 Impact of the Temperature τ. To study the effect of temperature τ on CGCL, we adjust the range of τ, and the results are shown in Figure 4. We observe that the temperature τ has played a vital role in the performance of CGCL. When the temperature τ is between 0.075 and 0.125, CGCL achieves the best results. Based on [32], small temperature coefficient models can better focus on difficult sample learning so that similar user (item) node embeddings are separated from each other in the embedding space. But excessive forcing to separate difficult samples can lead to the destruction of underlying semantic structures. The high temperature will make it difficult to separate hard negative instances from positive instances, resulting in low-quality embeddings. Therefore, choosing an appropriate temperature can make the embedding space distribution more uniform and achieve better results.

5.4 Impact of Data Sparsity Levels (Q3)

The data sparsity problem severely limits the representation ability of GNN. In this section, we will investigate whether the proposed method can effectively alleviate this problem. Similar to [15], we divide all users of the test set into five groups based on the number of user interactions and kept the total number of user interactions for each group the same. The results of LightGCN and CGCL on five groups of users are shown in Figure 5. From the figure, we can see that CGCL has achieved better results than LighGCN on all user groups, which indicates that exploring the relationship between the user and candidate item can effectively alleviate the data sparsity problem and enrich the semantic information of nodes. In addition, we found that with the number of interactions decreasing, CGCL achieve better results than LighGCN and made a greater performance gain (e.g., an about 28% improvement on the Yelp and Amazon-Books datasets in the G1 group), indicating that CGCL is more suitable for sparse interacting data and can provide users with

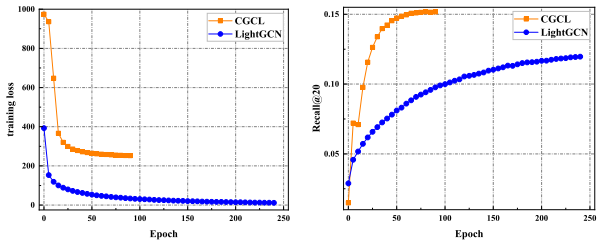


Figure 6: Training curves and test Recall@20 of LightGCN and CGCL for the sparsest Amazon-Books dataset(Take one result every five rounds)..

DataSet	Yelp		Gowalla		Amazon-Books	
Metric	R@20	N@20	R@20	N@20	R@20	N@20
LightGCN	0.1218	0.069	0.1942	0.1123	0.1201	0.0691
O s	0.1328	0.0761	0.2028	0.1191	0.1236	0.0712
O c	0.1342	0.0768	0.2061	0.1195	0.1343	0.0780
O cs	0.1331	0.0755	0.2058	0.1202	0.1262	0.0729
W/o s	0.1327	0.0755	0.2056	0.1177	0.1259	0.0727
W/o c	0.1412	0.0824	0.2025	0.1184	0.1521	0.0910
W/o cs	0.1447	0.0846	0.2197	0.1291	0.1534	0.0908
All	0.1457	0.0849	0.2205	0.1292	0.1538	0.0920

Table 3: Ablation Experimental

a better recommendation in real-world scenarios. Figure 6 shows the training curves of training loss and the result of Recall@20 on the validation set. Due to page limitations, we only show results for the sparsest dataset, Amazon Books. We found that CGCL took fewer epochs to reach convergence than LighGCN. This is because CGCL explores the relationship between the user and the candidate item, and the proposed contrastive learning objects can better explore hard negative samples, providing a larger gradient during training so that the speed of training is accelerated.

5.5 Ablation Experimental (Q4)

The proposed CGCL consists of three parts: (1) structural neighbor contrastive learning objects, (2) candidate contrastive learning objects, and (3) candidate structured neighbor contrastive learning objects. To verify the contribution of each part to the CGCL, we

DataSet	Yelp		Gowalla		Amazon-Books	
Metric	R@20	N@20	R@20	N@20	R@20	N@20
NGCF	0.0953	0.0519	0.1576	0.0893	0.0902	0.0495
+CGCL	0.1084	0.0609	0.1812	0.1048	0.1259	0.0731
Imp(%).	13.75	17.34	14.97	17.36	39.58	47.68
LightGCN	0.1218	0.0690	0.1942	0.1123	0.1201	0.0691
+CGCL	0.1457	0.0849	0.2205	0.1292	0.1538	0.0920
Imp(%).	19.62	23.04	13.54	15.05	27.73	32.56

Table 4: Performance comparison w.r.t. different backbones.

