공학석사 학위논문

Robust Bootstrapping-based Self-supervised Learning for Graph Collaborative Filtering

그래프 협업 필터링을 위한 강력한 부트스트래핑 기반 자기 지도 학습

2023년 02월

서울대학교 대학원

컴퓨터공학부

김 혜 린

Robust Bootstrapping-based Self-supervised Learning for Graph Collaborative Filtering

지도 교수 권 태 경

이 논문을 공학석사 학위논문으로 제출함
2023년 02월

서울대학교 대학원
컴퓨터공학부
김 혜 린

김혜린의 공학석사 학위논문을 인준함
2023년 02월

위 원 장	이 광 근	(인)
부위원장	권 태 경	(인)
위 원	허 충 길	(인)

Abstract

Contrastive learning (CL) based models are gaining traction in recommendation research, since their ability to extract self-supervised signals from raw data matches the requirements of recommender systems to solve the data sparsity issue. Despite their effectiveness, CL-based models have an important limitation: negative sampling. A negative sampling scheme allows positive but unobserved pairs to be selected as negative. To solve this problem, a bootstrapping-based self-supervised learning method that does not require negative sampling has been proposed. However, this method also has limitations. Because only positive samples are used, it is vulnerable to noisy interactions. Also, there is a sparsity issue in real-world data sets.

To tackle the above issues, we introduce a Robust Bootstrapping-based Self-supervised learning model for graph collaborative filtering, named RBS. RBS consists of two modules: a graph denoising module and a self-supervised learning module. The graph denoising module is designed to reduce the influence of noisy interactions. The self-supervised learning module consists of an online encoder and a target encoder. RBS trains its online encoder to predict the target encoder’ s representation, while the target encoder provides consistent targets by slowly approximating the online encoder. In addition, RBS effectively alleviates the data sparsity issue, by adding noises to encoder inputs. A comprehensive empirical study on three benchmark datasets demonstrates that RBS consistently and significantly outperforms all baseline methods.

Keywords : Collaborative Filtering, Bootstrapping-based Self-Supervised Learning, Denoising, Data Augmentation, Graph Neural Networks, Recommendation
Student Number : 2021-23995

Contents

List of Figures

List of Tables

Chapter 1. Introduction

These days, recommender systems have become more important to help users discover items of preference in real−world applications [1] such as E−commerce[3], news portals[4], and social media[5, 6]. The core of the recommender system is to predict whether a user will prefer an item, e.g., click, rate, or purchase, among other forms of interactions [11]. As a fundamental technique, collaborative filtering (CF) models [7, 8], which focus on exploiting past user−item interactions to achieve the prediction, have been widely applied. Earlier work like matrix factorization (MF) [9] projects a single ID of each user (or item) into an embedding vector. Deep neural networks [10] learn better representations.

Recently, Graph neural networks (GNN) have shown good performance in recommender systems as an effective graph collaborative filtering (CF) approach [11]. In particular, Contrastive learning (CL) based models are dominant in recommendation research, since they explicitly aim to distinguish positive user−item interactions from the negative counterparts. Despite their effectiveness, CL−based models have critical limitations. Since the negative interactions are not available in the implicit interaction graph, CL−based models assume that all unobserved interactions are negative. And a negative sampling scheme may choose positive but unobserved pairs as negative. To solve this problem, a bootstrapping−based self−supervised learning method [2, 13] that does not require negative sampling has been proposed. However, this method also has limitations. Since the bootstrapping−based self−supervised learning method uses only positive samples, it is vulnerable to noisy interactions. Especially, noisy interactions can be fatal in the GNN learning process (i.e., the message−passing scheme via aggregating neighborhood information). Therefore, even when the noise of a single node itself is reduced to some extent, the global impact of the aggregated noise from the neighborhood remains uncontrolled. Also, there is a sparsity issue in real−world datasets.

To tackle the aforementioned issues, we introduce a Robust Bootstrapping−based Self−supervised learning model for graph collaborative filtering, named RBS. RBS consists of two modules: a graph denoising module and a self−supervised learning module. The graph denoising module is devised to reduce the influence of noisy interactions. The self−supervised learning module consists of an online encoder and a target encoder. RBS trains its online encoder to predict the target encoder’ s representation, while the target encoder provides consistent targets by slowly approximating the online encoder. In addition, RBS effectively alleviates the data sparsity issue, by adding noises to encoder inputs. A comprehensive empirical study on three benchmark datasets demonstrates the effectiveness of RBS, which consistently and significantly outperforms all baseline methods.

The key contributions of RBS are summarized as follows:

- We devise the RBS method to alleviate the negative impact of noisy interactions in bootstrapping−based self−supervised learning.
- We use random−based noise to mitigate the data sparsity problem.
- Extensive evaluations performed on three real−world datasets demonstrate the superiority of RBS.

Chapter 2. Related Work

In this part, we describe the related work from three perspectives: graph neural networks, graph collaborative filtering, and self-supervised learning.

