Revisiting Item Promotion in GNN-based Collaborative Filtering: A Masked Targeted Topological Attack Perspective

Yongwei Wang Yong Liu Zhiqi Shen
Nanyang Technological University, Singapore
{yongwei.wang,stephenliu,zqshen}@ntu.edu.sg

ABSTRACT

Graph neural networks (GNN) based collaborative filtering (CF) have attracted increasing attention in e-commerce and social media platforms. However, there still lack efforts to evaluate the robustness of such CF systems in deployment. Fundamentally different from existing attacks, this work revisits the item promotion task and reformulates it from a targeted topological attack perspective for the first time. Specifically, we first develop a targeted attack formulation to maximally increase a target item's popularity. We then leverage gradient-based optimizations to find a solution. However, we observe the gradient estimates often appear noisy due to the discrete nature of a graph, which leads to a degradation of attack ability. To resolve noisy gradient effects, we then propose a masked attack objective that can remarkably enhance the topological attack ability. Furthermore, we design a computationally efficient approach to the proposed attack, thus making it feasible to evaluate large-large CF systems. Experiments on two real-world datasets show the effectiveness of our attack in analyzing the robustness of GNN-based CF more practically.

KEYWORDS

Item promotion attack, collaborative filtering, recommendation system, targeted topology attack, node masking

1 INTRODUCTION

Collaborative filtering-based recommendation systems (RS) aim to recommend a personalized list of top-K products (*a.k.a* items) to each user that match best with her/his interests [4, 5]. Due to the effectiveness in promoting items, RS have been widely adopted in popular platforms, ranging from short-video discovery (*e.g.,* TikTok) to e-shopping (*e.g.,* Amazon). The mainstream paradigm of RS is collaborative filtering (CF), which assumes that users with similar behaviors are likely to show interests to similar items [11]. As a result, CF attempts to exploit the observed user-item interactions, modeled as a user-item matrix (*a.k.a* a bipartite graph), to make predictions for the unobserved ones. To better capture such interactions, graph neural networks (GNN) [14] have attracted increasing attention in RS, and GNN-based RS achieve state-of-the-art performances in recommendation [9, 23]. Therefore, this work focuses on the GNN-based RS.

Instead of trying to improve a recommender's prediction accuracy, this work investigates how to maximally boost the ranking of a low-popularity item on a potentially deployed recommendation system [19]. An item attains a higher popularity than another one if it is displayed in a larger number of users' recommendation list. This task finds favorable applications in scenarios where a seller expects to maximize the profits by promoting her/his items to as many potential users as possible. An intuitive solution is to encourage a number of users to show preference (*i.e.,* adding positive rating) to the target item, *e.g.,* by sending vouchers. However, there exist two crucial challenges to make the solution valid. The first challenge is how to ensure the newly-added ratings do contribute to the target item's popularity; and the other one is how to minimize the seller's budget (*e.g.,* vouchers) by limiting the number of ratings to be added.

The item promotion scenario is closely related to the robustness of a collaborative filtering recommendation system. Existing works attempt to address the challenges above by creating and injecting numerous fake users into the data, a technique known as shilling attacks or data poisoning attacks [6, 18, 22, 25]. However, these existing methods were generally specially designed for matrix factorization-based collaborative filtering recommenders, a type of conventional RS. Thus they are inapplicable to evaluating the robustness of an advanced GNN-based collaborative filtering RS. To our best knowledge, only limited studies propose data poisoning methods that may apply for GNN-based RS [22, 25].

Unfortunately, these recently proposed methods still demand adding a large number of fake users/ratings. Besides, due to the statistical differences in rating between real and fake users, fake users may be detected and removed to mitigate the attack ability. These issues hinders attacks to take place in real scenes. Therefore, it is of urgency to develop practical and effective attacks to evaluate GNN-based collaborative filtering models in real scenes.

For the first time, this work proposes a simple yet effective item promotion method on GNN-based RS from a masked topological attack perspective. The developed objective function allows us to maximize a target item's popularity with only a small number of interaction changes in the user-item graph topology. Yet it is challenging to solve the optimization problem mainly due to its combinatorial nature. To effectively address this issue, we employ a gradient-based solution and then propose a node masking mechanism to significantly enhance the attack ability. Moreover, we present a resource-efficient approach to enable our method to evaluate the robustness of large-scale GNN-based collaborative filtering systems.