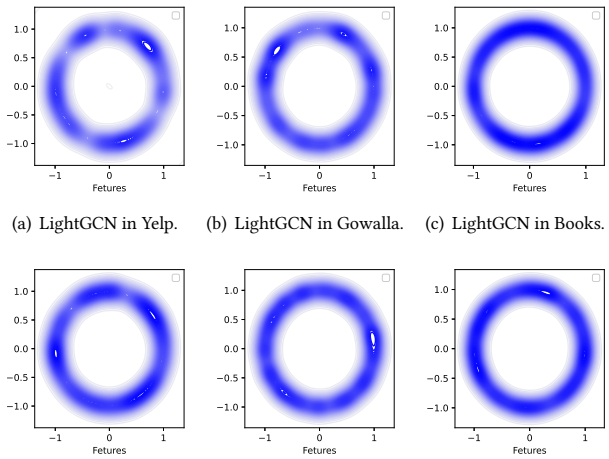
make ablation experiments. The experimental results are reported in Table 3. where O s, O c, and O cs refer only to retaining (1), (2), and (3) from the model, respectively. Similarly, W/o s, W/o c, and W/o cs refer to removing (1), (2), and (3) from the model, respectively. We can find that removing or only retaining part of the structure of the models can achieve better performance than LighGCN on all datasets, which indicates that the three contrastive learning objects proposed are all valid and can improve the quality of node embeddings learned by LighGCN.W/o s obtains the performance of preserving only a part of the structure of the model, which indicates the interdependence between the different components. In addition, we found that keeping two or all parts of the model can achieve better results than using only one. We speculate that using only one of the contrastive learning objects will lead to the learned embeddings being too close together. In contrast, using all the contrastive learning objects can make the embedding distribution of the nodes more reasonable.

5.6 Different GNN Backbones (Q5)

To further verify the performance of CGCL, the influence of different GNN backbones was analyzed. We compare the performance with different GNN backbones. The results are reported in Table 4. CGCL has greatly improved the performance on different GNN backbones, For example, on the Amazon-Books dataset, CGCL based on the LightGCN backbone achieves an improvement of nearly 30%, and CGCL based on the NGCF backbone achieved an improvement of more than 40%. This is because the introduced contrastive learning objects can improve the embedding quality and alleviate the disadvantages of GNN-based methods due to the lack of labeled training data. In addition, we find that CGCL achieves better results on the LightGCN backbone than on the NCGF backbone. This can be explained by the following two aspects: (1) LighGCN removes nonlinear activation and feature transformation from NGCF that are not conducive to model learning. (2) We think the more important thing is that LighGCN further removes the self-loop in NGCF. In CGCL, we choose embeddings with similar semantic information to construct contrastive pairs, but NGCF with the self-loop makes embeddings contain the information of two different property nodes, which decreases the semantic similarity of embeddings.

5.7 Visualizing Analysis (Q6)

To better understand CGCL and analyze whether the proposed three contrastive learning objects can improve the node embedding quality, we use Gaussian kernel density estimation (KDE) to visualize the embedding distribution of items in two-dimensional space



(a) LightGCN in Yelp. (b) LightGCN in Gowalla. (c) LightGCN in Books.

(d) CGCL in Yelp. (e) CGCL in Gowalla. (f) CGCL in Books.

Figure 7: Feature distribution of item representations learn from the datasets in $\mathbb{R}^2$ (The darker the color, the more items fall within that area).

and compare it with LighGCN. The results are shown in Figure 7. The node embeddings learned by LighGCN on the Yelp and Gowalla datasets fall into several fixed regions, and the node embeddings learn on the Amazon-Books dataset are approximately evenly distributed and cannot highlight the importance. However, the node embeddings learned by CGCL on the three datasets are uniformly distributed and prominently important, reflecting the alignment and uniformity of contrastive learning. This shows that the CGCL can improve the quality of node embeddings by establishing the relationship between the user and candidate item in different layers of embedding and better modeling the preferences of different users and the characteristics of items.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel graph contrastive learning method, called CGCL. In CGCL, we propose to explore the relationship between the user and the candidate item and use similar semantic embeddings to construct contrastive pairs. First, in order to establish the relationship between nodes and structural neighbors, we propose structural neighbor contrastive learning objects. Then, in order to establish the relationship between the user and candidate item, we propose candidate contrastive learning objects. Finally, in order to establish the relationship between users and the structural neighbors of the candidate item, we propose candidate structural neighbor contrastive learning objects.

ACKNOWLEDGMENTS

This work was supported by Shanghai Science and Technology Commission (No. 22YF1401100), Fundamental Research Funds for the Central Universities (No. 22D111210, 22D111207), and National Science Fund for Young Scholars (No. 62202095).