2.1. Graph Neural Networks

Graph neural networks (GNN) [17, 18] are a family of neural networks that have been widely used as the backbone encoder. A general GNN framework involves two key computations for each node v_i at every layer:

- AGGREGATE: aggregating messages from node v_i 's one-hop neighbors $\mathcal{N}_i$ and updating node representation from its representation in the previous layer and the aggregated messages
- COMBINE: combining the learned embeddings at each layer after K layers graph propagation

2.2. Graph Collaborative Filtering

Collaborative Filtering (CF) is a technique widely used in modern recommender systems. One of the common methods of the CF model is to parameterize users and items as embeddings and learn the embedding parameters by reconstructing historical user-item interactions [11, 19]. For instance, earlier CF models like matrix factorization (MF) [21] project the user's (or item's) ID into an embedding vector. The neural recommender models such as NCF [10] use the same embedding component while enhancing the interaction modeling with neural networks.

Recently emerged graph neural networks (GNN) shine a light on the modeling graph structure, especially high-hop neighbors, to guide the embedding learning [11]. Unlike traditional collaborative filtering methods, graph collaborative filtering models interaction data as a user-item graph and exploits the graph structure for the recommendation [14]. Graph Neural Networks (GNN) now have become widely acknowledged powerful architectures for modeling recommendation data [15]. This paradigm ends the regime of MLP-based recommendation models and boosts the neural recommender systems to a new level. A large number of GNN-based recommendation models claim that they have achieved state-of-the-art performance [11, 15, 22]. In particular, GCN [18] is the most prevalent variant of GNNs. GCN further promotes the development of graph neural recommendation models. For example, GCMC [23], which is the graph-based auto-encoder framework, is proposed for explicit matrix completion. NGCF [24] leverages high-order connectivity in the user-item graph to improve recommendations. Especially, LightGCN [11] is the most popular one, because it removes nonlinear activation and feature transformation of GCN to simplify the architecture recommendation.

2.3. Self-supervised Learning

Recently, self-supervised learning (SSL) approaches have been very successful in computer vision and natural language processing [2]. Studies on self-supervised learning can be roughly categorized into two branches: generative models [25, 31, 32] and contrastive models [12, 33, 34]. Auto-encoding is the most popular generative model that learns how to reconstruct the input data, and can on purpose add noise to improve model robustness [25]. Contrastive models learn how to compare via a Noise Contrastive Estimation (NCE) objective, which can be a global-local contrast [35] or global-global contrast approach [33]. The former focuses on modeling the relationship between the local part of a sample and its global context representation. While the latter directly performs a comparison between different samples, it usually requires multiple views of samples [33, 36].

As Contrastive Learning (CL) works in a self-supervised manner[39], it is essentially a possible solution to the problem of data sparsity [40] in recommender systems. Inspired by the success of CL in other fields, a new wave of research has emerged that integrates CL with recommendations [15]. A common approach [12] to applying CL to recommendation tasks is to first augment the user-item interaction graph with structural perturbations, and then maximize the consistency of representations under different views learned via a graph encoder. SGL [12], a state-of-the-art CL-based recommendation method, augments the original graph by performing node or edge dropout and adopts InfoNCE for CL.

Chapter 3. Methodology

In this chapter, we present the proposed Robust Bootstrapping-based Self-supervised learning model (RBS), which enhances the robustness of bootstrapping-based self-supervised learning for graph collaborative filtering.

3.1. Overview

The proposed RBS model consists of two modules, i.e., the graph denoising module and the self-supervised learning module. Specifically, the graph denoising module is devised for reducing the effect of noisy user-item interactions in representation learning of GNN-based CF [14]. Based on the estimated confidence scores of the user-item interactions, we discard interactions that are confidently estimated as noise in the graph denoising module. Then we propose the self-supervised module, which learns the representations of users and items without any assumptions about negative interactions. The overview of RBS is illustrated in Figure 1, and more details will be described in the following sections.

3.2. Problem Definition

Let $\mathcal{U}$ and $\mathcal{I}$ be the set of users and items respectively. Given a set of observed user-item interactions $\mathcal{R}$, the goal of top-K recommendation is to obtain the preference score $s(u,v) \in \mathbb{R}$, which indicates how much the user u prefers item v. Based on the preference scores, we can recommend K items with the highest scores for each user. To define the preference score by using the denoised representations of users and items, we focus on denoising the graph and training the encoder network that maps each user and item into a low-dimensional latent space where the users' preferences on the items are effectively captured.

