Adversarial Learning Data Augmentation for Graph Contrastive Learning in Recommendation

Junjie Huang[1,2] , Qi Cao[1] , Ruobing Xie[3] , Shaoliang Zhang[3] , Feng Xia[3] , Huawei Shen[1,2] , Xueqi Cheng[1,4]

[1]Data Intelligence System Research Center,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]WeChat, Tencent, Beijing, China
[4]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{huangjunjie17s, caoqi, shenhuawei, cxq}@ict.ac.cn,
{ruobingxie, modriczhang, xiafengxia}@tencent.com

May 18, 2023

Outline

Introduction

Related Work

GNN-based Recommendation

- Matrix Factorization (MF) methods: BPRMF (Rendle et al., 2012), DMF (Xue et al., IJCAI2017), NeuMF (He et al., WWW2017)
- Auto-encoder (AE) methods: Mult-VAE (Liang et al., WWW2018)
- Graph Neural Networks (GNNs): NGCF (Wang et al., SIGIR2019), LightGCN (He et al., SIGIR2020), DGCF (Wang et al., WWW2020)
- Most GNN methods in recommender system follow the message-passing scheme (Gilmer et al., ICML2017) to utilize the bipartite graph structure.

Contrastive Learning in Recommendation

- Contrastive Learning (CL) as a self-supervised manner, has been applied in Recommender Systems (RS), including SSL+DNN (Yao et al., CIKM2021), SGL (Wu et al., SIGIR2021), SimGCL (Yu et al., SIGIR2022), NCL (Lin et al., WWW2022).
- Graph Contrastive Learning (GCL) is often used to alleviate the **data sparsity** and **popularity bias** problem.

Outline

GNN-based CF I

Bipartite Graph in Recommendation

- As the fundamental recommender system, collaborative filtering (CF) can be modelled as a user-item bipratite graph as $G = (\mathcal{U}, \mathcal{I}, \mathcal{E})$, where $\mathcal{U}$ is the user set, $\mathcal{I}$ is the item set and $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{I}$ is the inter-set edges.
- $\mathcal{E}$ can be denoted as the user-item interaction matrix $\mathbf{R} \in \{0,1\}^{|\mathcal{U}| \times |\mathcal{I}|}$. The adjacency matrix $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix}$ is also widely used in He et al. (2020).

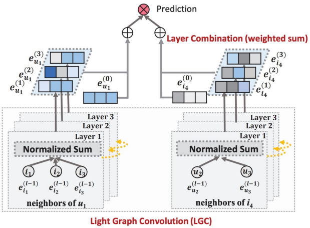
GNN-based Collaborative Filtering

- Based on the bipartite graph $\mathbf{A}$, the general GNN-based CF methods follow the message-passing scheme:

$$z_w^l = f_{\text{aggregate}}\left(\left\{z_v^{l-1} \mid v \in \mathcal{N}_w \cup \{w\}\right\}\right), z_w = f_{\text{update}}\left(\left[z_w^0, z_w^1, \ldots, z_w^L\right]\right),$$

where $\mathcal{N}$ denotes the neighbor set of node w and L denotes the number of GNN layers.

LightGCN



- LightGCN (He et al., SIGIR2020) applies a simple weighted sum aggregator:

$$Z^{l+1} = \left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right) Z^l, Z = \frac{1}{L+1}(Z^0 + Z^1 + \cdots + Z^L),$$

where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ is the diagonal matrix and Z^0 is initial trainable embeddings.

