Pre-training for Recommendation Unlearning

Guoxuan Chen
The University of Hong Kong
Hong Kong
guoxchen@foxmail.com

Lianghao Xia
The University of Hong Kong
Hong Kong
aka_xia@foxmail.com

Chao Huang*
The University of Hong Kong
Hong Kong
chaohuang75@gmail.com

Abstract

Modern recommender systems powered by Graph Neural Networks (GNNs) excel at modeling complex user-item interactions, yet increasingly face scenarios requiring selective forgetting of training data. Beyond user requests to remove specific interactions due to privacy concerns or preference changes, regulatory frameworks mandate recommender systems' ability to eliminate the influence of certain user data from models. This recommendation unlearning challenge presents unique difficulties as removing connections within interaction graphs creates ripple effects throughout the model, potentially impacting recommendations for numerous users. Traditional approaches suffer from significant drawbacks: fragmentation methods damage graph structure and diminish performance, while influence function techniques make assumptions that may not hold in complex GNNs, particularly with self-supervised or random architectures. To address these limitations, we propose a novel model-agnostic pre-training paradigm UnlearnRec that prepares systems for efficient unlearning operations. Our Influence Encoder takes unlearning requests together with existing model parameters and directly produces updated parameters of unlearned model with little fine-tuning, avoiding complete retraining while preserving model performance characteristics. Extensive evaluation on public benchmarks demonstrates that our method delivers exceptional unlearning effectiveness while providing more than 10x speedup compared to retraining approaches. We release our method implementation at: https://github.com/HKUDS/UnlearnRec.

CCS Concepts

• **Information systems** → **Recommender systems**; *Collaborative filtering*; • **Security and privacy** → *Privacy protections.*

Keywords

Machine Unlearning, Recommender Systems, Pre-training

ACM Reference Format:
Guoxuan Chen, Lianghao Xia, and Chao Huang. 2025. Pre-training for Recommendation Unlearning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730060

*Chao Huang is the Corresponding Author.

1 Introduction

Recommender systems (RS) [11, 19, 37] have emerged as a critical infrastructure in modern digital landscapes, driving user engagement and satisfaction across various platforms including e-commerce, streaming services, and social media. These systems utilize vast amounts of user data to generate personalized suggestions, leveraging learning algorithms to predict user preferences and behaviors. However, growing concerns about data privacy and legal recognition of "the right to be forgotten" [16] through regulations like General Data Protection Regulation [23], the Personal Information Protection and Electronic Documents Act [1] have fundamentally challenged these systems. Specifically, there is a growing need for recommendation unlearning, which involves removing specific user interactions and preferences from trained recommender models in compliance with user requests or regulatory requirements.

Graph Neural Networks (GNNs) have advanced Collaborative Filtering, establishing themselves as the leading paradigm for modern recommendation systems [5, 33, 34, 39]. These models excel at capturing complex user-item interaction graphs, yet implementing recommendation unlearning presents significant challenges due to their interconnected architecture. When users delete interactions, the process extends beyond removing data entries; the system must recalibrate recommendation probabilities for related items while ensuring deleted information becomes completely unrecoverable from both storage and inference capabilities. Crucially, this unlearning process must preserve overall prediction accuracy while making these adjustments. The challenge intensifies in GNN-based recommender systems where embeddings form an interdependent graph, collectively refined through training iterations, creating an information web that requires careful disentanglement.

Addressing recommendation unlearning challenges demands algorithms that can update models efficiently without complete retraining – a requirement that would be computationally prohibitive in real-world applications. Current approaches fall into two primary categories: i) Partition-based Methods (P-based methods)[2, 4, 7, 10] and ii) Influence Function-based Methods (IF-based methods)[30–32, 38]. P-based techniques strategically divide the interaction graph into multiple shards; when unlearning requests arrive, only affected shards undergo retraining before being reintegrated for prediction. However, this partitioning inevitably disrupts the graph's natural topology, degrading recommendation precision. Additionally, requests spanning numerous shards significantly increase processing time, limiting practical application. IF-based methods, while computationally faster, employ mathematical approximations to estimate unlearning impacts. These approaches face three critical limitations: their underlying mathematical assumptions often fail in complex real-world scenarios; they struggle with Self-Supervised Learning (SSL)-based GNNs where random network structures and supervision signals generated at runtime undermine gradient estimation accuracy; and they typically impose substantially higher memory requirements. These limitations highlight the urgent need for more robust and efficient unlearning solutions that maintain recommendation quality while respecting user privacy.

To address these challenges, we introduce a novel unlearning paradigm called UnlearnRec, a learnable, model-agnostic framework built around a pre-trained Influence Encoder (IE). In our UnlearnRec, IE can be pre-trained in advance and then either deployed directly or quickly fine-tuned when unlearning requests emerge. The core of our solution is the Influence Encoder, a learnable module specifically designed to predict how unlearning requests will impact GNN embeddings. We develop this encoder through a two-stage process of comprehensive pre-training followed by targeted fine-tuning when needed. The key contributions of our research are as follows:

- We introduce the first pretraining-based learnable paradigm for recommendation unlearning that efficiently processes requests and shifts embedding distributions while preserving accuracy.
- Our UnlearnRec framework is model-agnostic, enabling effective unlearning in state-of-the-art recommender systems.
- Comprehensive experiments demonstrate UnlearnRec's superior performance across different dimensions.

2 GNN-based Recommendation Unlearning

2.1 Interaction Graph for Recommendation

GNN-based methods have been shown to be the most effective solutions for CF [6, 34]. In the CF paradigm, a user set $\mathcal{U}$ ($|\mathcal{U}| = I$), an item set $\mathcal{V}$ ($|\mathcal{V}| = J$), and a user-item interaction matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ are utilized to represent the historical interaction records, based on which the encoders derive embeddings and make predictions. For each entry $a_{i,j}$ in the interaction matrix $\mathbf{A}$, $a_{i,j} = 1$ if user $u_i \in \mathcal{U}$ has interacted with item $v_j \in \mathcal{V}$, otherwise $a_{i,j} = 0$. The interaction edge set $\mathcal{E}$ corresponds one-to-one with the adjacency matrix $\mathbf{A}$, i.e., if there is $a_{i,j} = 1$ in $\mathbf{A}$, there exists an undirected edge $(u_i, v_j) \in \mathcal{E}$. Nowadays, GNN has demonstrated its superiority in modelling this sort of interaction graph denoted by $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, in which $\mathcal{U}$, $\mathcal{V}$ are two kinds of sets of graph nodes and $\mathcal{E}$ is the edge set.

2.2 SSL-based GNN Recommender

Based on interaction graph data, GNN first initializes an embedding vector for each node (user/item in $\mathcal{U}, \mathcal{V}$). It then performs multiple rounds of forward propagation and fusion along the edges of the graph, resulting in the final representation of last layer for each node. The final embeddings incorporate not only the information of the nodes themselves but also that of their multi-order neighbors, thereby encapsulating the structural information of the graph.

$$\mathbf{e}_{i,l} = \sum_{(v_j,u_i)\in\mathcal{E}} \frac{1}{\sqrt{\delta_i\delta_j}}\mathbf{e}_{j,l-1}, \quad \mathbf{e}_{j,l} = \sum_{(u_i,v_j)\in\mathcal{E}} \frac{1}{\sqrt{\delta_i\delta_j}}\mathbf{e}_{i,l-1} \tag{1}$$

$$\mathbf{E}_l = \mathbf{D}^{-\frac{1}{2}} \cdot \bar{\mathbf{A}} \cdot \mathbf{D}^{-\frac{1}{2}} \cdot \mathbf{E}_{l-1}, \qquad \text{user } u_i \in \mathcal{U}, \ \text{item } v_j \in \mathcal{V} \tag{2}$$

Eq. 1 shows an example of GNN, which is the most widely applied backbone architecture [13] for most GNN models [5, 20, 29, 35]. $\mathbf{e}_{i,l}, \mathbf{e}_{i,l-1} \in \mathbb{R}^d$ denote the l-layer and $(l-1)$-layer embedding vectors for user u_i, and analogously for item v_j. Therefore, $\mathbf{e}_{i,0}, \mathbf{e}_{j,0}$ represent the initial embeddings for u_i and v_j. And δ_i, δ_j denote the degrees of nodes u_i, v_j, for Lapalacian normalization. Eq. 2 gives the corresponding matrix form where $\bar{\mathbf{A}} \in \mathbb{R}^{(I+J)\times(I+J)}$ is the symmetric adjacent matrix for graph $\mathcal{G}$, derived from the interaction matrix $\mathbf{A}$ [28]. $\mathbf{D}$ is the diagonal degree matrix of $\bar{\mathbf{A}}$, and $\mathbf{E}_l \in \mathbb{R}^{(I+J)\times d}$ denotes the nodes' embedding matrix of layer l, where each row in $\mathbf{E}_l$ is an embedding for a node, i.e., $\mathbf{e}_{i,l}$ or $\mathbf{e}_{j,l}$. After L-layers' iteration, GNN aggregates the multi-order embeddings to output the final embeddings $\bar{\mathbf{e}}_i, \ \bar{\mathbf{e}}_j \in \mathbb{R}^d$, corresponding to row vectors in $\bar{\mathbf{E}}$, and the user-item relation prediction score $\hat{y}_{i,j}$.