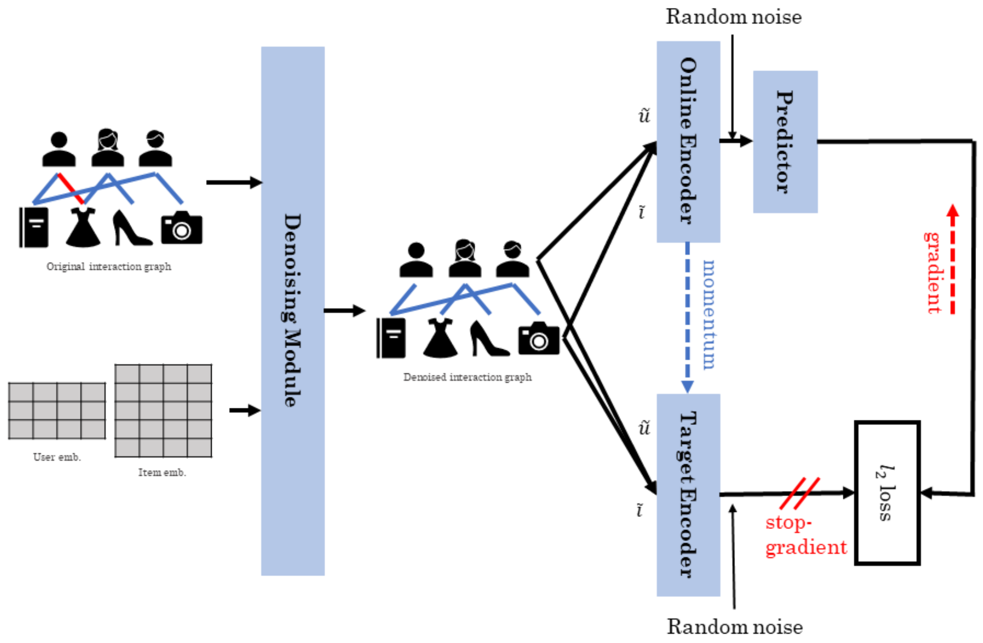

Figure 1: An illustration of RBS model architecture.

3.3. Graph Denoising Module

According to the homophily theory [37] of social networks, nodes with structural roles or similar features are more likely to interact with each other than nodes with different structural roles or features. Since node features might be unavailable in CF, we estimate the confidence scores [14] of observed interactions between users and items based on their structural similarity in the interaction graph. Let $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{I} = \{i_1, \dots, i_N\}$ be the set of M users and N items, respectively. Concretely, given the user−item interaction data $\boldsymbol{R} \in \mathbb{R}^{M \times N}$, we can construct the interaction graph $\mathcal{G}$, where e $r_{u,i} = 1$ entry in R corresponds to an edge between user u and item i in $\mathcal{G}$. Then we extract the nodes' one−hop neighbors as their structural features, that encode the local topological information of nodes. The embedding matrices $\boldsymbol{E}_U \in \mathbb{R}^{M\times d}$ and $\boldsymbol{E}_I \in \mathbb{R}^{N\times d}$ are set to transform the user and item structural features into the same hidden space:

$$\boldsymbol{H}_U = \boldsymbol{R}\boldsymbol{E}_I, \qquad \boldsymbol{H}_I = \boldsymbol{R}^{\mathsf{T}}\boldsymbol{E}_U \tag{1}$$

where $\boldsymbol{H}_U \in \mathbb{R}^{M\times d}$ and $\boldsymbol{H}_I \in \mathbb{R}^{N\times d}$ are the structural feature matrices of users and items, respectively.

We measure their structural similarity in the interaction graph $\mathcal{G}$ to estimate the confidence score of interaction between user u and item i. Given the structural features h_u and h_i, which are the u−th row of $\boldsymbol{H}_U$ and the i−th row of $\boldsymbol{H}_I$, respectively, we use the normalized cosine distance for the confidence score $c_{u,i}$ [14]:

$$c_{u,i} = (\cos(h_u, h_i) + 1)/2 \tag{2}$$

A large confidence score $c_{u,i}$ indicates a reliable interaction between user u and item i based on their structural similarity.

Based on the estimated confidence scores of user−item interactions, we apply a pruning strategy to the interaction graph $\mathcal{G}$ to conduct a denoised interaction graph, denoted as $\tilde{\mathcal{G}}$. To be specific, for the interactions that are considered noise with low confidence scores, the graph denoising module uses denoising strategy with pruning by removing them directly from the interaction graph. For each observed interaction $r_{u,i}$ the corresponding denoised interaction can be formally given as:

$$\tilde{r}_{u,i} = \begin{cases} 0, & c_{u,i} < \beta \\ 1, & c_{u,i} > \beta \end{cases} \tag{3}$$

where β is a hyperparameter representing the predefined threshold.

By pruning the interactions, we can obtain a denoised interaction matrix $\widetilde{\boldsymbol{R}}$, which can further form a denoised interaction graph $\tilde{G}$.To sum up, the graph denoising module mitigates the negative impact of noisy interactions in GNN by pruning them in the interaction graph.

3.4. Self-supervised Learning Module

We introduce a random noise−based data augmentation to alleviate the data sparsity problem in the proposed framework. Inspired by the adversarial examples [38] which are constructed by adding imperceptibly small perturbations to the input images, we directly add random noises to the input of the encoder for an efficient augmentation. Formally, given a node u and its representation e_u in the d−dimensional embedding space, we can implement the following representation−level augmentation:

$$e'_u = e_u + \Delta'_u \tag{4}$$

where the added noise vectors Δ'_u is subject to $||\Delta||_2 = \epsilon$ and $\Delta = \bar{\Delta} \odot sign(e_u)$, $\bar{\Delta} \in \mathbb{R}^d \sim U(0,1)$ [15]. There are two constraints. Δ should be numerically equivalent to a point on the hypersphere with radius ϵ. At the same time, e_u and Δ' should be in the same hyperoctant. Because of this, adding noise will not introduce large deviations of e_u that would make an invalid positive sample [15]. Note that, the random noise added for each node representation is different.

