General-Special Neighborhood-Enhanced Graph Contrastive Learning for Recommendation

Zezheng Yang
School of Computer Science and Technology
DongHua University
Shanghai, China
starplucking_yzz@163.com

Xiaoling Xia*
School of Computer Science and Technology
DongHua University
Shanghai, China
sherlysha@dhu.edu.cn

***Abstract*—Graph Neural Networks (GNNs) have gained widespread application in the field of recommendation systems due to their flexibility and effectiveness in modeling graph-structured data. GNN enable the collection of high-dimensional information from user-item interaction graphs through multiple layers of convolution and information propagation, facilitating a holistic modeling of users and items. However, GNN-based models face the challenge of over-smoothing due to sparsity, which hinders their ability to learn optimal node representations. To address this issue, graph contrastive learning algorithms have been introduced, yielding promising results. Yet, these graph contrastive learning-based methods tend to overlook the historical interactions.**

To tackle this problem, we propose a novel method called General and Specific neighborhood-enhanced Graph Contrastive Learning (GS-GCL). This method approaches node neighbor discovery and contrastive learning from both general and specific perspectives. We observe that in GNN even-pass information propagation encodes homogenous information that can be extracted as the general neighbor representation of the central nodes. Furthermore, we calculate the cosine similarity of a node's historical interactions to select the most similar nodes as its specific neighbors. To validate the effectiveness of our method, we conducted extensive experiments on three publicly available datasets. The results demonstrate significant improvements compared to the currently most effective method NCL method, particularly achieving a remarkable 14.14% enhancement on the Amazon-Books dataset. Our code repository can be found at: https://github.com/big-maomi/GS-GCL

Keywords-component; Recommender System, Collaborative Filtering, Contrastive Learning, Graph Neural Network

I. INTRODUCTION

In this age of information overload, recommendation systems have become indispensable key technology in both work and daily life [1], widely applied across various scenarios, including news [2] and movie recommendations [3], advertising suggestions, and e-commerce platforms [4]. On one hand, they can identify users' personal interests and recommend items that match their preferences, thus enhancing user experiences. On the other hand, recommendation systems can generate significant economic benefits for companies.

User-item interaction data can be naturally understood as a graph structure. Recently, researchers have introduced Graph Neural Network (GNN) into recommendation systems, achieving substantial advancements [5,6,7]. GNN aggregate information and update nodes in a way that allows for the comprehensive assimilation of information from the entire graph, greatly enriching the embeddings of nodes and significantly improving recommendation effectiveness. However, due to sparsity [8] and noise in the dataset, GNN-based methods often fail to learn the best node representations in specific scenarios, leading to the problem of over-smoothing.

To address this issue, contrastive learning has been introduced to GNN-based methods [9,10,11,12]. Contrastive learning is an unsupervised learning method that requires selecting suitable anchors, identifying relevant samples as positive instances and unrelated samples as negative instances. The goal is to minimizing the distance between anchors and positive instances while maximize the distance between anchors and negative examples to learn node representations. This technique has found widespread applications in pre-training models.

While graph contrastive learning methods have made some progress, they often focus on the graph structural perspective [11] and tend to overlook historical interactions.

From a historical interaction perspective, we can delve deeper into exploring the special neighbors of a node. For instance, in Fig. 1, both u_1 and u_2 have purchased many of the same items, which suggests that u_1 and u_2 can be regarded as similar users. On the other hand, since u_1 and u_3 have not purchased the same items, it's reasonable to consider that u_1 and u_3 are dissimilar users.

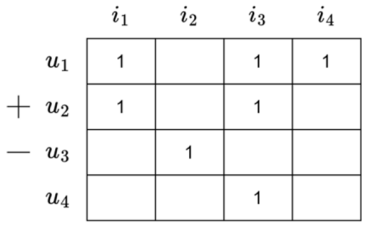

Figure 1. Special neighbors

In detail, we can identify special neighbor nodes by calculating the cosine similarity of historical interaction records for each node, such as the item interaction records between two user nodes.