$$\hat{y}_{i,j} = \bar{\mathbf{e}}_i^\top \bar{\mathbf{e}}_j, \qquad \bar{\mathbf{e}}_i, \ \bar{\mathbf{e}}_j \in \bar{\mathbf{E}}, \qquad \bar{\mathbf{E}} = \sum_{l=0}^{L-1} \mathbf{E}_l \tag{3}$$

BPR loss [24] is widely employed over observed interaction pairs $(u_i, v_{j^+}) \in \mathcal{E}$, and sampled negative pairs (u_i, v_{j^-}) to optimize GNN.

$$\mathcal{L}_{bpr} = \sum_{(u_i,v_{j^+},v_{j^-})} -\log \mathrm{sigm}(\hat{y}_{i,j^+} - \hat{y}_{i,j^-}) \tag{4}$$

where $sigm(\cdot)$ denotes the Sigmoid function. By minimizing Eq. 4, GNN recommenders can produce excellent performance but nowadays, SSL-based enhancement modules are widely used in GNNs to further achieve state-of-the-art performance.

$$\mathbf{E}_{\text{v1}}{=}\mathbf{View}_1(\{\mathbf{E}_l | l \in [0, L-1]\}), \ \mathbf{E}_{\text{v2}}{=}\mathbf{View}_2(\{\mathbf{E}_l | l \in [0, L-1]\}) \tag{5}$$

$$\mathcal{L}_{contrast} = \mathbf{Cross\text{-}view\text{-}similarity}(\mathbf{E}_{\text{v1}}, \ \mathbf{E}_{\text{v2}}) \tag{6}$$

Eq. 5,6 give an example of SSL loss function where the model creates two views of the embeddings (Eq. 5) and then optimizes the contrast loss (Eq. 6). Besides, there are many other kinds of designs for SSL [20, 29, 35]. Of utmost importance is that, SSL-based GNNs usually generate random or data-dependent network structures and supervision signals in $\mathbf{View}_{1|2}$ functions at runtime, making it hard to precisely estimate the gradients for the trainable embeddings by manual computation without actual training (see 2.3). To recap, we use $\mathcal{M}odel(\mathbf{E}_0, \bar{\mathbf{A}})$ to represent and sum up GNN RS of all kinds.

2.3 SSL-based Recommendation Unlearning

In real-life scenarios, there is always a demand from users to undo their interaction records in order to delete accidental inputs or mistakes, change preferences, or just protect their privacy. This corresponds to deleting edges in the graph $\mathcal{G}$. We use $\mathcal{E}_\Delta$ to denote the set of edges to be undone and $\mathcal{E}_r$ to denote remaining edges,i.e., $\mathcal{E}_\Delta = \mathcal{E} \setminus \mathcal{E}_r$. In practice, pairs in $\mathcal{E}$ not only form the adjacency matrix for GNN's forward propagation but also serve as the positive supervisory signals in the loss function(Eq. 4). Therefore, dropping a subset of edges not only changes the graph structure, but also turns the original positive labels into negative ones, definitely resulting in the distribution shift of embeddings. To avoid retraining, unlearning methods are proposed to fulfill the requirements that:

- It must shift embeddings so that unlearned edges are predicted as negatives while maintaining a distribution similar to retraining.
- Unlearned models can still achieve good prediction performance.
- Unlearning process must be more efficient than retraining.

In recommender systems, we mainly focus on edge unlearning and when some nodes need to be unlearned, we just need to unlearn all the edges associated with them, and directly delete the nodes.

For SSL-based GNNs, it is difficult to accurately estimate the difference between the old well-trained embeddings before unlearning and the newly obtained embeddings after unlearning through retraining, just by a manually-designed end-to-end function, i.e. IF like [30, 31], because the complicated and randomized SSL network structures and signals will produce unforeseeable embedding distributions at runtime through iterative training. So, the distribution shift estimated by IF methods would be inaccurate.

3 Methodology

Our goal is to build a model-agnostic pre-training paradigm, which produces a well-pretrained encoder $\mathbf{IE}(\cdot)$ that takes the unlearning requests and the original embeddings, and outputs the unlearned embeddings directly (or for further fine-tuning, see Fig. 1).

3.1 Influence Dependency Matrix

Here, we empirically derive a matrix as a means to construct the **IE**, and formalize the unlearning requests. From the BPR loss (Eq. 4), we can derive the approximate gradients over a positive interaction pair (u_i, v_{j^+}) and a negative pair (u_i, v_{j^-}):

$$\nabla_{\bar{\mathbf{e}}_i} = \frac{\partial \mathcal{L}_{bpr}}{\partial \bar{\mathbf{e}}_i} = \left(\text{sigm}(\hat{y}_{i,j^+} - \hat{y}_{i,j^-}) - 1\right) \cdot (\bar{\mathbf{e}}_{j^+} - \bar{\mathbf{e}}_{j^-}) \tag{7}$$

$$= c \cdot (\bar{\mathbf{e}}_{j^+} - \bar{\mathbf{e}}_{j^-}), \quad \text{sigm}(\cdot) < 1, \quad c < 0 \tag{8}$$

These gradients will be backpropagated to every node along the graph structure during training. However, once the interaction (u_i, v_{j^+}) then needs to be unlearned, $\bar{\mathbf{e}}_{j^+}$ should be turned into the negative side. A compensatory gradient $\nabla^{(c)}_{\bar{\mathbf{e}}_i}$ should be considered:

$$\nabla'_{\bar{\mathbf{e}}_i} = \nabla_{\bar{\mathbf{e}}_i} - \nabla^{(c)}_{\bar{\mathbf{e}}_i}, \quad \nabla^{(c)}_{\bar{\mathbf{e}}_i} = c' \bar{\mathbf{e}}_{j^+}, \quad c' < c, \ |c'| > c \tag{9}$$

Similarly, when a number of interaction pairs $\{(u_i, v_{j_1^+}), ..., (u_i, v_{j_n^+})\}$ need to be unlearned, the compensated gradient could roughly be:

$$\nabla'_{\bar{\mathbf{e}}_i} = \nabla_{\bar{\mathbf{e}}_i} - \nabla^{(c)}_{\bar{\mathbf{e}}_i} = \nabla_{\bar{\mathbf{e}}_i} - \left(c'_{i,j_1^+}\bar{\mathbf{e}}_{j_1^+} + c'_{i,j_2^+}\bar{\mathbf{e}}_{j_2^+} +, \cdots, c'_{i,j_n^+}\bar{\mathbf{e}}_{j_n^+}\right) \tag{10}$$

The above is merely a one-order estimation. But inspired by Eq. 8, we can find that, since $c < 0$, the training process $\bar{\mathbf{e}}_i^{(t+1)} = \bar{\mathbf{e}}_i^{(t)} - \eta c \cdot (\bar{\mathbf{e}}_{j^+}^{(t)} - \bar{\mathbf{e}}_{j^-}^{(t)})$ (learning rate $\eta > 0$) is actually a process of increasing embedding similarity between observed pairs (u_i, v_{j^+}), and decreasing that between negative pairs (u_i, v_{j^-}), i.e., bringing $\bar{\mathbf{e}}_i$ closer to $\bar{\mathbf{e}}_{j^+}$ and away from $\bar{\mathbf{e}}_{j^-}$, which aligns with the design philosophy of GNNs. Moreover, Eq. 9 shows that unlearning (u_i, v_{j^+}) means pushing the already-pulled-closer $\bar{\mathbf{e}}_{j^+}$ further away.

$$\bar{\mathbf{e}}_i^{(u)} \approx \bar{\mathbf{e}}_i^{(t)} - \eta \nabla'_{\bar{\mathbf{e}}_i^{(t)}} = \bar{\mathbf{e}}_i^{(t)} - \eta c \cdot (\bar{\mathbf{e}}_{j^+}^{(t)} - \bar{\mathbf{e}}_{j^-}^{(t)}) + \eta c' \bar{\mathbf{e}}_{j^+}^{(t)} \tag{11}$$

$$= \bar{\mathbf{e}}_i^{(t)} - \eta(c - c')\bar{\mathbf{e}}_{j^+}^{(t)} + \eta c \bar{\mathbf{e}}_{j^-}^{(t)} = \bar{\mathbf{e}}_i^{(t)} - \tilde{c}_1 \bar{\mathbf{e}}_{j^+}^{(t)} - \tilde{c}_2 \bar{\mathbf{e}}_{j^-}^{(t)} \tag{12}$$