Let f be the encoder network to generate the representations of users and items. We use LightGCN [11] as the encoder. To be specific, each encoder consists of a user encoder and an item encoder, and takes as input the denoised user and item embedding vector obtained from denoised interaction graph $\tilde{G}$.

The self−supervised learning module uses two distinct encoder networks with the same structure: online encoder f_θ and target encoder f_ξ, parameterized by θ and ξ, respectively. The key idea is to use the outputs of the target encoder as a target to train the online encoder while slowly improving the target encoder as well [2, 13]. The online encoder is trained to minimize the error between its output and the target, while the target network is gradually updated based on the momentum update [36] so as to keep its output consistent.

Precisely, for each denoised interaction $(\tilde{u},\tilde{\imath}) \in \tilde{\mathcal{R}}$, the RBS loss is defined based on the mean squared error of the prediction against each other (i.e., representations of $\tilde{u}$ and $\tilde{\imath}$) using the predictor $q_\theta: \mathbb{R}^d \rightarrow \mathbb{R}^d$ on top of the online encoder. It contains two error terms: one is to update the online user vector $f_\theta(\tilde{u})$ to accurately predict the target item vector $f_\xi(\tilde{\imath})$, and the other is to update the online item vector $f_\theta(\tilde{\imath})$ to predict as the target user vector $f_\xi(\tilde{u})$. Finally, the loss is described as follows:

$$\begin{aligned}\mathcal{L}_{\theta,\xi}(\tilde{u},\tilde{\imath}) &= l_2\left[q_\theta\left(f_\theta(\tilde{u})\right), f_\xi(\tilde{\imath})\right] + l_2\left[q_\theta\left(f_\theta(\tilde{\imath})\right), f_\xi(\tilde{u})\right] \\ &\approx -\frac{q_\theta\left(f_\theta(\tilde{u})\right)^{\mathsf{T}} f_\xi(\tilde{\imath})}{\left\|q_\theta\left(f_\theta(\tilde{u})\right)\right\|_2 \left\|f_\xi(\tilde{\imath})\right\|_2} - \frac{q_\theta\left(f_\theta(\tilde{\imath})\right)^{\mathsf{T}} f_\xi(\tilde{u})}{\left\|q_\theta\left(f_\theta(\tilde{\imath})\right)\right\|_2 \left\|f_\xi(\tilde{u})\right\|_2}\end{aligned} \qquad (5)$$

where $l_2[x,y]$ is the l_2 distance between two normalized vectors $\bar{x} = x/\|x\|_2$ and $\bar{y} = y/\|y\|_2$. As the mean squared errors between two normalized vectors are equivalent to the negative value of their inner product (Equation (4)), we simply use the inner product for the optimization.

In a nutshell, the parameters of the online encoder and target encoder are optimized by

$$\begin{aligned}\theta &\leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\theta,\xi} \\ \xi &\leftarrow \tau \cdot \xi + (1-\tau) \cdot \theta\end{aligned} \qquad (6)$$

η is the learning rate for stochastic optimization, and $\tau \in [0,1]$ is a momentum coefficient (also called as target decay) for momentum-based moving average. The online encoder f_θ and the predictor q_θ are effectively optimized by the gradients backpropagated from the loss (Equation (4)), while the target encoder f_ξ is updated with the moving average of the online encoder. By taking a large value of τ, the target encoder slowly approximates the online encoder. This momentum–based update makes ξ evolve more slowly than θ, which enables to bootstrap the representations by providing enhanced but consistent targets to the online encoders [2, 36].

3.5. Prediction

To retrieve K most preferred items for each user, we define the preference score $s(\tilde{u},\tilde{\imath})$ by using the representations of users and items. Since we minimize the prediction error between $\tilde{u}$ and $\tilde{\imath}$ for denoised interaction $(\tilde{u},\tilde{\imath})$, their positive relationship is encoded as the l_2 distance between their representations (Equation (4)). That is, a smaller value of $\mathcal{L}_{\theta,\xi}(\tilde{u},\tilde{\imath})$ indicates that the user–item pair $(\tilde{u},\tilde{\imath})$ is more likely to be interacted, which means the loss becomes inversely proportional to the preference score [13]. To consider the symmetric relationship between $\tilde{u}$ and $\tilde{\imath}$, we define the preference score on the cross–prediction task.

$$s(\tilde{u},\tilde{\imath}) = q_\theta\big(f_\theta(\tilde{u})\big)^\intercal f_\theta(\tilde{\imath}) + f_\theta(\tilde{u})^\intercal q_\theta\big(f_\theta(\tilde{\imath})\big) \tag{7}$$

To obtain the preference score, we use only the representation from the online encoder. The target encoder is discarded. Since the online encoder and the target encoder finally converge to equilibrium by the slow–moving average [2, 13], the preference score can be effectively inferred with only the online encoder. Considering the purpose of the target network to generate targets for training the online network, it makes sense to discard the target network at the end. Additionally, RBS uses the predictor to model interactions. The predictor can encode the relationship between users and items into representations. In conclusion, with the help of the predictor, RBS not only accurately computes the user–item preference scores but also optimizes the representation without explicit use of negative samples.