The key contributions of the method we propose are as follows:

979-8-3503-5993-0/23/$31.00 ©2023 IEEE

- We have introduced a graph contrastive learning algorithm named GS-GCL, which enhances model accuracy by creating contrastive pairs from both a general and specific perspective.
- In GS-GCL, we introduce the concepts of general neighbor contrastive learning objects and specific neighbor contrastive learning objects, enabling a more effective utilization of both graph structural information and historical interactions. This allows the model to learn better node representations.
- We conducted extensive experiments on three widely adopted public datasets, and the results consistently indicate the superiority of our approach over several competitive baseline methods, including recommendation techniques based on Deep Neural Network (DNN), Graph Neural Network (GNN) and Graph Contrastive Learning (GCL)

II. PRELIMINARY

Here, we provide a brief introduction to the principle of the Graph Neural Network (GNN)-based model.

Given a user set U and an item set I, we can construct a user-item interaction matrix $R \in \{0,1\}^{|U| \times |I|}$ based on historical interactions. When $R_{ui} = 1$ it signifies an interaction between the user and the item; otherwise, there is no interaction.

GNN constructs a graph using R. Let $V = \{U \cup I\}$ be the set of nodes, which includes both the user and item sets. E represents the set of edges, defined as $E = \{R_{ui} \mid R_{ui} = 1\}$.

Typically, the GNN-based methods derive informative user and item representations can be delineated into three sequential stages: aggregation, propagation, readout.

$$e_u^l = f_{prop}\left(e_u^{l-1}, f_{aggr}\left(\left\{e_i^{l-1} \mid i \in N_u\right\}\right)\right) \tag{1}$$

$$e_u = f_{readout}\left(e_u^0, \cdots, e_u^L\right) \tag{2}$$

where e_u^l denotes the embedding of node u at the l-th layer, e_u^0 denotes the ID embedding of node. N_u represents the set of items that the user has purchased. The aggregation functions can aggregate the information of neighbors in the previous layer. The propagation function can propagate the information obtained by the aggregation function and its own representation information of the previous layer to the next layer. Through L layers, we can obtain the representation of nodes at each layer. The readout function can synthesize the representations of nodes at different layers to generate the final representation of the node.

III. METHODOLOGY

In this section, we will introduce our proposed contrastive learning method enhanced with general and specific neighbors in three segments.

The first segment will provide an introduction to the classical GNN-based method LightGCN [5]. Subsequently, the second and third segments will delve into the general neighbor contrastive learning method and the specific neighbor contrastive learning method, with the goal of identifying the optimal anchors and positive-negative instances for contrastive learning. In the fourth segment, we will amalgamate the graph neural network method with the general neighbor contrastive learning and specific neighbor contrastive learning approaches to facilitate a more robust acquisition of node embeddings.

A. Backbone: LightGCN

The LightGCN is the backbone of GC-GCL. LightGCN is an outgrowth of NGCF [6], discarding feature transformations and non-linear activation functions, which were utilized in NGCF. This streamlined approach significantly reduces computational complexity while simultaneously enhancing precision.

The propagation of LightGCN as follows:

$$e_u^{l+1} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^l \tag{3}$$

$$e_i^{l+1} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i||N_u|}} e_u^l \tag{4}$$

After undergoing L layers of propagation, we obtain the L+1 representations of the nodes. Here, we use average function as readout function to compute the final representation of the nodes.

$$e_u = \frac{1}{L+1} \sum_{l=0}^{L} e_u^l \tag{5}$$

$$e_i = \frac{1}{L+1} \sum_{l=0}^{L} e_i^l \tag{6}$$

Hence, we acquire the final representation of the nodes. In the subsequent step, we employ inner products to compute the likelihood of interactions between user u and item i.