Eq. 12 is based on Eq. 9, where $\bar{\mathbf{e}}_i^{(u)}$ denotes the estimated unlearned embedding, $\bar{\mathbf{e}}_i^{(t)}, \bar{\mathbf{e}}_{j^+}^{(t)}, \bar{\mathbf{e}}_{j^-}^{(t)}$ are trained embeddings before unlearning, and $\tilde{c}_1, \tilde{c}_2 > 0$. Since the original negative pair (u_i, v_{j^-}) has already been pushed away from each other, it is unnecessary to do that again. The only valuable part for unlearning is $\bar{\mathbf{e}}_i^{(u)} \approx \bar{\mathbf{e}}_i^{(t)} - \tilde{c}_1 \bar{\mathbf{e}}_{j^+}^{(t)}$ because only (u_i, v_{j^+}) is changed from positive to negative. Extending Eq. 12 with Eq. 10, we can similarly obtain:

$$\bar{\mathbf{e}}_i^{(u)} \approx \bar{\mathbf{e}}_i^{(t)} - \tilde{c}_{i,j_1^+}\bar{\mathbf{e}}_{j_1^+}^{(t)} - \tilde{c}_{i,j_2^+}\bar{\mathbf{e}}_{j_2^+}^{(t)} -, \cdots, -\tilde{c}_{i,j_n^+}\bar{\mathbf{e}}_{j_n^+}^{(t)} \tag{13}$$

$$\bar{\mathbf{E}}^{(u)} \approx \bar{\mathbf{E}}^{(t)} - \left(\tilde{\mathbf{C}} \odot \bar{\mathbf{A}}_\Delta\right) \cdot \bar{\mathbf{E}}^{(t)} \tag{14}$$

From Eq. 13, Eq. 14, we find that matrix $\bar{\mathbf{A}}_\Delta$ happens to be the symmetric adjacent matrix constructed from the unlearning edges' set $\mathcal{E}_\Delta$. $\tilde{\mathbf{C}}$ is the coefficient matrix and $\odot$ denotes the element-wise multiplication. Most importantly, $\bar{\mathbf{A}}_\Delta$ actually reflects the interdependence among all unlearning requests, which collaboratively influence the direction of embedding distribution drift druing unlearning. So, we define $\bar{\mathbf{A}}_\Delta$ as the **Influence Dependency Matrix** (IDM), the graph defined by which is correspondingly called the Influence Dependency Graph (IDG). The graph built by $\mathcal{E}_r$ is then referred to as Residual Graph (RG). $\bar{\mathbf{A}}$ takes effect during learning, while $\bar{\mathbf{A}}_\Delta$ takes effect during unlearning, serving as a low-pass filter.

3.2 Learnable Influence Estimation

3.2.1 **Trainable Influence Encoder.** Inspired by IDM, we empirically propose a trainable encoder $\tilde{\mathbf{E}}_0 = \mathbf{IE}(\bar{\mathbf{A}}_\Delta, \mathbf{E}_0)$ that takes the unlearning requests $\mathcal{E}_\Delta$ (to directly construct IDM $\bar{\mathbf{A}}_\Delta$), and the original 0-layer embeddings of the trained model $\boldsymbol{M}$ to be unlearned as inputs, and outputs the revised 0-layer embeddings $\tilde{\mathbf{E}}_0$.

$$\bar{\mathbf{H}} = \sum_{l=0}^{L_u - 1} \mathbf{H}_l, \qquad \mathbf{H}_l = \mathbf{D}_\Delta^{-\frac{1}{2}} \cdot \bar{\mathbf{A}}_\Delta \cdot \mathbf{D}_\Delta^{-\frac{1}{2}} \cdot \mathbf{H}_{l-1} \tag{15}$$

where $\mathbf{D}_\Delta$ is the diagonal degree matrix for $\bar{\mathbf{A}}_\Delta$ and $\mathbf{H}_l \in \mathbb{R}^{(I+J)\times d}$ denotes the trainable **Influence Estimation Matrix** (IEM) that will be iteratively propagated along the graph IDG for L_u times, resulting in the readout IEM $\bar{\mathbf{H}}$, in which each row represents a trainable influence estimation vector for each node (user/item) due to unlearning, e.g., $\mathbf{h}_i, \mathbf{h}_j \in \bar{\mathbf{H}}$ for user i and item j. Multiple iterations of propagation and aggregation inject more multi-hop influence dependencies and structure information of the IDG into the influence matrix IEM. Similarly, we introduce:

$$\mathbf{E}_{w,l} = \mathbf{D}_\Delta^{-\frac{1}{2}} \bar{\mathbf{A}}_\Delta \mathbf{D}_\Delta^{-\frac{1}{2}} \cdot \mathbf{E}_{w,l-1}, \quad \mathbf{E}_{w,0} = \bar{\mathbf{E}} \odot \mathbf{W}_\eta, \ \ l \in [0, L_e) \tag{16}$$

where $\bar{\mathbf{E}} \in \mathbb{R}^{(I+J)\times d}$ is the readout embeddings of model $\boldsymbol{M}$ before unlearning but here they are fixed and non-trainable. $\mathbf{W}_\eta \in \mathbb{R}^{(I+J)\times 1}$ is trainable weight initialized with small values around $\mathbf{0}$. Eq. 16 is inspired by Eq. 14 but injects higher-order information of interdependent influences. We let $\bar{\mathbf{E}}_w := \mathbf{E}_{w,L_e-1}$, the last layer as readout. Combining Eq. 15 and Eq. 16, we have the final estimation:

$$\tilde{\mathbf{E}}_0 = \Delta\bar{\mathbf{E}}_0 + \mathbf{E}_0, \ \ \Delta\bar{\mathbf{E}}_0 = \mathbf{MLP}(\Delta\mathbf{E}_0), \quad \Delta\mathbf{E}_0 = -\bar{\mathbf{E}}_w + \bar{\mathbf{H}} \tag{17}$$

$$\mathbf{MLP}(\mathbf{X}): \quad \mathbf{X}_l = \delta\left(\mathbf{W}_{l-1}\mathbf{X}_{l-1} + \mathbf{b}_{l-1}\right), \quad l \in [0, L_m) \tag{18}$$

where $\tilde{\mathbf{E}}_0$ denotes the revised 0-layer embeddings for $\boldsymbol{M}$ after unlearning and $\mathbf{MLP}(\cdot)$ denotes a multilayer perceptron in which parameters are fixed during pre-training and updated during fine-tuning. The purpose of the entire pipeline from Eq. 15 to Eq. 17 is to quickly calculate the embedding distribution shift $\Delta\bar{\mathbf{E}}_0$ caused by unlearning when requests $\mathcal{E}_\Delta$ arrive. The only trainable parameters during pre-training are $\mathbf{H}_0$ and $\mathbf{W}_\eta$ that are initialized around $\mathbf{0}$ and should be well pre-trained before unlearning requests come.

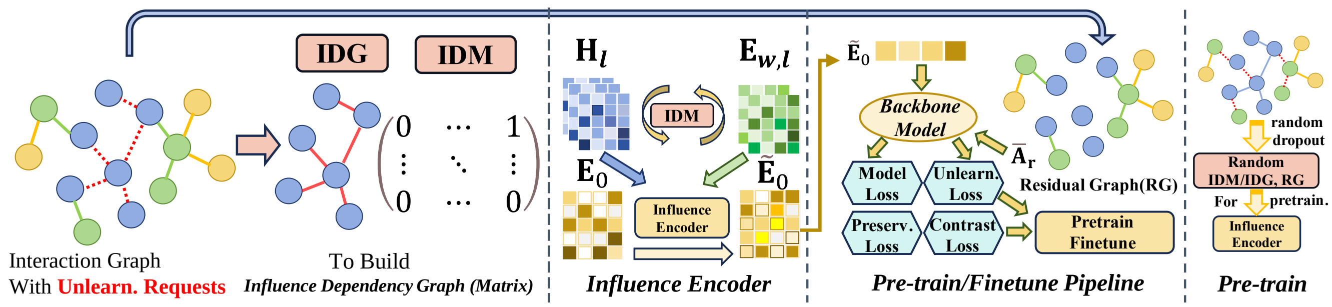


Figure 1: Overall framework of the proposed UnlearnRec paradigm.

3.2.2 **Process Unlearning.** We can directly construct the $\bar{\mathbf{A}}_\Delta$ upon arrival of unlearning requests $\mathcal{E}_\Delta$ and then calculate $\tilde{\mathbf{E}}_0$ through $\mathbf{IE}(\bar{\mathbf{A}}_\Delta, \mathbf{E}_0)$. The unlearned model $\boldsymbol{M}_u$ will be $\mathcal{M}odel(\mathbf{E}_0 := \tilde{\mathbf{E}}_0,\ \bar{\mathbf{A}} := \bar{\mathbf{A}}_r)$ where $\bar{\mathbf{A}}_r$ is the symmetric adjacent matrix built from $\mathcal{E}_r$. Parameters in $\mathbf{IE}(\cdot)$ should be well-pretrained in advance.