Chapter 4. Experiments

In this section, we conduct evaluations to justify the superiority of RBS. We first describe comparison results with the state-of-the-art recommendation models for the top-K recommendation. Then, validate the effectiveness of each component through an ablation study.

4.1. Datasets

In our experiments, we adopt three real-world datasets: CiteULike, Movielens-1M, and LastFM.

- CiteULike [26]: allows users to create their own collections of articles.
- Movielens-1M [29]: has been widely used in previous studies on recommendation, which contains movie ratings.
- LastFM [28]: contains users and songs.

The statistics of all three datasets are summarized in Table 1.

Table 1: Statistics of the datasets.

Dataset	CiteULike	Movielens-1M	LastFM
#Users	5,551	6,040	1,892
#Items	16,980	3,629	17,632
#Interactions	106,852	423,208	48,288
Density	0.113%	1.931%	0.145%

4.2. Baselines

We compare the proposed RBS with the following CF models:

- **NGCF** [24]: A neighbor-based method to encode the neighbors of users (and items) using Graph Convolutional Networks (GCN). Multi-hop neighbors can also be considered based on the GCN layer stack.
- **LightGCN (LGCN)** [11]: This method devised a light graph convolution for training efficiency and generation ability.
- **SGL** [12]: This model utilizes self-supervised learning to improve the accuracy and robustness of LightGCN for the recommendation. In our experiments, we adopt the SGL equipped with the Edge Dropout.
- **BUIR** [13]: BUIR has a two-branch architecture which consists of a target network and an online network. And it only uses positive examples for self-supervised learning.

For a fair comparison, we refer to the best hyperparameter settings reported in the original papers of the baselines and then fine-tune the hyperparameters of the baselines.

4.3. Evaluation Metrics

As we focus on the top-K recommendation task for implicit feedback, we evaluate the performance of each model by using three widely-used ranking metrics.

- Precision (P@K): Precision measures how many test items are included in the list of top-K items.
- Recall (R@K): Recall means how many test items are included in the total items that users are interested in.
- NDCG (N@K): Normalized Discounted Cumulative Gain.

NDCG assigns higher scores on the upper−ranked test items.

4.4. Implementation Details

We implement the proposed framework by using PyTorch. Adam [30] is applied to optimize all the parameters with the learning rate η initialized as 0.001. We adopt a single linear layer for the predictor q_θ and fix the momentum coefficient τ to 0.995. We set batch size 1024 and embedding size $d = 250$, and 1−layer LightGCN as the encoder. Besides, we tune the pruning threshold β in $\{0.38, 0.39, 0.4\}$. For the baselines, each experiment is performed with codes provided by the authors and reported hyperparameters. We set up the maximum number of epochs to 500 and adopt the early stopping strategy.

4.5. Overall Performance

First and foremost, we compared the top−K recommendation performance of RBS and baselines. Table 2, Table 3, and Table 4 present the comparison results on three real−world datasets, respectively.

Table 2: The overall performance comparison on the CiteULike dataset. The best result is bolded and the runner-up is underlined.

CiteULike						
K	Metric	NGCF	LGCN	SGL	BUIR	RBS
10	P	0.0295	0.0654	0.1613	<u>0.1950</u>	**0.1982**
	R	0.1125	0.1395	0.1573	<u>0.1620</u>	**0.1638**
	N	0.1028	0.1430	0.1582	<u>0.1843</u>	**0.1887**
20	P	0.0386	0.0663	0.1717	<u>0.2451</u>	**0.2507**
	R	0.1288	0.1877	0.2158	<u>0.2326</u>	**0.2376**
	N	0.1253	0.1494	0.1737	<u>0.2011</u>	**0.2064**
50	P	0.0479	0.0686	0.2654	<u>0.3543</u>	**0.3596**
	R	0.1315	0.1865	0.3225	<u>0.3536</u>	**0.3588**
	N	0.1417	0.1464	0.2015	<u>0.2376</u>	**0.2428**

Table 3: The overall performance comparison on the Movielens-1M dataset. The best result is bolded and the runner-up is underlined.

Movielens-1M						
K	Metric	NGCF	LGCN	SGL	BUIR	RBS
10	P	0.1289	0.2051	0.2889	<u>0.3310</u>	**0.3329**
	R	0.0747	0.1099	0.1182	<u>0.1295</u>	**0.1335**
	N	0.1651	0.2592	0.2933	<u>0.3519</u>	**0.3543**
20	P	0.1405	0.2052	0.2776	<u>0.3202</u>	**0.3263**
	R	0.1372	0.1895	0.1937	<u>0.2025</u>	**0.2081**
	N	0.1880	0.2587	0.2954	<u>0.3310</u>	**0.3359**
50	P	0.1557	0. 1597	0.2936	<u>0.3749</u>	**0.3877**
	R	0.1375	0.1857	<u>0.3384</u>	0.3366	**0.3491**
	N	0.1877	0.2563	0.2993	<u>0.3388</u>	**0.3476**

Table 4: The overall performance comparison on the LastFM dataset. The best result is bolded and the runner-up is underlined.