Our major contributions can be summarized as follows:

Conference'17, July 2017, Washington, DC, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220078

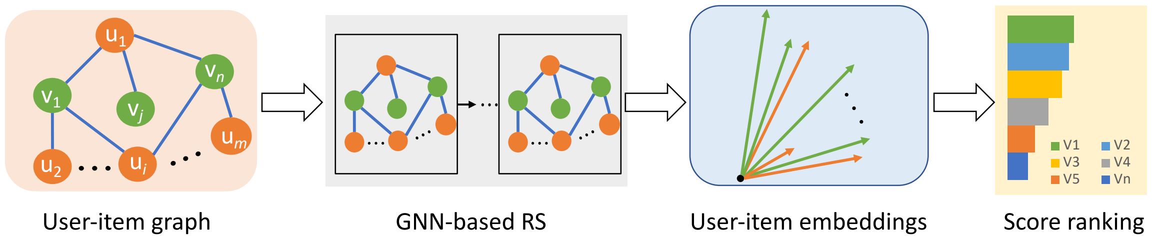


Figure 1: Illustration of an advanced GNN-based collaborative filtering model in recommender system. Users and items consist of a bipartite graph which is then input to a GNN-based collaborative filtering RS to generate user and item embeddings. Items that match best with a user in the embedding space will appear in the user's recommendation list.

- This work revisits the item promotion task in a GNN-based collaborative filtering system, and re-formulates it as a targeted topological attack problem for the first time. This new formulation is fundamentally different from existing mainstream item promotion attacks in which we do not create and inject fake user profiles into the system.
- We develop a novel node masking mechanism that can remarkably boost the attack ability of vanilla gradient-based optimization solutions. To address the memory consumption issue in large-scale graphs in real deployment, we further propose a resource-efficient approach that significantly reduces the memory cost from the quadratic to a linear one regarding the number of nodes.
- We conduct experiments on real-world recommendation datasets with advanced GNN-based collaborative filtering models. Our results reveal that our proposed methods can substantially promote an item's popularity even given limited perturbation budgets, and it is demonstrated to consistently outperform baseline attack methods.

2 RELATED WORK

2.1 GNN-based Collaborative Filtering

Collaborative filtering is a mainstream research in recommendation systems to predict users' preferences given collaborative signals. The essence of collaborative filtering is to learn user and item representations (*a.k.a* embeddings) jointly by leveraging the user-item interaction graph [27]. Then, items will be recommended to a user whose embeddings match the best with the user's embedding. Early explorations in collaborative filtering mainly focus on the matrix-factorization (MF) model [12, 16] and its variants that encode the interaction history [10, 15]. However, these methods only utilize a user's one-hop neighbors to generate the embeddings.

Inspired by the recent progress in GNN studies that exploit multi-hop neighbors in node embedding, GNN-based collaborative filtering methods have been proposed and achieved the state-of-the-art performances. Wang et al. [23] proposed neural graph collaborative filtering (NGCF), a new collaborative filtering framework based on graph neural networks to capture the higher-order connectivity of user/item nodes. More recently, He et al. proposed LightGCN [9] to simplify and improve NGCF. Specifically, LightGCN removed the use of feature transformation and nonlinear activation in network design, since these two components were observed to have negative effects on model training. To supplement supervised learning, self-supervised graph learning (SGL) [26] explored self-supervised learning and achieved the sate-of-the-art performance in the context of collaborative filtering to assist learning node and item representations.

2.2 Promoting Items in Collaborative Filtering

Although user-item interactions can effectively assist collaborative filtering, some of them may be intentionally falsified to mislead the recommender system. In the collaborative filtering regime, a most common threat is the item promotion attack [6, 18, 22, 24], in which an attacker aims to influence a specific item recommendation list of users. More concretely, the attacker may be an incentive-driven item owner and craves to increase the chance of their own items to be recommended by a victim collaborative model.

Many existing item promotion attacks can be broadly classified into two categories: model-agnostic attacks and model-specific attacks. Model-agnostic attacks (*e.g.,* RandFilter attack [17], average attack [17]) do not assume knowledge of victim collaborative models, therefore they can apply to both conventional collaborative filtering models and GNN-based ones. In contrast, model-specific attacks design attacking strategies only applicable for certain types of collaborative filtering models. For example, Li et al. [18] formulated an integrity attack objective for MF-based models [1, 13], then solved the attack problem using gradient-based optimization methods. Fang et al. [6] proposed to utilize the influence function to select and craft fake users for top-*K* MF-based recommenders. Tang et al. observed that many model-specific attacks lacked exactness in gradient computation, then proposed a more precise solution to improve the attack ability. Wu et al. designed a neural network-instantiated influence module and incorporated it into a triple generative adversarial network to craft fake users to launch attacks.