GNN-based CF III

Loss Function

- Most GNN-based CF methods (e.g., NGCF (Wang et al., SIGIR2019), DGCF (Wang et al., WWW2020), and LightGCN (He et al., SIGIR2020)) use the pairwise Bayesian Personalized Ranking (BPR) loss function for the model training:

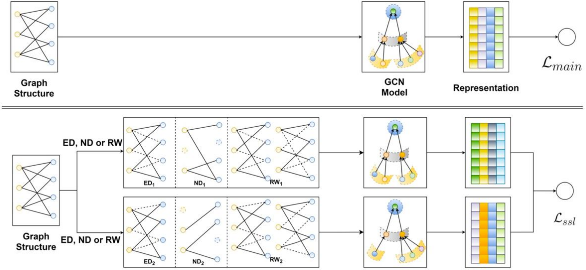
$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i,j)\in\mathcal{O}} -\log\sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right),$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{O}^+, (u,j) \in \mathcal{O}^-\}$, $\mathcal{O}^+$ and $\mathcal{O}^-$ are the observed and unobserved interactions, respectively.

Graph Contrastive Learning in Recommendation

Data Augmentation

- Common data augmentation is the perturbation of the graph structure due to the absence of node features.(e.g., Edge-dropping (Wu et al., SIGIR2021).)
- **InfoMin** principle that the good set of views shares the *minimal* ir



SGL (Wu et al., SIGIR2021)

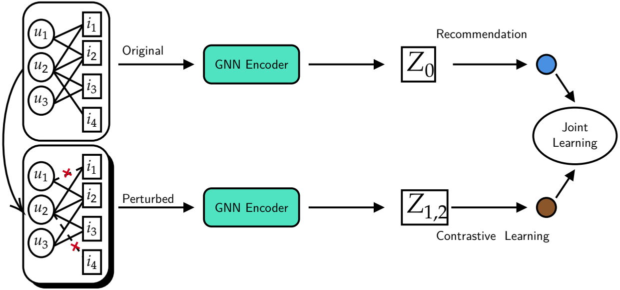
Contrastive Loss

- Augmented views of the same user node are treated as the positive pairs (i.e., $\{(z'_u, z''_u)\}$), and the views of different user nodes are treated as the negative pairs (i.e., $\{(z'_u, z''_v\})$).
- InfoNCE Loss: Maximization principle (**InfoMax**) that aims to *maximize* the correspondence between the representations of the nodes in its different augmented graphs.

$$\mathcal{L}_{\text{NCE}}^{\mathcal{U}} = \sum_{u \in \mathcal{U}} -\log \frac{\exp\left(sim\left(\mathbf{z}'_u, \mathbf{z}''_u\right)/\tau\right)}{\sum_{v \in \mathcal{U}} \exp\left(sim\left(\mathbf{z}'_u, \mathbf{z}''_v\right)/\tau\right)},$$

where τ is the temperature hyper-parameters and sim is the similarity function (e.g., cosine function).
- Analogously, contrastive loss is also adopted on the item side (i.e., $\mathcal{L}_{\text{NCE}}^{\mathcal{I}}$). The final contrastive loss is the combination of two losses as $\mathcal{L}_{\text{NCE}} = \mathcal{L}_{\text{NCE}}^{\mathcal{U}} + \mathcal{L}_{\text{NCE}}^{\mathcal{I}}$.

- Joint training scheme $\mathcal{L} = \mathcal{L}_{\text{Rec}} + \lambda_1 \mathcal{L}_{\text{NCE}} + \lambda_2 \mathcal{L}_{\text{Reg}}$
  - Contrastive learning in recommender systems usually adopts the joint learning strategy to train their model instead of pre-training and fine-tuning strategies.
  - Both pretext tasks and downstream tasks are optimized jointly.
  - SGL (Wu et al., SIGIR2021) demonstrate that joint training will achieve better performance, the pretext tasks and downstream tasks are mutually enhanced with each other.