LastFM						
K	Metric	NGCF	LGCN	SGL	BUIR	RBS
10	P	0.0676	0.1271	0.1708	<u>0.1866</u>	**0.1909**
	R	0.1259	0.1420	0.1620	<u>0.1693</u>	**0.1722**
	N	0.1520	0.1963	0.2043	<u>0.2235</u>	**0.2263**
20	P	0.0834	0.1260	0.1822	<u>0.2482</u>	**0.2521**
	R	0.1601	0.2397	<u>0.2501</u>	0.2472	**0.2521**
	N	0.1735	0.2234	0.2320	<u>0.2468</u>	**0.2521**
50	P	0.1170	0.1268	0.2825	<u>0.3842</u>	**0.3886**
	R	0.2204	0.2409	0.3624	<u>0.3842</u>	**0.3906**
	N	0.2032	0.2231	0.2943	<u>0.3087</u>	**0.3119**

The overall performance comparison results between RBS and baseline models are shown in Table 2, Table 3, and Table 4. The best result is bolded and the runner-up is underlined. As can be observed, RBS outperforms all baseline models on all datasets. The performance of the self-supervised learning-based models (SGL, BUIR, and RBS) is significantly higher than the basic GNN-based models (NGCF and LGCN). This indicates that self-supervised learning improves the performance of GNN-based methods. Also, compared to the discriminative models (NGCF, LGCN, and SGL) the performance of RBS is better. This means that the imperfect assumption that regards the unobserved interaction as a negative interaction limits the ability to capture user preferences. Compared to BUIR, which is a bootstrapping-based model, the performance of RBS is higher. Through this, it can be seen that denoising plays an important role in capturing the user's exact preference.

4.6. Ablation Study

To validate the contribution of each module in RBS, we conduct an ablation study on the CiteULike dataset. We adopt Precision, Recall, and NDCG as the metrics. We present the experimental results in Figure 2, Figure 3, and Figure 4. The following figures are graphs showing the relative performance when RBS performance is set to 100%. ‘w/o D’ is a model removed the graph denoising module, and ‘w/o N’ is a model without random−based noise. ‘w/o both’ is a model removed both. From Figure 2, Figure3, and Figure4, we can observe that the performance of RBS degrades significantly when the graph denoising module or random−based noise is removed. Overall, the ablation study indicates that both the graph denoising module and random−based noise contribute to RBS.

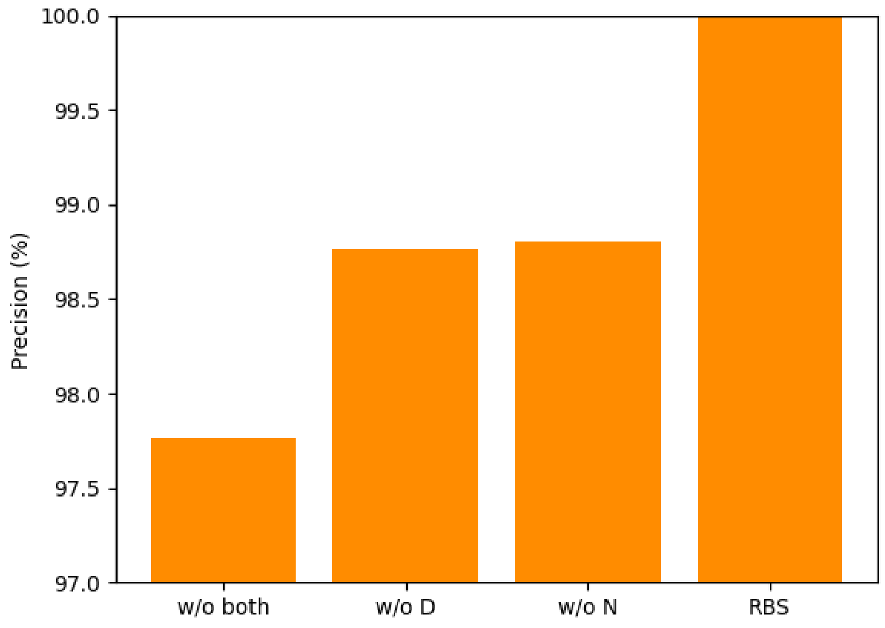

Figure 2: Ablation study on CiteULike dataset – Precision.