$$\hat{y}_{ui} = e_u^T e_i \tag{7}$$

In this case, we utilize the Bayesian Personalized Ranking (BPR) loss [15] as the model's loss function, as depicted below:

$$L_{BPR} = \sum_{(u,i,j) \in O} -\log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) \tag{8}$$

where σ denotes sigmoid function. O indicates train data, the train data comprises a series of triplets (u, i, j), where $R_{ui} = 1, R_{uj} = 0$

B. Contrastive Learning with General Neighbors

As depicted in the Fig. 2, we observe that the first-order neighbors of a node belong to a different node type, while the second-order neighbors pertain to nodes of the same type. For instance, the first-order neighbors of a user node are item nodes, and the second-order neighbors are other user nodes. We have a strong rationale to believe that the user node and its second-order neighbors exhibit similarity, as they have collectively purchased the same item. In this context, we designate the second-order neighbors as the general neighbors of a user.

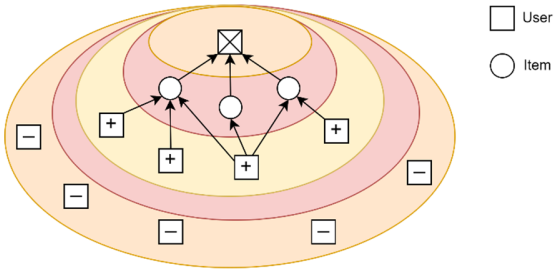

Figure 2. General neighbors

Moreover, taking into account graph neural networks, even-order information propagation on the graph naturally aggregates information from nodes with homogeneous structural neighbors. We can use e_u^k, e_i^k (k = 0,2,4…) as general neighbors. Then we use the initial embedding of the central node as anchors, the general neighbors of the central node as positive samples, other nodes as negative samples.

According to InfoNCE [13]. The loss for general neighbor contrastive learning on the user side can be expressed as:

$$L_G^U = \sum_{u \in U} -\log \frac{\exp\left(\left(e_u^k \cdot e_u^0 / \tau\right)\right)}{\sum_{u \in U} \exp\left(\left(e_u^k \cdot e_u^0 / \tau\right)\right)} \tag{9}$$

where e_u^k denotes the representation of node user at the k-th layer, k is even number, τ is the temperature hyper-parameter of softmax function.

The loss for general neighbor contrastive learning on the item side can be expressed as:

$$L_G^I = \sum_{i \in I} -\log \frac{\exp\left(\left(e_i^k \cdot e_i^0 / \tau\right)\right)}{\sum_{j \in I} \exp\left(\left(e_i^k \cdot e_j^0 / \tau\right)\right)} \tag{10}$$

The overall general neighbor contrastive learning loss is obtained by taking a weighted sum of the user-side and item-side general neighbor contrastive learning losses:

$$L_G = L_G^U + \alpha L_G^I \tag{11}$$

where α is the weight of the loss of general contrastive learning loss in the item side.

C. *Contrastive Learning with Special Neighbors*

The general neighbor contrastive loss generally explores the neighbors of the central node. However, since there is no direct connection between the general neighbor nodes and the central node, all neighbors are treated equally, which inevitably introduces noise and makes the model lack focus.

To address this issue, we consider identifying specific neighbors of a central node to extend the contrastive pairs. Taking inspiration from userCF [14] and itemCF [15], we can use co-occurrence information to calculate the cosine similarity between the central node and all homogenous nodes. The node with the highest similarity is chosen as the specific neighbor node. We can utilize the initial features of the central node as the anchor, the initial features of the specific neighbor as the positive sample, and the initial features of other nodes as negative samples.

The function for computing similarity is as follows:

$$\cos(x, y) = \frac{x \bullet y}{\|x\|\|y\|} \tag{12}$$

where x and y donate different nodes historical interactions, x and y are item history vectors.

For each node, we need to compute the nearest node neighbors based on their historical interactions.

$$w = \arg\max_v \cos\left(h_u, h_v\right), v \epsilon U, v \neq u \tag{13}$$

$$k = \arg\max_j \cos\left(h_i, h_j\right), j \epsilon I, j \neq i \tag{14}$$

where h_u is the historical interactions of user u , w is the index of most similar node of user, k is the index of most similar node of item.