Unfortunately, the model-agnostic attack methods were specially designed for MF-based models, thus they are not applicable to promoting items in GNN-based collaborative filtering RS. Meanwhile, recent studies show that graph neural networks can be vulnerable to adversarial attacks — some unnoticeable feature or edge perturbations may significantly reduce the performance of a GNN model [3, 7, 29, 33]. Intuitively, adversarial attacks can be leveraged to promote items in GNN-based recommendation models. However, these existing attacks focus on GNN-based classifiers, leaving the vulnerability of GNN-based collaborative filtering largely unexplored.

Indeed, there are three major differences between a GNN-based classification model [28] and a GNN-based collaborative filtering model [27]. First, a classification decision can be made based on the logit of one node only, while a recommendation list returned by a GNN recommender involves ranking the prediction scores of all item nodes [9, 23]. Therefore, unlike fooling one node only in a GNN-classifier attack, attacking a GNN recommender requires to manipulating predictions of multiple nodes simultaneously. Thus, it makes the latter case special and more challenging. Second, a node classification model consists of both edges and associative semantic features, and manipulating features can effectively enhance the attack ability [33]. By contrast, a GNN recommender usually only contains user-item interactions, thus increasing the attack difficulty. Third, input graph formats and their scales are also different. The input to the former model is often a small symmetric graph, while there often includes a large bipartite graph (e.g., tens of thousands of user and item nodes) in the latter one [9, 26]. Therefore, memory-efficient attacks remains to be developed.

3 METHODOLOGY

3.1 Preliminaries

This study focuses on the LightGCN architecture [9], a state-of-the-art backbone in GNN-based collaborative filtering models [26, 31, 32].

Let $\mathcal{U} = \{u_1, \cdots, u_M\}$ and $\mathcal{I} = \{i_1, \cdots, i_N\}$ denote the set of M users and N items, respectively. Let $\mathcal{O}^+ = \{r_{ui} | u \in \mathcal{U}, i \in \mathcal{I}\}$ denote the interaction feedback of user u for item i. Here we consider an implicit feedback as in many real recommenders [22], i.e., $r_{ui} \in \{0, 1\}$, where 1 indicates a positive recommendation and 0 means an unknown entry. Denote the user-item rating binary matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ with entries as r_{ui} $(u = 1, \cdots, M;\ i = 1, \cdots, N)$. Then, we can construct a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ and $\mathcal{E} = \mathcal{O}^+$ represent the vertex (or node) set and edge set, respectively.

GNN-based collaborative filtering leverages the user-item graph $\mathcal{G}$ to learn embeddings. To be specific, it performs neighborhood aggregations iteratively on $\mathcal{G}$ to update a node's representation [9, 26]. The propagation rule for the l−th $(l = 1, \cdots, L)$ layer can be formally defined as,

$$\begin{aligned} \mathbf{z}_u^{(l)} &= g\Big(\sum_{j \in \mathcal{N}_u} \widetilde{\mathbf{R}}_{u,j} \cdot \mathbf{z}_j^{(l-1)} \Big) \\ \mathbf{z}_i^{(l)} &= g\Big(\sum_{j' \in \mathcal{N}_i} \widetilde{\mathbf{R}}^T \cdot \mathbf{z}_{j'}^{(l-1)} \Big) \end{aligned} \tag{1}$$

where $\mathbf{z}_u^{(l)} \in \mathbb{R}^d$ denotes the feature vector of user u at layer l, $\mathbf{z}_u^{(0)} = \mathbf{w}_u \in \mathbb{R}^d$ denotes the trainable and randomly initialized feature vector for user u, $g(\cdot)$ is an activation function which is often set as an identity function in recent works, $\mathcal{N}_u = \{j | (u, j) \in \mathcal{E}\}$ represents items within the neighborhood of a user u, and $\widetilde{\mathbf{R}}_{u,j}$ denotes the (u, j)-th entry of a normalized user-item rating matrix $\widetilde{\mathbf{R}}$, i.e., $\widetilde{\mathbf{R}} = \Lambda_L^{-1/2}\ \mathbf{R}\ \Lambda_R^{-1/2}$. Here $\Lambda_L \in \mathbb{R}^{M \times M}$ is a diagonal matrix with (u, u)-th entry as the degree of user u, $\Lambda_R \in \mathbb{R}^{N \times N}$ denotes a diagonal matrix with (i, i)-th entry as the degree of item i. Similarly, we have notations for item i's propagation rule by changing the subscript from u to i.