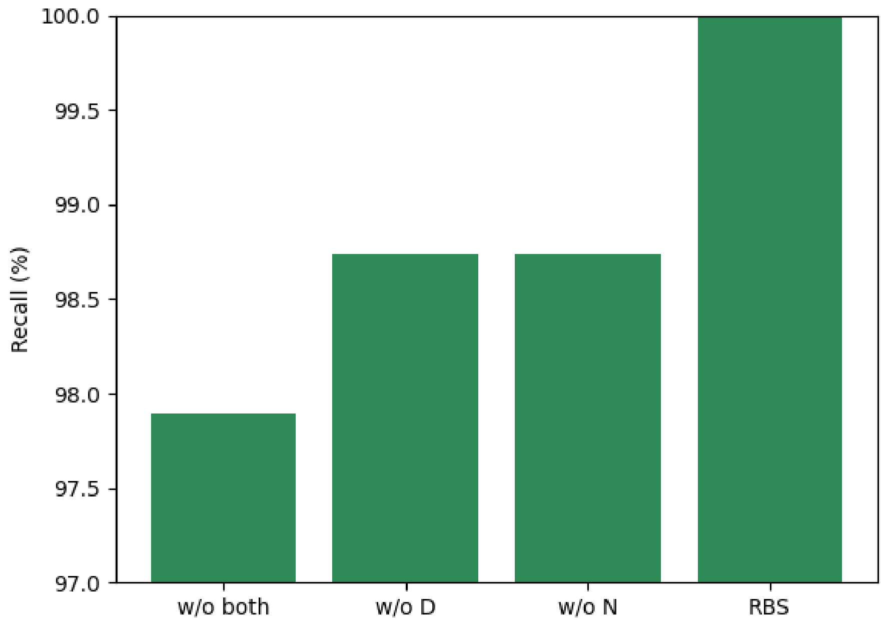

Figure 3: Ablation study on CiteULike dataset – Recall.

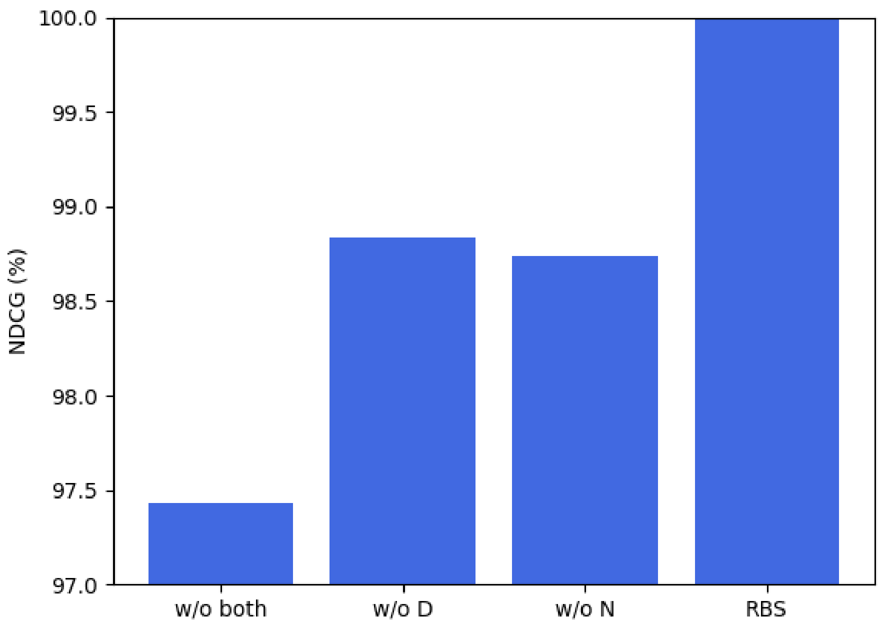

Figure 4: Ablation study on CiteULike dataset – NDCG.

Chapter 5. Conclusion

In this paper, we propose a Robust Bootstrapping−based Self−supervised learning model for graph collaborative filtering, named RBS. To reduce the negative impact of noisy interactions, the graph denoising module applies denoising by pruning based on the confidence score. RBS also significantly improves recommendation performance by adding random−based noise to the encoder input to alleviate the data sparsity problem. The extensive experiments demonstrate that RBS outperforms all other baselines in terms of top−K recommendation.

Bibliography

[1] Ansari, A., Essegaier, S., & Kohli, R. (2000). Internet Recommendation Systems. Journal of Marketing Research, 37(3), 363–375.
[2] Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., ... & Valko, M. (2020). Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems, 33, 21271-21284.
[3] Nie, L., Wang, W., Hong, R., Wang, M., & Tian, Q. (2019, October). Multimodal dialog system: Generating responses via adaptive decoders. In Proceedings of the 27th ACM International Conference on Multimedia (pp. 1098-1106).
[4] Lu, H., Zhang, M., & Ma, S. (2018, June). Between clicks and satisfaction: Study on multi-phase user preferences and satisfaction for online news reading. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 435-444).
[5] Ren, Z., Liang, S., Li, P., Wang, S., & de Rijke, M. (2017, February). Social collaborative viewpoint regression with explainable recommendations. In Proceedings of the tenth ACM international conference on web search and data mining (pp. 485-494).
[6] Wang, W., Huang, M., Xu, X. S., Shen, F., & Nie, L. (2018, June). Chat more: Deepening and widening the chatting topic via a deep model. In The 41st international acm sigir conference on research & development in information retrieval (pp. 255-264).
[7] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In The adaptive web (pp. 291-324). Springer, Berlin, Heidelberg.
[8] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009.
[9] Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback.