REFERENCES

[1] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. *CoRR* (2023).

[2] Huiyuan Chen, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2022. Graph Neural Transport Networks with Non-local Attentions for Recommender Systems. In *WWW*. 1955–1964.

[3] 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 *AAAI*. 27–34.

[4] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[5] Hakim Hafidi, Mounir Ghogho, Philippe Ciblat, and Ananthram Swami. 2020. GraphCL: Contrastive Self-Supervised Learning of Graph Representations. *CoRR* (2020).

[6] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML*. 4116–4126.

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[8] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware Explainable Recommendation by Modeling Aspects. In *CIKM*. 1661–1670.

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[10] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Trans. Knowl. Data Eng.* 12 (2018), 2354–2366.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[13] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. 2020. Hierarchical Fashion Graph Network for Personalized Outfit Recommendation. In *SIGIR*. 159–168.

[14] Guogang Liao, Ze Wang, Xiaoxu Wu, Xiaowen Shi, Chuheng Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. Cross DQN: Cross Deep Q Network for Ads Allocation in Feed. In *WWW*. 401–409.

[15] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning. In *WWW*.

[16] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware Message-Passing GCN for Recommendation. In *WWW*. 1296–1305.

[17] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In *CIKM*. 1243–1252.

[18] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *CIKM*. 1253–1262.

[19] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*.

[20] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*. 259–270.

[21] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[23] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. 1–35.

[24] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. 815–823.

[25] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, Khaled B. Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *CIKM*. 1619–1629.

[26] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-graph Convolution Collaborative Filtering. In *ICDM*. 1306–1311.

[27] Yijun Tian, Chuxu Zhang, Zhichun Guo, Chao Huang, Ronald A. Metoyer, and Nitesh V. Chawla. 2022. RecipeRec: A Heterogeneous Graph Learning Model for Recipe Recommendation. In *IJCAI*. 3466–3472.

[28] Ke Tu, Peng Cui, Daixin Wang, Zhiqiang Zhang, Jun Zhou, Yuan Qi, and Wenwu Zhu. 2021. Conditional Graph Attention Networks for Distilling and Refining Knowledge Graphs in Recommendation. In *CIKM*. 1834–1843.

[29] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* (2017).

[30] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* (2018).

[31] Vikas Verma, Meng Qu, Kenji Kawaguchi, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. 2021. GraphMix: Improved Training of GNNs for Semi-Supervised Learning. In *AAAI*. 10024–10032.

[32] Feng Wang and Huaping Liu. 2021. Understanding the Behaviour of Contrastive Loss. In *CVPR*. 2495–2504.

[33] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*. 2000–2010.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[35] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *SIGIR*. 1001–1010.

[36] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2020. Multi-Component Graph Convolutional Collaborative Filtering. In *AAAI*. 6267–6274.

[37] Chunyu Wei, Jian Liang, Di Liu, and Fei Wang. 2022. Contrastive Graph Structure Learning via Information Bottleneck for Recommendation. In *NIPS*.

[38] Chuhan Wu, Fangzhao Wu, Tao Qi, Qi Liu, Xuan Tian, Jie Li, Wei He, Yongfeng Huang, and Xing Xie. 2022. FeedRec: News Feed Recommendation with Various User Feedbacks. In *WWW*. 2088–2097.

[39] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *SIGIR*. 726–735.

[40] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *TKDE* (2021).

[41] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. 2022. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *WWW*. 1070–1079.

[42] Jiaxing Xu, Jinjie Ni, Sophi Shilpa Gururajapathy, and Yiping Ke. 2022. A Class-Aware Representation Refinement Framework for Graph Classification. *CoRR* (2022).

[43] Siyong Xu, Cheng Yang, Chuan Shi, Yuan Fang, Yuxin Guo, Tianchi Yang, Luhao Zhang, and Maodi Hu. 2021. Topic-aware Heterogeneous Graph Neural Network for Link Prediction. In *CIKM*. 2261–2270.

[44] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. 3203–3209.

[45] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. 2022. AutoGCL: Automated Graph Contrastive Learning via Learnable View Generators. In *AAAI*. 8892–8900.

[46] 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 *SIGIR*. 1294–1303.

[47] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2022. Self-Supervised Learning for Recommender Systems: A Survey. *CoRR* (2022).

[48] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *CIKM*. 4722–4726.

[49] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *CoRR* (2018).

[50] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. 2017. Deep Interest Network for Click-Through Rate Prediction. *CoRR* (2017).

[51] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*. 2069–2080.