After obtaining representations of L layers, the embedding of a user (or item) node can be constructed by combining the representations computed at each layer,

$$\mathbf{z}_u = f_{comb}(\mathbf{z}_u^{(l)}),\ \mathbf{z}_i = f_{comb}(\mathbf{z}_i^{(l)}),\ \forall l \in [L] \tag{2}$$

where $f_{comb}(\cdot)$ denotes a representation combination function that may adopt representations from the final layer only, or utilize concatenation or weighted sum of representations from different layers [9, 23, 26].

In a GNN-based collaborative filtering, a typical way to obtain a recommendation prediction is by matching the embedding of a user with that of an item,

$$\hat{r}_{u,i} = \langle \mathbf{z}_u,\ \mathbf{z}_i \rangle \tag{3}$$

where $< \cdot >$ denotes an inner product, $\hat{r}_{u,i}$ is a rating score estimate that indicates how likely a user u would select an item i. The model training can be framed into a supervised learning setting or a self-supervised learning paradigm. In deployment, a pretrained collaborative filtering model first predicts rating scores for each user, then it ranks and recommends items with top-K highest scores to a user.

3.2 Targeted Topological Attacks

In a deployed recommender, a malicious item owner intends to promote a target item t to as many users as possible, a scenario called item promotion attack. Different from existing works that attempt to craft and inject fake users to graph $\mathcal{G}$, this work formulates it from a targeted topological attack perspective. We assume the attacker (*i.e.,* malicious item owner) has a white-box access to $\mathcal{G}$. We also assume that the attacker has a capability to persuade a few users to give high positive ratings to the target item (*e.g.,* sending vouchers). The attacking process can be formally defined as,

$$\begin{aligned} &\max_{\mathbf{R}^{atk}} \mathcal{L}_{atk}\Big(f_\theta(\mathbf{R}^{atk})_t\Big) \\ &\quad \text{s.t. } ||\mathbf{R}^{atk} - \mathbf{R}||_0 \le \Delta, \\ &\quad \mathbf{R}^{atk} \in \{0, 1\}^{M \times N} \end{aligned} \tag{4}$$

where $\mathcal{L}_{atk}$ denotes an objective function to improve the ranking of a specific item t, f_θ denotes a GNN-based collaborative filtering model parameterized by θ, $\mathbf{R}^{atk}$ denotes a manipulated user-item rating matrix by an attacker, $|| \cdot ||_0$ is an ℓ_0 norm, and Δ represents a perturbation budget for the attack, *i.e.,* the maximum number of user-item interactions allowed to manipulate.

For an arbitrary user $u \in \mathcal{U}$, denote the recommendation prediction scores for each item $i \in \mathcal{I}$ as $\mathbf{s}_u = [\hat{r}_{u,1}, \cdots, \hat{r}_{u,N}]$. The collaborative filtering system then ranks all entries in $\mathbf{s}_u$ and selects top-K items and recommend them to user u, denoted as $\Omega_K^u = [i_1^u, \cdots, i_K^u]$. Often, the target item t does not lie in the recommendation set Ω_K^u, thus requiring to be promoted into the set with a minimal perturbation budget.

To achieve the item promotion purpose, we formulate an objective function as,

$$\mathcal{L}_{atk} = \frac{1}{M} \sum_{u \in \mathcal{U}} \Big[\lambda \log\sigma(\hat{r}_{u,t}) - (1 - \lambda) \sum_{j \in \Omega_K^u, j \ne t} \log\sigma(\hat{r}_{u,j}) \Big] \tag{5}$$

where λ is a hyperparameter to balance score penalizations, $\sigma(\cdot)$ denotes a sigmoid activation function $\sigma(x) = 1/\big(1 + \exp(-x)\big)$ that converts predicted score values to the $(0, 1)$ interval.