Outline

Graph Data Augmentation With Edge Operating I

Edge-Dropping Data Augmentations

- Generally edge-dropping is as follows:

$$s_1(G) = \mathbf{A}_1 = \mathbf{A} \odot \mathbf{M}_1, \quad s_2(G) = \mathbf{A}_2 = \mathbf{A} \odot \mathbf{M}_2,$$

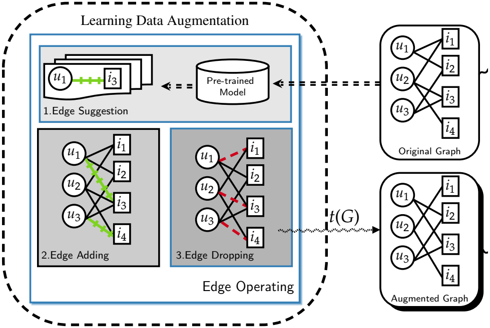
  where $\odot$ is the Hadamard product and $\mathbf{M}_1, \mathbf{M}_2 \in \{0,1\}^{|V|\times|V|}$ are two masking matrices to be applied on the original graph G to generate two augmented graph adjacency matrix $\mathbf{A}_1$ and $\mathbf{A}_2$.

- Sampling edges follow a uniform distribution to keep $(1-\rho) \times |\mathcal{E}|$ edges, where ρ is the edge-dropping ratio. ρ is usually set to a small value (e.g., 0.1).

- Weakness:
  - High complexity of randomly sampling edges from $\mathbf{A}$ is $\mathcal{O}((|V|)^2)$.
  - Introduce noises by randomly adding edges.

Edge-Operating Data Augmentation

- A new data augmentation in recommender systems (i.e., **edge-operating** including both edge-adding and edge-dropping).
  - Edge Suggestion
  - Edge Adding
  - Edge Dropping



Learning Data Augmentation

- Learnable edge operator model t
  - We use a Multi-layer Perception (MLP) to learn the weight for every edge candidate $e_{u,i}$ as follows:

    $$\omega_{u,i} = \mathrm{MLP}\left([z_u \odot z_i] \,\|\, \mathbb{1}_{\mathcal{E}}(e_{u,i})\right),$$

    where $\odot$ is the Hadamard product, z_u and z_i are the embeddings for user u and item i, $\|$ is the concatenation operator and $\mathbb{1}_{\mathcal{E}}(e_{u,i})$ indicates if edge $e_{u,i}$ belongs to original or added edges.
  - Gumbel-Max reparameterization (Jang et al., ICRL2017) to get the probability $p_{u,i}$ for edge $e_{u,i}$ by

    $$p_{u,i} = \mathrm{sigmoid}\left(\frac{(\log\delta - \log(1-\delta) + \omega_{u,i})}{\tau}\right),$$

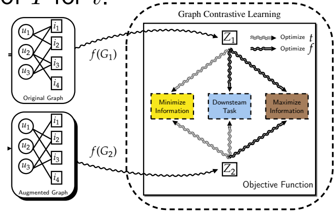
    where $\delta \sim$ Uniform(0,1) and τ is the temperature hyperparameter.
  - We use $p_{u,i}$ to construct augmented graphs $t(G) = \mathbf{A}' = \begin{pmatrix} \mathbf{0} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0} \end{pmatrix}$, where $\mathbf{P} \in R^{|\mathcal{U}|\times|\mathcal{I}|}$ is the probability matrix.

- **InfoMin and InfoMax:**
  - Overall Objective Functions:

$$\min_t \ \lambda_t I(f(G); f(t(G))) + \mathcal{L}(f(t(G)), y)$$
$$\max_f \ I(f(G); f(t(G))) - \mathcal{L}(f(G), y),$$

where $I(X_1; X_2)$ is the mutual information between two random variables X_1 and X_2, t is the data augmentation learner, f is the GNN encoder and $\mathcal{L}$ is the task relevant supervised loss function. λ_t is used to control the influence of I for t.

  - t: MLP
  - f: LightGCN
  - $\mathcal{L}$: BPR
  - I: InfoNCE Estimator



Objective Function II

- ## Mutual Information (MI) Estimator:
  - We use InfoNCE as the MI Estimator

$$I(f(G), f(t(G)) \to -\mathcal{L}_{\mathsf{NCE}} = \frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(sim\left(z_{i,1}, z_{i,2}\right)\right)}{\sum_{i'=1, i'\neq i}^{B} \exp\left(sim\left(z_{i,1}, z_{i',2}\right)\right)},$$

where sim is the cosine similarity to measure the agreement between two representations, z is the node representaton encoded by $f(G)$ and $f(t(G))$, and B is the batch size.