arXiv preprint arXiv:1205.2618.
[10] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).
[11] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020, July). 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 (pp. 639-648).
[12] Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021, July). Self-supervised graph learning for recommendation. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval (pp. 726-735).
[13] Lee, D., Kang, S., Ju, H., Park, C., & Yu, H. (2021, July). Bootstrapping user and item representations for one-class collaborative filtering. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 317-326).
[14] Tian, C., Xie, Y., Li, Y., Yang, N., & Zhao, W. X. (2022, July). Learning to Denoise Unreliable Interactions for Graph Collaborative Filtering. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 122-132).
[15] Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022, July). Are graph augmentations necessary? simple graph contrastive learning for recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1294-1303).
[16] Wang, W., Feng, F., He, X., Nie, L., & Chua, T. S. (2021, March). Denoising implicit feedback for recommendation. In Proceedings of the 14th ACM international conference on web search and data mining (pp. 373-381).
[17] Wu, L., Lin, H., Tan, C., Gao, Z., & Li, S. Z. (2021). Self-supervised learning on graphs: Contrastive, generative, or predictive. IEEE Transactions on Knowledge and Data Engineering.

[18] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In 5th International Conference on Learning Representations.
[19] Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems (pp. 191-198).
[20] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018, July). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 974-983).
[21] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
[22] Yu, J., Yin, H., Li, J., Wang, Q., Hung, N. Q. V., & Zhang, X. (2021, April). Self-supervised multi-channel hypergraph convolutional network for social recommendation. In Proceedings of the Web Conference 2021 (pp. 413-424).
[23] Berg, R. V. D., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263.
[24] Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019, July). Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (pp. 165-174).
[25] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
[26] Wang, H., Chen, B., & Li, W. J. (2013, June). Collaborative topic regression with social regularization for tag recommendation. In Twenty-Third International Joint Conference on Artificial Intelligence.
[27] Li, M., Zhang, S., Zhu, F., Qian, W., Zang, L., Han, J., & Hu, S. (2020, April). Symmetric metric learning with adaptive margin for recommendation. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 04, pp. 4634-4641).
[28] Brusilovsky, P., Cantador, I., Koren, Y., Kuflik, T., & Weimer, M.

(2010, September). Workshop on information heterogeneity and fusion in recommender systems (HetRec 2010). In Proceedings of the fourth ACM conference on Recommender systems (pp. 375-376).
[29] Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4), 1-19.
[30] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[31] Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018, April). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).
[32] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., & Graves, A. (2016). Conditional image generation with pixelcnn decoders. Advances in neural information processing systems, 29.
[33] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.
[34] Komodakis, N., & Gidaris, S. (2018, April). Unsupervised representation learning by predicting image rotations. In International Conference on Learning Representations (ICLR).
[35] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670.
[36] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9729-9738).
[37] McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. Annual review of sociology, 415-444.
[38] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint

arXiv:1412.6572.
[39] Yu, J., Yin, H., Xia, X., Chen, T., Li, J., & Huang, Z. (2022). Self-Supervised Learning for Recommender Systems: A Survey. arXiv preprint arXiv:2203.15876.
[40] Yu, J., Gao, M., Li, J., Yin, H., & Liu, H. (2018, October). Adaptive implicit friends identification over heterogeneous network for social recommendation. In Proceedings of the 27th ACM international conference on information and knowledge management (pp. 357-366).
[41] Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748

초록

대조 학습 기반 모델은 원시 데이터에서 자체 감독 신호를 추출하는 기능이 데이터 희소성 문제를 해결하기 위한 추천 시스템의 요구 사항과 일치하기 때문에 추천 연구에서 주목을 받고 있다. 이러한 효율성에도 불구하고 대조 학습 기반 모델에는 중요한 한계점이 있다. 바로 네거티브 샘플링이다. 네거티브 샘플링 방식을 사용하면 사용자의 취향에 맞는 항목이지만 상호작용이 관찰되지 않은 사용자–항목 쌍을 네거티브로 선택할 수 있다. 이를 해결하기 위해 네거티브 샘플링이 필요하지 않은 부트스트래핑 기반의 자기 지도 학습 방법이 제안되었다. 그러나 이 방법에도 한계점이 있다. 관찰된 샘플만 사용하기 때문에 노이즈가 있는 상호 작용에 취약하다. 또한 실제 데이터 셋에는 희소성 문제가 있다.

위의 문제를 해결하기 위해 그래프 협업 필터링을 위한 강력한 부트스트래핑 기반 자기 지도 학습 모델, RBS를 소개한다. RBS는 그래프 노이즈 제거 모듈과 자가 지도 학습 모듈의 두 가지 모듈로 구성된다. 그래프 노이즈 제거 모듈은 잡음이 있는 상호 작용의 영향을 줄이기 위해 설계되었다. 자기 지도 학습 모듈은 온라인 인코더와 타깃 인코더로 구성된다. RBS는 타깃 인코더의 표현을 예측하도록 온라인 인코더를 학습하는 반면, 타깃 인코더는 온라인 인코더를 천천히 근사하여 일관된 타깃을 제공한다. 또한 RBS는 인코더 입력에 노이즈를 추가하여 데이터 희소성 문제를 효과적으로 완화한다. 3가지 벤치마크 데이터 셋에 대한 포괄적인 경험적 연구는 RBS가 모든 기준 모델을 일관되고 크게 능가한다는 것을 보여준다.