By substituting Eq. (5) into Eq. (4), we obtain a constraint optimization problem in the white-box targeted toplogical attack setting. Unfortunately, this is a combinatorial problem and finding an exact solution is NP-hard in computational complexity. Alternatively, similar to white-box adversarial attacks on images, we can leverage the gradient-based optimization to approximate the solution [8, 20].

First, we relax a discrete $\boldsymbol{R}$ as a continuous multivariable. We then compute its saliency map based on the gradients of $\boldsymbol{R}^{atk}$ with respect to each variable. The saliency map measures the contributions of every pair of user-item interactions to maximize the attack objective function in Eq. (5). To satisfy the perturbation budget in Eq. (4), we select Δ users that have highest intensity in the saliency map, and do a gradient ascent to update $\boldsymbol{R}$. Specifically, the topological attack can be expressed as,

$$\boldsymbol{R}^{atk} = \mathcal{P}\Big(\boldsymbol{R} + \boldsymbol{M} \odot \mathrm{sign}\big(\nabla_{\boldsymbol{R}} \mathcal{L}_{atk}\big)\Big) \tag{6}$$

where $\mathcal{P}$ is a projection operator that clips the perturbed $\boldsymbol{R}$ back to the $\{0, 1\}^{M\times N}$ space, $\odot$ denotes an element-wise product, $\mathrm{sign}(\cdot)$ is a sign function, $\boldsymbol{M} \in \{0, 1\}^{M\times N}$ denotes a binary mask that can be computed based on the gradient saliency map,

$$\boldsymbol{M}_{u,i} = \begin{cases} 1, & if\ \Big([\nabla_{\boldsymbol{R}} \mathcal{L}_{atk}]_{u,i} > 0\Big) \cap \Big((u, i) \in \Omega_g\Big) \\ 0, & otherwise \end{cases} \tag{7}$$

where Ω_g is an index set that contains the top-Δ largest values of $\nabla_{\boldsymbol{R}} \mathcal{L}_{atk}$ and it can be formally defined as,

$$\underset{\Omega_g \subset \mathcal{G}, |\Omega_g| = \Delta}{\arg\max} \sum_{(u,i) \in \Omega_g} \nabla_{\boldsymbol{R}_{u,i}} \mathcal{L}_{atk} \tag{8}$$

A physical interpretation to the binary mask $\boldsymbol{M}$ is how new user-item interactions should be established in order to maximally promote a target item.

3.3 Node Masking Mechanism

As described in the previous section, we utilize a gradient-based optimization approach to approximate the optimal solution to Eq. (4). Given a limited perturbation budget, we select and create user-item pair candidates that achieve highest responses in the gradient saliency map. However, the attack ability can be further improved due to potential issues in this approach. First, the gradient estimates can be noisy due to the discrete nature of variables in $\boldsymbol{R}$. Also, the quantization errors due to the utilization of the sign function may hamper the effectiveness of gradient ascent.

Notice that the derivative $\nabla_{\boldsymbol{R}_{u,i}} \mathcal{L}_{atk}$ in Eq. (6) is a summation of individual derivatives computed from the log scores of M user nodes w.r.t. the binary variable $\boldsymbol{R}_{u,i}$. To negate noisy effects in gradient estimates, an intuitive way is to adopt a subset of user nodes by masking out unimportant ones. We prefer preserving nodes with high predicted scores $\hat{r}_{u,t}$ than those with lower ones $\hat{r}_{u',t}$ for the target item t. This is because item t is more likely enter into the top-K recommendation list of the user u than user u' after a single step gradient ascent.

We design a pre-filtering step for the node masking mechanism based on predicted scores from the GNN-based collaborative filtering system. Specifically, we compose a user subset $\mathcal{U}' \subset \mathcal{U}$ that satisfies,

$$\mathcal{U}' = \Big\{u \mid u \in \mathcal{U},\ \sigma(\hat{r}_{u',t}) \geq \gamma\Big\} \tag{9}$$

where γ denotes a masking threshold parameter. Then, a masked objective function $\mathcal{L}^m_{atk}$ can be expressed as,

$$\mathcal{L}^m_{atk} = \frac{1}{|\mathcal{U}'|} \sum_{u \in \mathcal{U}'} \Big[\lambda \log\sigma(\hat{r}_{u,t}) - (1 - \lambda) \sum_{j \in \Omega^u_K, j \neq t} \log\sigma(\hat{r}_{u,j})\Big] \tag{10}$$

Clearly, Eq. (5) is a special case of Eq. (10) by setting γ to be 0. It is worth noting that our node masking mechanism is clearly different from masked strategy used in [7]. First of all, the research tasks are different: our work targets an item promotion task in a collaborative filtering system that involves ranking, while work [7] deals with a classification task. Second, we rank predicted scores and mask out user nodes with low prediction confidence below a threshold, while work [7] necessitates a comparison with the groundtruth labels of nodes and removes the incorrectly classified ones. Moreover, the objective functions are fundamentally different because of different tasks.