Objective Function III

- Training LDA-GCL
  - Fix t:

$$\mathcal{L}_f = \mathcal{L}_{\mathsf{BPR}}(f(G), y) + \lambda_{ssl}\mathcal{L}_{\mathsf{NCE}}\ (f(G), f(t(G))) + \lambda_{reg}\|f\|_2^2,$$

where λ_{ssl} and λ_{reg} are the hyper-parameters to control the weights of the InfoNCE loss function and the regularization term.

  - Fix f:

$$\mathcal{L}_t = \mathcal{L}_{\mathsf{BPR}}(f(t(G)), y) - \lambda_2\mathcal{L}_{\mathsf{NCE}}\ (f(G), f(t(G))) + \lambda_{reg}\|t\|_2^2,$$

where $\lambda_2 = \lambda_t \times \lambda_{ssl}$ and λ_{reg} are the hyper-parameters to control the weights of the InfoNCE loss function and the regularization term.

Training LDA-GCL

Input: Original bipartite graph $G(\mathcal{U}, \mathcal{I}, \mathcal{E})$; Pre-trained GNN encoder f_0; GNN encoder f; Edge operator model t; Epoch T;
Output: Node representation Z

Generate added edges $\mathcal{E}_1$ from pre-trained model f_0.
Merge added edges $\mathcal{E}_1$ and original edges $\mathcal{E}$ into edge candidates $\mathcal{E}_2$.
Initialize the parameters of edge operator model t and GNN encoder f
for $epoch = 1, ..., T$ **do**
  for each mini-batch interactions $B = \{(u_1, i_1, i_2)\}$ **do**
    Get node set V with user set U and item set I in mini-batch data
    `/* Optimize` t `*/`
    Freeze GNN encoder f; unfreeze edge operator t
    Apply t on $\mathcal{E}_2$ to get augmented graph $t(G)$ and Apply f to get the embeddings Z_1, Z_2 for node V from G
    Compute loss in Equation 15 with Z_1 and Z_2; Back propagation, update t.
    `/* Optimize` f `*/`
    Freeze edge operator t; unfreeze of GNN encoder f
    Apply t on $\mathcal{E}_2$ to get augmented graph $t(G)$ and Apply f to get the embeddings Z_1, Z_2 for node V from G
    Compute loss in Equation 15 with Z_1 and Z_2; Back propagation, update f.
    `/* Judge early stopping condition */`
    if Z_1 match the early stopping condition **then**
      Stop training algorithm; Return the best GNN encoder f_{opt}
    end if
  end for
end for
return $Z = f_{opt}(G)$

Algorithm 1: LDA-GCL Training Algorithm

Outline

- **RQ1**: How does LDA-GCL perform in recommendation tasks as compared with the state-of-the-art CF models and GCL models?
- **RQ2**: If LDA-GCL performs well, what component benefits our LDA-GCL in collaborative filtering tasks?
- **RQ3**: What hyper-parameters affect the effectiveness of the proposed LDA-GCL?

Experimental Settings

Table: Statistics of the datasets used in this paper.