3.3 Multi-task Loss Functions for Pretraining

We now introduce the loss functions for UnlearnRec's pretraining paradigm, in which the only trainable parameters are $\mathbf{H}_0$ and $\mathbf{W}_\eta$ during pretraining. $\mathbf{MLP}(\cdot)$ will be updated during fine-tuning.

3.3.1 **Model Loss Function.** In Eq. 19, the trained model $\boldsymbol{M}$ to be unlearned, is initialized with $\tilde{\mathbf{E}}_0$ as the 0-layer embedding and computes the loss $\mathcal{L}_M$ which encompasses various complex loss functions designed by $\boldsymbol{M}$, such as SSL-based loss, BPR loss (unlearned edges should be negative) and others. $\tilde{\mathbf{E}}$ is the revised final embeddings obtained from the forward and readout of $\boldsymbol{M}$ based on $\tilde{\mathbf{E}}_0$ and $\bar{\mathbf{A}}_r$, which is the symmetric adjacent matrix built from $\mathcal{E}_r$.

$$\mathcal{L}_M = \mathcal{L}oss_M\left(\mathcal{M}odel(\tilde{\mathbf{E}}_0,\ \bar{\mathbf{A}}_r)\right),\quad \tilde{\mathbf{E}} = \mathbf{Fwd}_M(\tilde{\mathbf{E}}_0,\ \bar{\mathbf{A}}_r) \tag{19}$$

3.3.2 **Unlearning Loss.** Eq. 20 is referred to as the unlearning loss, which is used to enforce a decrease in the predicted scores of the interaction pairs (i_Δ, j_Δ) to be unlearned, as well as in the corresponding embedding similarities between i_Δ and j_Δ.

$$\mathcal{L}_u = \sum_{(u_{i_\Delta}, v_{j_\Delta})} -\log \operatorname{sigm}(-\tilde{\mathbf{e}}_{i_\Delta}^\top \tilde{\mathbf{e}}_{j_\Delta}),\quad (i_\Delta, j_\Delta) \in \mathcal{E}_\Delta \tag{20}$$

3.3.3 **Preserving Loss.** In Eq. 21, $\Psi(\bar{\mathbf{E}})$ generates a vector describing the embedding distribution of remaining positive pairs (u_i, v_{j^+}) in $\mathcal{E}_r$ based on $\boldsymbol{M}$'s original final embeddings $\bar{\mathbf{E}}$. Analogously, we can obtain $\Psi(\tilde{\mathbf{E}})$ for the distribution of revised final embeddings $\tilde{\mathbf{E}}$.

$$\Psi(\bar{\mathbf{E}}) = \left[\log \frac{\exp\left(\bar{\mathbf{e}}_i^\top \bar{\mathbf{e}}_{j^+}/\tau\right)}{\sum_{(*,v_{k^+})\in\mathcal{E}_r} \exp\left(\bar{\mathbf{e}}_i^\top \bar{\mathbf{e}}_{k^+}/\tau\right)}, \cdots \middle|\ (u_i, v_{j^+}) \in \mathcal{E}_r\right] \tag{21}$$

We can align the two vectors because typically, the unlearning set $\mathcal{E}_\Delta$ represents only a small proportion of the entire $\mathcal{E}$. As a result, the embedding distribution of the remaining unaffected portion retains a significant amount of useful information. This is beneficial to preserving predictive performance of $\boldsymbol{M}$ after unlearning.

$$\mathcal{L}_p = \mathbf{Align}\left(\Psi(\tilde{\mathbf{E}}),\ \ \Psi(\bar{\mathbf{E}})\right) \tag{22}$$

Eq. 22 is named preserving loss and we use L_2 distance as $\mathbf{Align}(\cdot)$.

3.3.4 **Contrast Loss.** The distinct aspect of recommending unlearning compared to general GNN unlearning lies in that when a user unlearns certain interactions, it implies, to some extent, a decrease in the probability of recommending similar interactions.

$$\mathbf{H}'_l = \mathbf{D}_{\Delta'}^{-\frac{1}{2}} \cdot \bar{\mathbf{A}}'_\Delta \cdot \mathbf{D}_{\Delta'}^{-\frac{1}{2}} \cdot \mathbf{H}_{l-1},\quad \bar{\mathbf{A}}'_\Delta = \mathbf{Dropout}(\bar{\mathbf{A}}_\Delta) \tag{23}$$

$\mathbf{Dropout}(\cdot)$ randomly removes a small portion (e.g. $\rho\%$) of unlearning edges from $\bar{\mathbf{A}}_\Delta$ and is then used for forward propagation. Next, we align the generated $\mathbf{H}'_l$ by $\mathbf{Dropout}(\cdot)$ with the original $\mathbf{H}_l$.

$$\mathcal{L}_c = \sum_{i\in\mathcal{U}\cup\mathcal{V}} -\log \frac{\exp\left(\cos(\mathbf{h}_i, \mathbf{h}'_i)/\tau\right)}{\sum_{i'\in\mathcal{U}\cup\mathcal{V}} \exp\left(cos(\mathbf{h}_{i'}, \mathbf{h}'_{i'})/\tau\right)} \tag{24}$$

where $\mathbf{h}_i, \mathbf{h}'_i$ denote row vectors in $\mathbf{H}_l, \mathbf{H}'_l$,respectively. To some extent, it leverages the $\mathbf{H}'_l$ generated on $\bar{\mathbf{A}}'_\Delta$ to predict the complete $\mathbf{H}_l$, thereby implicitly utilizing a partial IDM $\bar{\mathbf{A}}'_\Delta$ to complete and predict the full IDM $\bar{\mathbf{A}}_\Delta$, by which, it enables the incorporation of more contextual influence and correlations. To summarize all of the above, we we obtain the final loss function for pre-training:

$$\mathcal{L} = \mathcal{L}_M + \lambda_u\mathcal{L}_u + \lambda_p\mathcal{L}_p + \lambda_c\mathcal{L}_c = \mathcal{L}oss\left(\boldsymbol{M}, \bar{\mathbf{A}}_\Delta\right) \tag{25}$$

where $\mathcal{L}_M$ contains all the SSL random network structures of model $\boldsymbol{M}$, which will drive the post-unlearning embeddings to conform to the specific distribution characteristics of the model. $\mathcal{L}_p$ is responsible for maintaining the predictive performance while $\mathcal{L}_u$ is used for unlearning, and they represent a trade-off. $\mathcal{L}_c$ controls the influence generalization. The final loss $\mathcal{L}$ can be computed as long as the IDM $\bar{\mathbf{A}}_\Delta$ and the trained model $\boldsymbol{M}$ to be unlearned are given.

3.4 Pretraining Paradigm for Unlearning

We train the **IE** during the pretraining and adjust both the **IE** and the RS model based on unlearning requests during the finetuning.

3.4.1 **Pre-training.** In each simulation round, we randomly select a subset $\mathcal{E}_\Delta^{(s)}$ from $\mathcal{E}$ (e.g., ρ %) as the simulated unlearning set to construct the IDM $\bar{\mathbf{A}}_\Delta^{(s)}$. Then, we optimize the loss function $\mathcal{L}$ to train the parameters. Currently, only $\mathbf{H}_0$ in Eq. 15 and $\mathbf{W}_\eta$ in Eq. 16 are trainable, while the other parameters like $\bar{\mathbf{E}}$ remain fixed. Please refer to lines 1 to 8 of Algorithm 1 for more details.

3.4.2 **Fine-tuning.** The well-pretrained $\mathbf{IE}(\cdot)$ can be directly utilized to perform unlearning. Howver, to achieve better performance and unlearning efficacy, if necessary, we can still fine-tune $\mathbf{IE}(\cdot)$ and the unlearned model $\boldsymbol{M}_u$ when actual unlearning requests arrive. During fine-tuning, we usually only need to optimize $\mathcal{L}_M+\lambda_u\mathcal{L}_u$ to fine-tune the $\mathbf{MLP}(\cdot)$ in $\mathbf{IE}(\cdot)$ (Eq. 18) and $\mathbf{E}_0$ in $\boldsymbol{M}_u$ (Eq. 1), which enables us to accomplish the fine-tuning process very efficiently. Please refer to lines 9 and 10 of Algorithm 1 for more information.