The loss for special neighbor contrastive learning on the user side can be expressed as:

$$L_S^U = \sum_{u \in U} -\log \frac{\exp\left(\left(e_u^0 \cdot e_w^0 / \tau\right)\right)}{\sum_{v \in U} \exp\left(\left(e_u^0 \cdot e_v^0 / \tau\right)\right)} \tag{15}$$

The loss for special neighbor contrastive learning on the item side can be expressed as:

$$L_S^I = \sum_{i \in I} -\log \frac{\exp\left(\left(e_i^0 \cdot e_k^0 / \tau\right)\right)}{\sum_{j \in I} \exp\left(\left(e_i^0 \cdot e_j^0 / \tau\right)\right)} \tag{16}$$

The overall special neighbor contrastive learning loss is obtained by taking a weighted sum of the user-side and item-side special contrastive learning losses:

$$L_S = L_S^U + \alpha L_S^I \tag{17}$$

D. *Optimization*

We will train the model by combining the loss from the graph neural network method, the general neighbor contrastive learning loss, and the specific neighbor contrastive learning loss. The loss function can be expressed as follows:

$$L = L_{BPR} + \lambda_1 L_S + \lambda_2 L_G + \lambda_3 \|\Theta\|_2 \tag{18}$$

where Θ indicates the parameters of model, $\lambda_1, \lambda_2, \lambda_3$ indicate weights of the contrastive objects and the model parameters.

IV. Experiments

In this section, we carried out comprehensive experiments on three commonly used public datasets and conducted relevant analyses.

A. Datasets

To better evaluate GS-GCL, we compared it with the currently most effective method NCL. As shown in the Table I, we selected three datasets used by NCL, Gowalla [16], Yelp, and Amazon-Books [17]. We adopted the same data splitting method as NCL, with an 8:1:1 ratio for the training set, validation set, and test set. During the training process, for each positive instance, we consistently sampled a negative item to create the training set, allowing us to train using Bayesian Personalized Ranking (BPR).

TABLE I. DATASET INFO

Dataset Info	Datasets		
	Yelp	*Gowalla*	*Amazon-books*
Users	45,478	29,859	58,145
Items	30,709	40,989	58,052
Interactions	1,777,765	1,027,464	2,517,437
Density	0.00127	0.00084	0.00075

B. Compared Models

We selected multiple models for comparison, including models based on Deep Neural Networks (DNN), Graph Neural Networks (GNN), and Graph Contrastive Learning (GCL).

1) DNN-based models

a) NeuMF [18] combines matrix factorization and multilayer perceptron to capture representations of nodes,

b) BPRMFF [19] uses pairwise ranking to capture the order of positive interactions over negative ones.

c) FISM [20] combines the power of deep learning with traditional matrix factorization to capture complex user-item interactions

2) GNN-based models

a) DGCF [21] leverages GNN and MF to capture user-item interactions.

b) NGCF [6] utilizes embeddings in the user-item interaction graph to establish intricate connections between nodes that span multiple layers or higher-order relationships.

c) LightGCN [5] simplifies the NGCF model by removing the non-linear activation functions and feature transformations.

3) GCL-based models

a) SGL [11] enhances recommender system performance by creating various contrastive learning perspectives through techniques like node drop and random walk within the user-item interaction graph.

b) NCL [9] utilizes LightGCN as its foundational framework and conducts contrastive learning from both structural neighbors and semantic neighbors.

C. Metrics

We utilized two commonly used metrics, Recall@N and NDCG@N, with N values set at 10, 20, and 50.

D. Implementation Details.

All methods, including baseline comparison methods and our proposed GS-GCL, are implemented through Recbole [22]. Specifically, we employ the Adam optimizer with a batch size of 4096 and set the embedding size to 64. We use NDCG@10 as the evaluation metric and stop training if the performance remains lower than the best value for 10 batches.