Datasets	#Users	#Items	#Interactions	%Density
Yelp	45,478	30,709	1,777,765	0.127
Gowalla	29,859	40,989	1,027,464	0.084
Amazon-Book	58,145	58,052	2,517,437	0.075
Alibaba-iFashion	300,000	81,614	1,607,813	0.007

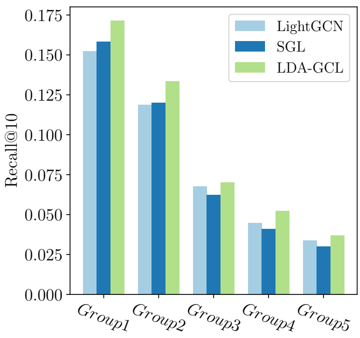
- Datasets: Yelp, Gowalla, Amazon-Book and Alibaba-iFashion.
- Data splits: 80/10/10 - training/validation/testing data split 5 times
- Baselines:
  - Matrix Factorization: **BPRMF**/**NeuMF**/**DMF**
  - Graph Neural Networks: **NGCF**/**DGCF**/ **LightGCN**
  - Graph Contrastive Learning: **SGL**/**SimGCL**/**NCL**
- Metrics: Recall@N and NDCG@N (10, 20, 50)

Performance Comparision

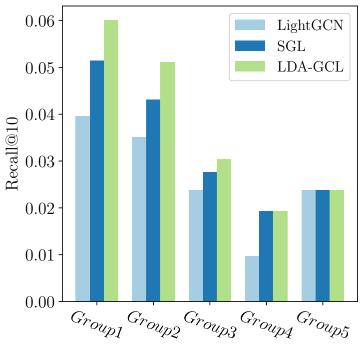
Table: Performance Comparison of Different Baseline Models. The best result is **bolded** and the second result is underlined. * indicates the statistical significance for $p < 0.05$

		Matrix Factorization			Graph Neural Networks			Graph Contrastive Learning			
Dataset	Metric	BPRMF	NeuMF	DMF	NGCF	DGCF	LightGCN	SGL	SimGCL	NCL	LDA-GCL
Yelp	Recall@10	0.0499	0.0367	0.0372	0.0514	0.0606	0.0616	0.0664	0.0743	0.0713	**0.0751***
	Recall@20	0.0829	0.0629	0.0631	0.0857	0.0987	0.1001	0.1072	0.1185	0.1135	**0.1190***
	Recall@50	0.1549	0.1227	0.1215	0.1596	0.1798	0.1817	0.1928	0.2068	0.1997	**0.2101***
	NDCG@10	0.0335	0.0242	0.0248	0.0346	0.0412	0.0419	0.0456	0.0515	0.0489	**0.0518***
	NDCG@20	0.0438	0.0324	0.0327	0.0453	0.0530	0.0538	0.0581	0.0652	0.0619	**0.0653***
	NDCG@50	0.0622	0.0477	0.0476	0.0642	0.0738	0.0748	0.0801	0.0878	0.0841	**0.0886***
Amazon-Book	Recall@10	0.0619	0.0442	0.0313	0.0575	0.0787	0.0783	0.0844	0.0872	0.0947	**0.0975***
	Recall@20	0.0971	0.0726	0.0522	0.0920	0.1191	0.1210	0.1281	0.1251	0.1395	**0.1456***
	Recall@50	0.1676	0.1331	0.0984	0.1624	0.1965	0.2055	0.2117	0.1934	0.2201	**0.2346***
	NDCG@10	0.0431	0.0295	0.0216	0.0400	0.0563	0.0553	0.0606	0.0643	0.0685	**0.0699***
	NDCG@20	0.0537	0.0382	0.0280	0.0505	0.0681	0.0682	0.0739	0.0758	0.0822	**0.0845***
	NDCG@50	0.0721	0.0539	0.0400	0.0688	0.0887	0.0902	0.0956	0.0936	0.1034	**0.1078***
Gowalla	Recall@10	0.1040	0.0882	0.0634	0.0992	0.1343	0.1355	0.1386	0.1487	0.1496	**0.1505**
	Recall@20	0.1525	0.1307	0.0945	0.1462	0.1917	0.1969	0.1969	0.2123	0.2131	**0.2144**
	Recall@50	0.2476	0.2161	0.1559	0.2383	0.2972	0.3093	0.3055	0.3208	0.3228	**0.3284***
	NDCG@10	0.0738	0.0603	0.0450	0.0703	0.0963	0.0961	0.0999	0.1078	0.1081	**0.1085**
	NDCG@20	0.0878	0.0727	0.0540	0.0838	0.1127	0.1136	0.1166	0.1259	0.1263	**0.1268**
	NDCG@50	0.1109	0.0935	0.0692	0.1062	0.1384	0.1411	0.1431	0.1525	0.1534	**0.1547**
Alibaba-iFashion	Recall@10	0.0297	0.0157	0.0138	0.0355	0.0361	0.0402	0.0518	0.0450	0.0490	**0.0605***
	Recall@20	0.0458	0.0264	0.0229	0.0565	0.0549	0.0612	0.0774	0.0651	0.0729	**0.0882***
	Recall@50	0.0784	0.0501	0.0443	0.0994	0.0910	0.1015	0.1258	0.1029	0.1178	**0.1381***
	NDCG@10	0.0158	0.0079	0.0071	0.0185	0.0194	0.0216	0.0280	0.0252	0.0267	**0.0335***
	NDCG@20	0.0199	0.0106	0.0094	0.0237	0.0241	0.0269	0.0344	0.0303	0.0328	**0.0405***
	NDCG@50	0.0264	0.0152	0.0137	0.0323	0.0313	0.0350	0.0440	0.0378	0.0417	**0.0504***