Algorithm 1: Pre-train./Fine-tun. Paradigm of UnlearnRec

```
   Input: Trained model $M$ and graph data $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$
   Output: Well-pretrained $\mathbf{IE}(\cdot)$ and unlearned $M_u$.
1  Initialize $\mathbf{H}_0$ in Eq. 15 and $\mathbf{W}_\eta$ in Eq. 16 around $\mathbf{0}$. Initialize
     $\mathbf{W}_l$,$\mathbf{b}_l$ in Eq. 18 with identity matrix and $\mathbf{0}$, respectively
2  Lock all other parameters and leave only $\mathbf{H}_0$,$\mathbf{W}_\eta$ trainable
3  for run $i_{pre} = 1$ to $N_{pretrain}$ do
4      Sample a subset $\mathcal{E}_\Delta^{(s)}$ from $\mathcal{E}$ to construct IDM $\bar{\mathbf{A}}_\Delta^{(s)}$
5      for epoch $j_{tr} = 1$ to $N_{train}$ do
6          Minimize $\mathcal{L}oss(M, \bar{\mathbf{A}}_\Delta^{(s)})$ (Eq. 25) to update $\mathbf{H}_0$, $\mathbf{W}_\eta$;
7      end
8  end
9  Actual unlearning requests $\mathcal{E}_\Delta$ come, $\mathcal{E}_r = \mathcal{E} \setminus \mathcal{E}_\Delta$, build $\bar{\mathbf{A}}_r$.
10 Fix all parameters except those in $\mathbf{MLP}(\cdot)$ and $\mathbf{E}_0$. Minimize
     $\mathcal{L}_M + \lambda_u\mathcal{L}_u$ based on ($\tilde{\mathbf{E}}_0$,$\bar{\mathbf{A}}_r$) to fine-tune $\mathbf{MLP}(\cdot)$ and $\mathbf{E}_0$.
11 Return Well-pretrained $\mathbf{IE}(\cdot)$ and $M_u = \mathcal{M}odel(\tilde{\mathbf{E}}_0, \bar{\mathbf{A}}_r)$
```

4 Evaluation

We conducted numerous experiments to validate our paradigm UnlearnRecand address the following research questions (RQs):

- **RQ1**: Can the proposed paradigm UnlearnRec effectively preserve the predictive performance of the model after unlearning, compared to baselines and retrained model?
- **RQ2**: Can our UnlearnRec paradigm effectively unlearn the requested interactions in comparison to the baseline approaches?
- **RQ3**: How do the components of the proposed UnlearnRec paradigm affect the effectiveness of unlearning?
- **RQ4**: How does our UnlearnRec method alter the distribution curve of the embeddings of the model that needs to be unlearned?
- **RQ5**: How efficient is our UnlearnRec approach in comparison to existing techniques in terms of time and memory?
- **RQ6**: Can the UnlearnRec method effectively reduce the recommended probability of items similar to the unlearned items?

4.1 Experimental Settings

4.1.1 **Evaluation Datasets.** We validate the unlearning effectiveness of our UnlearnRec using three commonly used real-world datasets: Movielens-1M, Gowalla, and Yelp2018. **Movielens-1M** is widely used public user behavior collection, containing one million ratings from users on movies while the **Gowalla** dataset comprises user check-in records at geographic locations between January and June 2010, collected from the Gowalla platform. And the **Yelp2018** dataset is derived from the Yelp platform and includes user ratings on venues spanning January to June 2018. For training the model $\boldsymbol{m}$ to be unlearned, we allocate 20% of the total interaction edges as the test set, while the remaining edges are utilized for training.

Table 1: Statistical details of experimental datasets.

Dataset	# Users	# Items	# Interactions	Interaction Density
Gowalla	25557	19747	294983	5.85×10^{-4}
Yelp2018	31668	38048	1561406	1.30×10^{-3}
Movielens-1M	6940	3706	1000209	3.89×10^{-2}

4.1.2 **Prediction Metrics.** Following the standard evaluation protocols commonly used for the CF tasks [28, 36], We employ a full-rank evaluation approach where we rank all uninteracted items alongside the positive items from the test set for each user. We utilize two widely used metrics, namely Recall@N and NDCG@N [27, 29], to assess the prediction performance of the trained model.

4.1.3 **Unlearning Metrics.** Following the common validation methods of other unlearning works [30, 31], we conduct adversarial attacks and utilize the *Membership Inference* (MI) [21] as the metric to evaluate the unlearning effectiveness. MI can indicate whether the unlearned model leaks information about the unlearned edges, specifically whether the existence of unlearned edges can be inferred from the unlearned model's embeddings, or remain completely concealed. We use MI-BF to denote the ratio of the average probability for presence of edges in $\mathcal{E}_\Delta$ before and after unlearning, and MI-NG to measure the ratio between average recommended probability of unlearned edges in model $\boldsymbol{M}_u$ and probability of edges obtained through negative sampling. MI-BF and MI-NG should exceed 1 and the higher, the better.

4.1.4 **Adversarial Attacks.** Following other unlearning works [30, 31], we conduct adversarial attacks to validate our proposed paradigm. Specifically, we select adversarial edges by randomly sampling the least probable edges, with their probability based on a GCN trained on the original dataset. Afterwards, we train the GNN models using the attacked graph that incorporates these adversarial edges. Subsequently, we employ all baseline unlearning methods to unlearn the adversarial edges, and evaluate the unlearning performance of each baseline using the aforementioned metrics.

4.1.5 **Backbone Models.** We employ the following GNN models as our backbone models and the targets for unlearning to evaluate the effectiveness of our paradigm in unlearning.

- **LightGCN** [13] utilizes fundamental GCN architectures to enhance performance in recommendation tasks ("GCN" for short).
- **SGL** [29] presents multiple techniques for enhancing graph contrastive learning through graph and feature augmentations
- **SimGCL** [35] is a contrastive learning model with simple feature-level augmentation techniques utilizing random permutation.

4.1.6 **Baseline Methods.** We compare our UnlearnRec paradigm with the state-of-the-art baselines from various perspectives.

- **GIF** [30] is a state-of-the-art method for graph unlearning that employs the influence function (IF) to estimate the shifts in parameters related to the graph structures needing to be unlearned.
- **CEU** [31] is a state-of-the-art IF-based method designed to expedite the unlearning process and providing theoretical guarantee.
- **GraphEraser** [7] is a partition-based approach for unlearning, where the graph is divided into multiple shards based on nodes using embedding clustering and community detection techniques.

These shards are subsequently combined and when doing unlearning, the method just selectively retrains the relevant shards.

- **RecEraser** [4] is a recent partition-based method for recommendation unlearning, which differs from GraphEraser by partitioning the edges instead of nodes into shards to avoid losing edge information. Additionally, it incorporates a more sophisticated attention aggregation method with more parameters.
- **Retrain**. When unlearning requests are received, a full retraining of the entire graph model is carried out with model parameters being re-initialized. This can be regraded as ground truth.

4.1.7 **Implementation Settings.** We develop our UnlearnRec using PyTorch and employ the Adam optimizer with its default parameters. $\mathbf{H}_0$ and $\mathbf{W}_\eta$ in influence encoder $\mathbf{IE}(\cdot)$ are initialized around $\mathbf{0}$ before pre-training and they are fixed during fine-tuning. $\mathbf{W}_l$,$\mathbf{b}_l$ in $\mathbf{MLP}(\cdot)$ (Eq. 18) are initialized with identity matrix and $\mathbf{0}$, respectively and they are non-trainable until fune-tuning phase. The batch size for pretraining is selected from {512, 1024} and the embedding size is 128 for all models. Besides, we use 3 layers for all GNN models, 3 layers for Eq. 15,16, and 2 layers for $\mathbf{MLP}(\cdot)$. $\rho \in \{5, 15, 20\}$ and λ_u is chosen from {1,0.5,0.1} while λ_p is tuned from {1,0.1,0.01,0.005,0.001}. And λ_c is selected from $\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$. All the temperatures τ are chosen from {0.1, 1, 10}. We use the released code for baselines with grid search and we conduct all the tests on the same device with an NVIDIA GeForce RTX 3090 GPU.

4.2 Model Utility Analysis (RQ1)

We commence the evaluation of the unlearning utility of the proposed UnlearnRec paradigm in comparison to the baseline methods. The results are tabulated in Table 2, and the following observations can be made based on the Recall@20 and NDCG@20 results.

- **Superior performance retention ability**: After unlearning the well-trained model, our method demonstrates excellent performance retention across multiple datasets compared to retraining. It is noteworthy that on dense datasets such as Movielens, all baselines can maintain relatively good predictive performance. The performance retention capability of the IF-based unlearning method is superior to partition-based methods. This is because partitioning the graph inevitably disrupts its structure, which can have a negative impact on predictive performance.
- **Drawbacks of partition-based methods**: Traditional partition-based methods partition the graph and, upon receiving an unlearning request, only retrain the affected shards. This approach, which disrupts the graph structure, undoubtedly impacts the predictive performance of GNN and requires time for unlearning. However, within the partition-based category, RecEraser slightly outperforms GraphEraser in terms of performance retention. This is because RecEraser partitions graphs based on edges, which is more reasonable for recommendation systems that prioritize interactive edges. Compared to IF-based methods, partition-based methods may exhibit lower predictive performance, but offer more controllable unlearning effects. These methods require careful training of the affected shards.
- **Limits for IF-based methods**: IF-based methods calculate unlearning impact through mathematical computations, exhibiting excellent performance retention as unlearning effects can be minimal with little embedding distribution change. Their advantage is computational speed through direct IF calculations. However, this end-to-end estimation may not be accurate for SSL-based GNNs with stochastic structures, leading to influence underestimation and average unlearning effects in practice.