E. Overall Performance

From the Table II, it's evident that methods GNN-based methods outperform those DNN-based methods, indicating that GNN are more effective at modeling the data by leveraging message passing to gather global information and obtain better node representations. GCL-based methods significantly outperform GNN-based methods, highlighting the utility of introducing contrastive learning in GNN-based approaches

Furthermore, across all three datasets, the proposed GS-GCL method consistently outperforms other approaches. GS-GCL exhibits superior performance compared to baseline methods, particularly in NDCG@10 and Recall@10, which are crucial in practical applications.

F. Analysis of experimental results for GS-GCL

In this subsection, we further analyze the experimental results of GS-GCL on three datasets

1) Ablation Study of GS-GCL.

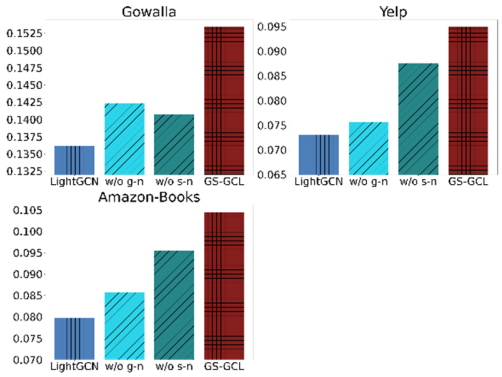

Figure 3. Performance of GS-GCL without general neighbors and special neighbors Recall@10

As shown in the Fig. 3, "w/o g-n" represents using only special neighbor contrastive learning, while "w/o s-n" represents using only general neighbor contrastive learning.

Through ablation experiments, we can observe that whether using general neighbor contrastive learning or special neighbor contrastive learning alone, the performance is better than the baseline method LightGCN. When both general and special neighbor contrastive learning are used simultaneously, it leads to superior results compared to using a single neighbor contrastive learning. From these findings, we can conclude that the two neighbor contrastive learning methods complement each other and achieve better results

TABLE II. PERFORMANCE COMPARISON OF GC-GCL AND OTHER MODELS

Dataset	Metric	Models									
		DNN-based models			GNN-based models			GCL-based models			
		BPRMF	NeuMF	FISM	NGCF	DGCF	LightGCN	SGL	NCL	GS-GCL	Imp (%).
Yelp	Recall@10	0.0652	0.0532	0.0712	0.0632	0.0719	0.0731	0.0834	0.0921	0.0949	3.15
	NDCG@10	0.0461	0.0376	0.0516	0.0447	0.0512	0.0521	0.0603	0.0679	0.0715	5.46
	Recall@20	0.1039	0.0886	0.1114	0.1027	0.1136	0.1164	0.1291	0.1378	0.1391	1.02
	NDCG@20	0.0582	0.0487	0.0637	0.0566	0.0639	0.0656	0.0741	0.0818	0.085	4.04
	Recall@50	0.1863	0.1656	0.1966	0.1866	0.1991	0.2015	0.2143	0.2246	0.2226	-0.93
	NDCG@50	0.0794	0.0687	0.0857	0.0785	0.0871	0.0877	0.0957	0.1045	0.1071	2.39
Amazon-Books	Recall@10	0.0605	0.0509	0.0723	0.0621	0.0741	0.0796	0.0901	0.0935	0.1045	12
	NDCG@10	0.0432	0.0353	0.0508	0.0431	0.0533	0.0564	0.0652	0.0674	0.0775	14.14
	Recall@20	0.0948	0.0824	0.1101	0.0975	0.1134	0.1207	0.1342	0.1382	0.15	8.62
	NDCG@20	0.0529	0.0452	0.0625	0.0536	0.0632	0.0691	0.0787	0.0817	0.0913	12.02
	Recall@50	0.1679	0.1448	0.1833	0.1703	0.1909	0.2016	0.2161	0.2173	0.2307	6.07
	NDCG@50	0.0725	0.0616	0.0817	0.0721	0.0846	0.0902	0.0994	0.1021	0.1128	10.16
Gowalla	Recall@10	0.1157	0.1042	0.1069	0.1198	0.1255	0.1371	0.1471	0.1501	0.1534	2.27
	NDCG@10	0.0835	0.0731	0.0759	0.0851	0.0901	0.0875	0.1051	0.1073	0.1102	1.85
	Recall@20	0.1697	0.1535	0.1611	0.1749	0.1828	0.1974	0.2086	0.2131	0.2175	1.97
	NDCG@20	0.0979	0.0873	0.0922	0.1017	0.1065	0.1156	0.1232	0.1265	0.1286	1.66
	Recall@50	0.2749	0.251	0.2669	0.2814	0.2876	0.3047	0.3189	0.3256	0.3322	1.93
	NDCG@50	0.1452	0.111	0.1169	0.1275	0.1325	0.1413	0.1498	0.1543	0.1568	1.69