Sparsity Analysis



(a) Gowalla



(b) Alibaba-iFashion

Figure: Performance analysis over different users groups. G_1 is the group of users with the *lowest* interaction number.

Ablation Study

Table: Performance comparison of different variants of LDA-GCL.

Method	Gowalla		Alibaba-iFashion	
	Recall@10	NDCG@10	Recall@10	NDCG@10
LightGCN	0.1342	0.0962	0.0395	0.0212
DA-GCL(0.0,0.0)	0.1488	0.1085	0.0497	0.0274
DA-GCL(0.1,0.0)	0.1492	0.1083	0.0529	0.0289
DA-GCL(0.0,0.1)	0.1487	0.1067	0.0544	0.0299
DA-GCL(0.1,0.1)	0.1479	0.1063	0.0553	0.0303
DA-GCL(0.0,0.5)	0.1412	0.1010	0.0533	0.0290
DA-GCL(0.1,0.5)	0.1409	0.1003	0.0542	0.0296
DA-GCL(0.0,1.0)	0.1369	0.0973	0.0520	0.0282
DA-GCL(0.1,1.0)	0.1359	0.0963	0.0526	0.0285
LDA-GCL (w NGCF)	0.1488	0.1078	0.0589	0.0322
LDA-GCL (w/o EA)	0.1499	0.1087	0.0579	0.0319
LDA-GCL	0.1512	0.1090	0.0599	0.0330

Outline

Conclusion and Future Work

Conclusion

- A theoretically motivated learnable data augmentation model for GCL in recommendation, instead of heuristic designs. (**InfoMin** and **InfoMax**)
- An adversarial framework that can better enhance the effect of GCL in the recommendation.
- Our model achieves state-of-the-art performance on several public benchmark datasets.
- The relevant analytical experiments prove the efficiency of the model design.

Future work

- To make improvements on the efficiency in future work. A potential boosting scheme is the pre-trained edge operator models.

Q&A

Thank you!

感谢您的聆听和反馈

Reference I

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272. PMLR, 2017.

X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pages 639–648, 2020.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *WWW*, pages 689–698, 2018.

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

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua. Neural graph collaborative filtering. In *SIGIR*, pages 165–174, 2019.

X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua. Disentangled graph collaborative filtering. In *SIGIR*, pages 1001–1010, 2020.

J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie. Self-supervised graph learning for recommendation. In *SIGIR*, pages 726–735, 2021.

H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.

T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang, et al. Self-supervised learning for large-scale item recommendations. In *CIKM*, pages 4321–4330, 2021.

J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *WWW*, pages 413–424, 2021.