4.3 Model Efficacy Analysis (RQ2)

We proceed to evaluate and analyze the effects of unlearning, with the experimental results presented in Table 2. The following discussions primarily revolve around the MI-BF and MI-NG metrics.

- **Outstanding unlearning efficacy.**: Based on MI-BF and MI-NG metrics, UnlearnRec shows strong unlearning results. Across datasets and backbones, UnlearnRec achieves MI-BF and MI-NG values above 1. MI-BF > 1 shows reduced probability of recommending edges marked for unlearning versus before. Similarly, MI-NG > 1 means unlearned edges are less likely to be recommended than negative samples, protecting user privacy after unlearning. This shows UnlearnRec effectively simulates edge removal through pre-training and identifies these effects via fine-tuning. UnlearnRec works with various SSL-based GNNs and matches each model's embedding distribution through unlearning (detailed later). This comes from using $\mathcal{L}_M$ to capture loss functions, training methods, random processes, and network structure of each GNN, yielding similar embedding distributions.
- **Limitations of partition-based methods.**: As mentioned earlier, partitioning the graph disrupts its overall integrity, leading to negative impacts on predictive performance after unlearning. Additionally, many SSL-based methods are designed to address the sparsity issue in recommendation systems, where the interaction graph is sparse. Consequently, in SSL-based GNNs, when an unlearning request is made, almost all edges and nodes are more or less affected since they are involved in the formation and maintenance of self-supervised labels. Therefore, training only a subset of shards is unlikely to yield satisfactory unlearning results.Specifically, it is challenging to achieve MI-NG > 1.5.
- **Ineffectiveness of IFs for SSL-based GNNs.**: IF-based methods need complex mathematical designs. But training SSL-based GNNs is more random and unpredictable. SSL-based GNNs use random network structures and signals, making gradient estimation hard. This makes IF methods less effective for SSL-based GNNs, leading to poor unlearning. Table 2 shows that with SimGCL and SGL backbones, CEU and GIF perform poorly on MI-NG, sometimes below 1. This means that while unlearning reduces recommendations of learned edges, these rates stay higher than for negative samples. This suggests learned edges were once positive samples, risking user data leaks.

4.4 Ablation Study (RQ3)

We conducted the ablation experiments on the Movielens and Gowalla datasets using the SimGCL model as the backbone and made the following findings. The results are shown in Table 3

- -$\mathcal{L}_u$: When we remove the loss function $\mathcal{L}_u$, we observe that the model's predictive performance remains relatively unchanged in terms of Recall and NDCG. However, there is a significant

Table 2: Overall unlearning utility and efficacy comparison on Gowalla, Movielens-1M, and Yelp2018 datasets.

Method		Retrain			RecEraser			GraphEraser			CEU			GIF			Ours		
Backbone		GCN	SimGCL	SGL	GCN	SimGCL	SGL	GCN	SimGCL	SGL	GCN	SimGCL	SGL	GCN	SimGCL	SGL	GCN	SimGCL	SGL
ML-1m	Recall	0.2287	0.2336	0.2339	0.2033	0.2106	0.2099	0.2008	0.1995	0.1989	0.2219	0.2239	0.2234	0.2111	0.2117	0.2177	0.2282	0.2336	0.2330
	NDCG	0.3209	0.3325	0.3328	0.2807	0.2978	0.2887	0.2709	0.2752	0.2797	0.3152	0.3217	0.3163	0.2912	0.3067	0.3031	0.3206	0.3323	0.3296
	MI-BF	9.7865	6.2314	20.984	1.8103	1.8863	1.9968	1.7327	1.9054	2.1676	1.0601	1.8883	1.1281	1.0426	1.5980	1.1063	6.5071	3.1981	14.0285
	MI-NG	6.2327	3.9845	11.344	1.6734	1.2572	1.6589	1.3160	1.2202	1.8691	0.7462	1.8018	0.8396	0.9351	1.6629	0.8123	**4.2766**	**2.3575**	**9.3661**
Gowalla	Recall	0.2407	0.2605	0.2502	0.2211	0.2367	0.2279	0.2191	0.2238	0.2213	0.2390	0.2516	0.2421	0.2391	0.2486	0.2394	0.2398	0.2540	0.2477
	NDCG	0.1584	0.1703	0.1618	0.1432	0.1453	0.1486	0.1399	0.1453	0.1429	0.1505	0.1618	0.1558	0.1508	0.1599	0.1524	0.1545	0.1670	0.1611
	MI-BF	7.6745	15.4290	8.7485	1.4975	1.2150	1.7941	1.5865	1.0238	1.8937	1.1470	1.0322	1.1283	1.0030	1.0432	1.0076	5.3729	4.3520	2.6285
	MI-NG	3.9265	7.7154	4.4323	0.9473	0.8916	1.0357	0.8182	0.6511	0.9995	0.5345	0.5644	0.5317	0.5059	0.5883	0.5108	**2.8091**	**1.8075**	**1.3369**
Yelp	Recall	0.0625	0.0648	0.0635	0.0587	0.0614	0.0599	0.0574	0.0610	0.0578	0.0597	0.0633	0.0621	0.0580	0.0630	0.0624	0.0617	0.0640	0.0633
	NDCG	0.0511	0.0531	0.0512	0.0475	0.0501	0.0482	0.0466	0.0478	0.0460	0.0493	0.0516	0.0497	0.0461	0.0513	0.0501	0.0501	0.0519	0.0513
	MI-BF	6.2634	11.1054	9.9890	1.5473	1.2490	1.4003	1.4832	1.2837	1.3896	1.1415	1.4724	1.0751	1.0113	1.5555	1.0503	5.6431	6.0197	4.0866
	MI-NG	3.3154	5.8891	4.4316	0.9043	0.6777	0.6195	0.8968	0.6873	0.6536	0.6055	0.7977	0.5801	0.5188	0.8539	0.5689	2.9978	2.4956	2.0492

Table 3: Ablation study of UnlearnRec based on SimGCL.

Datasets	Movielens-1M				Gowalla			
Metrics	Recall	NDCG	MI-BF	MI-NG	Recall	NDCG	MI-BF	MI-NG
- $\mathcal{L}_u$	0.2338	0.3331	1.5347	0.5948	0.2543	0.1677	1.3452	0.5639
- $\mathcal{L}_p$	0.1863	0.2682	4.8245	3.7239	0.17107	0.1117	6.8021	3.3367
- $\mathcal{L}_c$	0.2300	0.3262	3.0013	2.1789	0.2507	0.1633	4.1979	1.6625
- FineTune	0.2211	0.3188	2.2226	1.6998	0.2398	0.1563	3.3232	1.3047
Ours	0.2336	0.3323	3.1981	2.3575	0.2540	0.1670	4.3520	1.8075

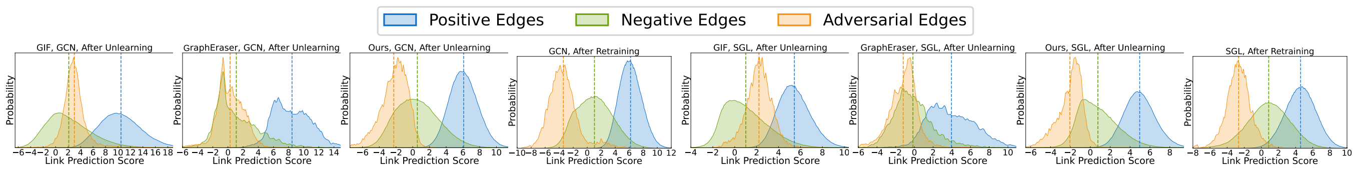
drop in the unlearning efficacy, specifically MI-BF and MI-NG. This is because $\mathcal{L}_u$ plays a role in reducing the probability of recommending edges that need to be unlearned. Nevertheless, we found that even without $\mathcal{L}_u$, MI-BF still exceeds 1, indicating a decrease in the probability of recommending the unlearned edges. This is due to the presence of the BPR loss in $\mathcal{L}_M$, which contributes to reducing the similarity between the two nodes in the interaction pairs that need to be unlearned. However, solely relying on this BPR loss is insufficient to eliminate the effects caused by the unlearning edges. Therefore, MI-NG remains below 1, indicating that the probability of recommending unlearned edges is still higher than that of negative samples.