2) Impact of the Temperature

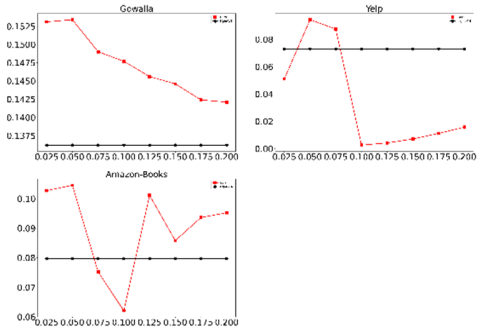

Figure 4. Impact of the Temperature

To analyze the impact of the temperature coefficient, we conducted multiple experiments with different values. The results, as shown in the Fig. 4, indicate that using excessively large τ values leads to suboptimal performance. Meanwhile, when the temperature coefficient is set to be around 0.050, the model exhibits better accuracy.

3) Impact of the alpha

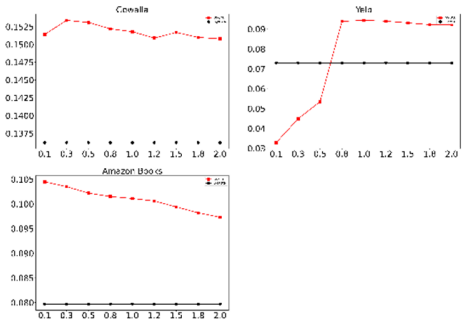

Figure 5. Impact of the alpha

To analyze the impact of α, we conducted multiple experiments with different values. The results, as depicted in the Fig. 5, show that setting the hyperparameter α to be around 0.8 leads to the best performance, indicating that both users and items are valuable for the contrastive learning algorithm.

V. CONCLUSIONS

In this paper, we introduced a recommendation model based on general and special neighbor contrastive learning. We explored neighbors from both general and special perspectives,

using the central node as the anchor. By minimizing the distance between the central node and its general and special neighbors and maximizing the distance from other nodes, we effectively leveraged graph structure information and historical data. Our experiments on three widely-used public datasets demonstrated promising results.

In the future, we are contemplating the utilization of multiple similar specialized neighbors for contrastive learning to further enhance the enrichment of node information, enabling the model to acquire more robust encoding.

REFERENCES

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

[2] Huang, and Xing Xie. 2022. FeedRec: News Feed Recommendation with Various User Feedbacks. In WWW. 2088‑2097.

[3] 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).

[4] 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).

[5] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 639–648.

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

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

[8] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. 2020. Self-supervised learning for deep models in recommendations. arXiv e-prints (2020).

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

[10] 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.

[11] 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

[12] 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.

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

[14] Goldberg, David, et al. Using collaborative filtering to weave an information tapestry.Communications of the ACM 35.12 (1992): 61-71.

[15] Linden, Greg, Brent Smith, and Jeremy York. Amazon. corn Recommenders: Item-to-item collaborative filtering. IEEE Internet computing I (2003): 76-80.

[16] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 1082–1090.

[17] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 43–52.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web. 173–182

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

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

[21] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. 1001–1010.

[22] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In Proceedings of the 30th ACM International Conference