- -$\mathcal{L}_p$: The purpose of $\mathcal{L}_p$ is to preserve useful information from the original trained model to some extent. Its goal is to prevent the model from deviating too far from the original embedding distribution during the pre-training process, where the edges requiring unlearning are randomly simulated. After all, in practice, the edges that need to be unlearned typically constitute a small portion of all edges. Removing $\mathcal{L}_p$ significantly compromises the predictive performance during the unlearning process, even though the unlearning effects, such as MI-BF and MI-NG, may increase. However, this trade-off is not worth it.
- -$\mathcal{L}_c$: The influence of $\mathcal{L}_c$ on the overall outcome, whether it is utility or unlearning efficacy, is relatively small. $\mathcal{L}_c$ can be considered as a regularization term, primarily used to smooth the randomly simulated unlearning requests during the pre-training process. Therefore, the weight of $\mathcal{L}_c$, denoted as λ_c, does not need to be very large. To some extent, the smoothing effect of $\mathcal{L}_c$ on unlearning requests may slightly decrease the probability of recommending the interactions similar to unlearning requests. However, generally speaking, these interactions are still regarded as positive pairs by the recommendation system and are still likely to be recommended like other normal positive instances.
- -*FineTune*: In our experiment of adversarial attacks, we found that even without fine-tuning the pre-trained influence encoder (IE), we could still achieve good results (both in utility and efficacy) when unlearning requests were received and directly applied to unlearn using **IE**. This indicates that our pre-trained **IE** is capable of effectively identifying and mitigating the impact of adversarial edges on embeddings. In fact, we can consider the $\Delta\bar{\mathbf{E}}_0$ in **IE** as learnable noises. When unlearning requests arrive, the embeddings of affected nodes are first masked by the noises $\Delta\bar{\mathbf{E}}_0$ in **IE**, and then fine-tuned to adapt to the specific contextual changes brought by the unlearning requests. To put it simply, the pre-training stage is analogous to the noise injection process of a diffusion model, while the fine-tuning stage is similar to the reconstruction and generation process of diffusion [9, 14, 25].

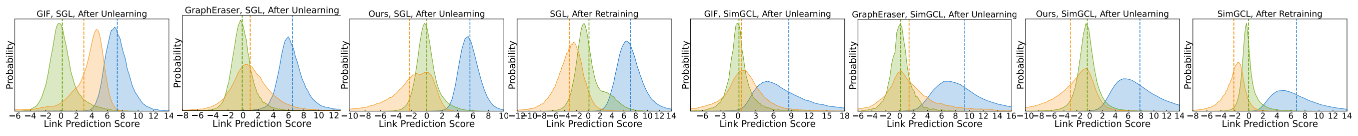
4.5 Embedding Distribution Shift Study (RQ4)

In this section, we mainly discuss the changes in the embedding distribution of various recommendation system models before and after unlearning, as well as their comparison with the results obtained from retraining. The predicted score distribution before unlearning is shown in Fig. 3, while the distribution after unlearning is shown in Fig. 2. The dashed lines in the figures represent the mean predicted scores for the corresponding types of edges. Based on these results, we have the following observations.

- **Adversarial Edges**. Before unlearning, the predicted scores for all adversarial edges in various models are close to positive instances, which is reasonable since they were labeled as positive during the training process. These Adversarial Edges are sampled based on a set of items that a trained GCN predicts as the least likely to be interacted by users. Therefore, after removing the adversarial edges and retraining, we found that the mean predicted scores for them are lower than regular negative instances.
- **P-based Methods vs. IF-based Methods**. In general, P-based methods demonstrate better controllability and efficacy in terms of unlearning efficacy compared to IF-based methods. From Fig. 2, it can be observed that the mean predicted scores for adversarial edges given by GraphEraser are more closer to the mean of negative instances, while in many backbones, GIF still exhibits significantly higher predicted scores for adversarial edges compared to negative edges (e.g., GIF for SGL in both datasets). Although GraphEraser outperforms GIF in terms of unlearning efficacy, it still falls short of meeting the requirements, as evident in the picture of the SGL on the Yelp dataset, where the scores for adversarial edges still remain higher than negative instances. This implies that even after unlearning, it is still possible to infer

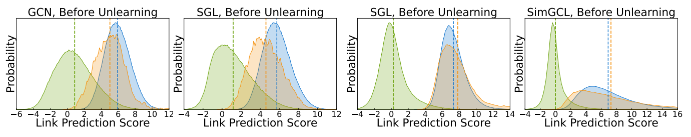


(a) Prediction distribution of GCN after unlearning on Movielens-1M dataset (b) Prediction distribution of SGL after unlearning on Movielens-1M dataset



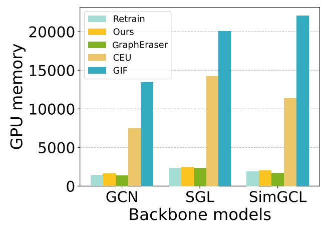
(c) Prediction distribution of SGL after unlearning on Yelp2018 dataset (d) Prediction distribution of SimGCL after unlearning on Yelp2018 dataset

Figure 2: Visualization for the predictions of positive, negative, and adversarial edges in the Movielens-1M and Yelp2018 datasets. Four methods with three backbone models are compared, including the IF-based method GIF, the partition-based method GraphEraser, the retraining-based exact unlearning, and our proposed UnlearnRec, on the GCN, SGL, SimGCL backbones.

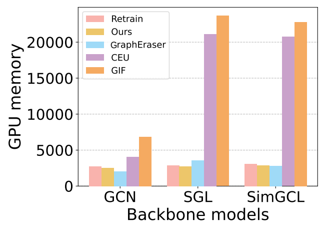


(a) Distribution on ML-1M before unleaning (b) Distribution on Yelp before unleaning

Figure 3: Visualization of the prediction distributions for Movielens-1M and Yelp2018 datasets before unlearning.



(a) GPU memory usage on Movielens-1M



(b) GPU memory usage on Gowalla

Figure 4: Comparison of GPU memory usage with different methods on various backbones based on two datasets.

the existence of these edges based on the embeddings.

- **Our UnlearnRec Paradigm**. Our method demonstrates good unlearning efficacy on both datasets and across three backbones. In all experiments, after unlearning, the mean predicted scores for adversarial edges are significantly lower than negative instances. Furthermore, the score distribution of adversarial edges is enveloped by the score distribution of negative instances, ensuring minimal information leakage about these edges after unlearning. In other words, it becomes impossible to infer the existence of these edges based on the embeddings. Visually, the embedding distribution of our model closely resembles that of retraining.

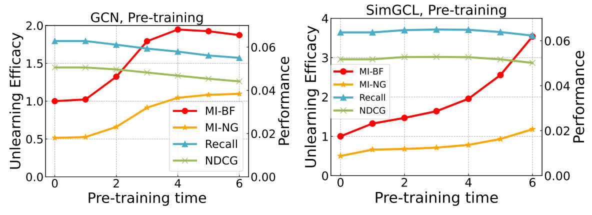
4.6 Unlearning Efficiency Analysis (RQ5)

To assess the efficiency of our UnlearnRec paradigm, we conduct a comparative analysis of various models in terms of GPU memory and processing time when unlearning requests come. Fig. 4 shows and GPU memory cost and Tab. 4 shows the processing time, in which *Ours(0)* means no finetuning and *Ours(1)* denotes funetuning for 1 epoch. And we have following discussions.

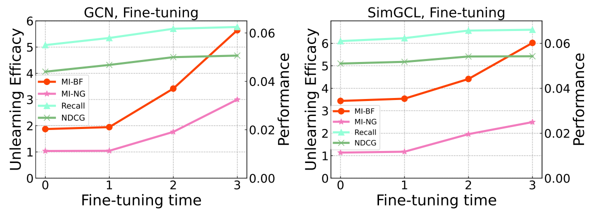
Table 4: The time for processing unlearning requests of different methods based on the SimGCL model and three datasets.

Dataset	Retrain	GraphEraser	GIF	CEU	Ours(0)	Ours(1)	Ours(2)	Ours(3)
ML-1M	1208.354	993.064	8.278	6.943	0.312	7.034	14.221	21.517
Yelp2018	3029.606	1834.413	146.384	52.216	0.468	19.71	39.201	56.023
Gowalla	839.07	795.995	131.753	39.089	0.432	3.37	6.734	10.026

- **High Memory Cost for IF-based Methods**. IF methods typically require second-order gradients (Hessian matrix) to estimate unlearning effects. This requires storing computation graphs and using the first-order graph to calculate Hessian, consuming significant GPU memory. With sufficient GPU memory for the full dataset's second-order graph, unlearning is fast (like GIF/CEU on Movielens). Without sufficient memory, batching becomes necessary, increasing time and reducing IF accuracy. CEU utilizes HVP [22] and CG [26] to estimate inverse Hessian, saving some computational resources. However, it still requires storing graphs, which are complex due to SSL-based GNNs' stochastic structures. This increases computational complexity and reduces estimation accuracy of IF-based unlearning methods.
- **High Time Cost for P-based Methods**. P-based methods require retraining the shards affected by unlearning requests and then aggregating all the shards. This process takes time, especially when unlearning requests involve a large number of shards. P-based methods are relatively efficient in terms of GPU memory consumption. Their GPU consumption is comparable to retraining, or sometimes even lower, as they only need to retrain a portion of the graph. P-based methods are relatively controllable, and with patient training, they can achieve good unlearning efficacy. However, due to the disruption of the graph's topology, they may suffer from poor utility performance. Moreover, the prolonged retraining of multiple shards reduces the practicality.
- **UnlearnRec'Advantages**. Our method strikes a good balance between memory and time. The GPU memory required by our paradigm is comparable to that of P-based methods and retraining, and significantly smaller than that of IF-based methods. The processing time is also much shorter compared to P-based methods and retraining, while being comparable to IF on relatively large datasets. In our adversarial attack experiments, we achieve



(a) Unlearning efficacy and performance changes during pre-training



(b) Unlearning efficacy and performance changes during fine-tuning

Figure 5: Unlearning efficacy and performance v.s. pre-training and fine-tuning processes of UnlearnRec.

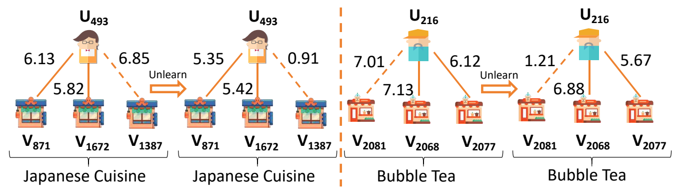


Figure 6: Case study on the unlearning effects for user-item interactions of similar categories based on Yelp2018 dataset.

satisfactory pretrained **IE** in around 10-15 pretraining epochs, and when specific unlearning requests arise, fine-tuning can be completed in 3-5 epochs. These can be found in Fig. 5.

4.7 Unlearning Case Study (RQ6)

Here, we provide some examples of unlearning cases in Fig. 6. It can be observed that when a particular interaction is unlearned, its prediction score is significantly reduced (comparable to that of a negative interaction edge), and the prediction scores of items similar to it also decrease slightly. However, the scores for these similar items remain sufficiently high for them to be considered positive instances by the recommendation system and potentially recommended. When a user actively unlearns a significant number of similar items, the probability of recommended items from that category might also drop noticeably. This phenomenon aligns with our common sense: typically, when a user repeatedly removes interactions with the same type of items, it usually implies that they wish to no longer be recommended similar items. This demonstrates our method's strong generalization capability in unlearning tasks, along with its sophisticated modeling of complex dependencies and mutual influences between multiple unlearning requests.

4.8 Unlearning Ratio Study

Table 5 summarizes the unlearning results of removing varying proportions of edges from the original dataset, evaluated through both model utility preservation and unlearning effectiveness metrics. In reality, unlearning requests typically only account for a small fraction of the dataset. From Tab. 5, it can be observed that as the proportion of unlearning increases, the predictive performance of the recommendation system slightly decreases in order to maintain a high level of unlearning efficacy. And our method maintains consistently high unlearning efficacy (MI-NG > 1 across all tests) when processing requests to remove varying proportions of user-item interactions. This demonstrates the robustness of our approach.

Table 5: Utility and efficacy *w.r.t.* different unlearning ratios.

Unlearn Ratios	0.10%	0.30%	0.50%	0.75%	1.00%	1.50%	2.00%
Recall	0.2564	0.2543	0.2504	0.2462	0.2411	0.2321	0.2302
NDCG	0.1682	0.1653	0.1641	0.1634	0.1615	0.1598	0.1557
MI-BF	4.8497	5.8521	6.1612	7.0485	7.8845	8.4836	9.7556
MI-NG	1.1943	1.1670	1.0715	1.0401	1.0283	1.0306	1.0288

5 Related Work

Partition-Based Approaches to Machine Unlearning. Machine unlearning [2, 3, 8] has garnered significant attention in recent years, with the goal of removing the influence of specific training data from model parameters while avoiding full retraining. Several approaches address these challenges across different domains. The *SISA* concept [7] partitions models into multiple shards and performs unlearning by retraining only the relevant shards. Similarly, [17] enhances collaborative filtering recommendation through optimized sequential training and hypergraph-based user clustering, enabling efficient recommendation unlearning.

Influence Function Methods for Unlearning. Other efforts improve time efficiency by eliminating the retraining process entirely. Influence Function approaches represent a prominent direction, using mathematical methods to quickly estimate unlearning impact and directly modify model parameters with minimal computational overhead. For instance, [12] introduces certified data removal using influence functions, while [15] proposes approximate unlearning based on influence analysis for linear and logistic models. [38] leverages influence functions for recommendation unlearning, achieving substantial speedup compared to retraining. A recent survey [18] systematizes design principles and taxonomies while highlighting challenges like dynamic unlearning and interpretability, underscoring the unique complexities of recommendation unlearning compared to traditional unlearning paradigms.

6 Conclusion

In this work, we introduce UnlearnRec: a novel pretraining paradigm for recommendation unlearning that is both model-agnostic and learnable. Our approach effectively addresses the complex challenges of unlearning in modern recommender systems, particularly excelling in advanced SSL-based architectures where traditional methods fall short. Through a strategic two-phase process of comprehensive pre-training followed by targeted fine-tuning with carefully designed optimization objectives, we demonstrate through extensive empirical evaluation that UnlearnRec achieves a good balance. Our experiments across multiple benchmark datasets confirm that our method delivers exceptional unlearning performance while maintaining computational efficiency and memory utilization.

References

[1] Privacy Act. 2000. Personal information protection and electronic documents act. *Department of Justice, Canada. Full text available at http://laws. justice. gc. ca/en/P-8.6/text. html* (2000), 4356–4364.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.

[3] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 463–480.

[4] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*. 2768–2777.

[5] Guoxuan Chen, Lianghao Xia, and Chao Huang. 2025. LightGNN: Simple Graph Neural Network for Recommendation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*. 549–558. doi:10.1145/3701551.3703536

[6] 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 Conference on Artificial Intelligence (AAAI)*, Vol. 34. 27–34.

[7] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 499–513.

[8] Weilin Cong and Mehrdad Mahdavi. 2022. GraphEditor: An Efficient Graph Representation Learning and Unlearning Approach. (2022).

[9] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2023. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023).

[10] Yonatan Dukler, Benjamin Bowman, Alessandro Achille, Aditya Golatkar, Ashwin Swaminathan, and Stefano Soatto. 2023. Safe: Machine unlearning with shard graphs. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 17108–17118.

[11] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems (TRS)* 1, 1 (2023), 1–51.

[12] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030* (2019).

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

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Informationf Processing Systems (NeurIPS)* 33 (2020), 6840–6851.

[15] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. 2021. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2008–2016.

[16] Chanhee Kwak, Junyeong Lee, Kyuhong Park, and Heeseok Lee. 2017. Let machines unlearn–machine unlearning and the right to be forgotten. (2017).

[17] Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Junlin Liu, and Jun Wang. 2024. Making recommender systems forget: Learning and unlearning for erasable recommendation. *Knowledge-Based Systems* 283 (2024), 111124.

[18] Yuyuan Li, Xiaohua Feng, Chaochao Chen, and Qiang Yang. 2024. A Survey on Recommendation Unlearning: Fundamentals, Taxonomy, Evaluation, and Open Questions. *arXiv preprint arXiv:2412.12836* (2024).

[19] Zongwei Li, Lianghao Xia, and Chao Huang. 2024. Recdiff: diffusion model for social recommendation. In *International Conference on Information and Knowledge Management (CIKM)*. 1346–1355.

[20] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *ACM Web Conference (WWW)*. 2320–2329.

[21] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 11–20.

[22] Barak A Pearlmutter. 1994. Fast exact multiplication by the Hessian. *Neural computation* 6, 1 (1994), 147–160.

[23] Formerly Data Protection. 2018. General data protection regulation (GDPR). *Intersoft Consulting, Accessed in October* 24, 1 (2018).

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[25] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*. PMLR, 2256–2265.

[26] Trond Steihaug. 1983. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.* 20, 3 (1983), 626–637.

[27] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. *International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)* (2023).

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 165–174.

[29] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 726–735.

[30] Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. 2023. GIF: A General Graph Unlearning Strategy via Influence Function. In *ACM Web Conference (WWW)*. 651–661.

[31] Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. 2023. Certified edge unlearning for graph neural networks. In *International Conference on Knowledge Discovery and Data Mining (KDD)*. 2606–2617.

[32] Kun Wu, Jie Shen, Yue Ning, and Wendy Hui Wang. 2022. Fast Yet Effective Graph Unlearning through Influence Analysis. (2022).

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 32, 1 (2020), 4–24.

[34] Lianghao Xia, Chao Huang, Chunzhen Huang, Kangyi Lin, Tao Yu, and Ben Kao. 2023. Automated Self-Supervised Learning for Recommendation. In *The ACM Web Conference (WWW)*. 992–1002.

[35] 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 *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1294–1303.

[36] An Zhang, Wenchang Ma, Xiang Wang, and Tat-Seng Chua. 2022. Incorporating bias-aware margins into contrastive loss for collaborative filtering. *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), 7866–7878.

[37] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* 52, 1 (2019), 1–38.

[38] Yang Zhang, Zhiyu Hu, Yimeng Bai, Jiancan Wu, Qifan Wang, and Fuli Feng. 2024. Recommendation unlearning via influence function. *ACM Transactions on Recommender Systems* 3, 2 (2024), 1–23.

[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.