3.4 Scaling to Large-scale Graphs

The gradient ascent-based solution in Eq. (6) requires to compute gradients w.r.t. each entry in $\boldsymbol{R}$. This approach works well on a graphics processing unit (GPU) card with limited memory when the input user-item interaction matrix $\boldsymbol{R}$ is of relatively small size. In real deployment scenes, however, with a dense gradient $\nabla_{\boldsymbol{R}} \mathcal{L}^m_{atk}$, computational issues can arise if it involves a large-scale graph that consists of thousands even millions of user and item nodes.

Proposition 1. *In a one-layer GNN-based collaborative filtering model defined in Eq. (1), the partial derivatives satisfy* $\nabla_{\boldsymbol{R}_{u,t}} \mathcal{L}^m_{atk} > \nabla_{\boldsymbol{R}_{u,j}} \mathcal{L}^m_{atk}, (j \neq t)$ *if* $< \mathbf{z}^{(0)}_u, \mathbf{z}^{(0)}_i > \to 1$ *for* $u = 1, \cdots, M, i = 1, \cdots, N$.

Remark. The analysis above indicates that there only necessitates to compute gradients with respect the target item t, i.e.,, $\nabla_{\boldsymbol{R}_{u,t}} (u = 1, \cdots, M)$ in Eq. (6). Empirically, we observe that $\nabla_{\boldsymbol{R}_{u,t}} \mathcal{L}^m_{atk} > \nabla_{\boldsymbol{R}_{u,j}} \mathcal{L}^m_{atk}, (j \neq t)$ also holds for the multi-layer well trained GNN-based collaborative filtering models. In this way, the memory consumption can be reduced from $\mathcal{S}(M \times N)$ to $\mathcal{S}(M)$, which is a significant cost reduction especially when N is a large value.

In implementation (*e.g.,* using PyTorch [21]), we can split matrix $\boldsymbol{R}$ into three separate tensors: $\boldsymbol{R} = \big[\boldsymbol{R}_{:,:t-1}, \boldsymbol{R}_{:,t}, \boldsymbol{R}_{:,t+1:N}\big]$, where only tensor $\boldsymbol{R}_{:,t}$ requires a gradient computation. Then we do a regular forward process to compute the targeted loss as in Eq. (10), and then backpropagate gradients to tensor $\boldsymbol{R}_{:,t}$.

The algorithm of the proposed method is presented in Algorithm 1.

4 EXPERIMENTS

In this section, we demonstrate the effectiveness of our item promotion attack method in GNN-based collaborative filtering systems. We first introduce the experimental setup, then conduct experiments

Algorithm 1: The proposed scalable algorithm for masked targeted attacks on GNN-based collaborative filtering models.

```
Data: A pretrained $f_\theta$ that consists of $\mathbf{w}_u$ and $\mathbf{w}_i$, data $\boldsymbol{R} \in \mathbb{R}^{M \times N}$, target
      item $t$, perturbation budget $\Delta$, parameter $\lambda$, masking threshold $\gamma$.
Result: A perturbed $\boldsymbol{R}^{atk}$ that satisfies Eq. (8).
// Initialization and forward
1  Initialize embeddings $\mathbf{z}_u^{(0)} = \mathbf{w}_u, \mathbf{z}_i^{(0)} = \mathbf{w}_i$, set $l = 1$ ;
2  Rewrite $\boldsymbol{R}$: $\boldsymbol{R} \leftarrow \left[\boldsymbol{R}_{:,:t-1}, \boldsymbol{R}_{:,t}, \boldsymbol{R}_{:,t+1:N}\right]$;
3  Normalize $\boldsymbol{R}$: $\widetilde{\mathrm{R}} \leftarrow \Lambda_L^{-1/2}\ \mathrm{R}\ \Lambda_R^{-1/2}$ ;
4  while $l \leq L$ do
5      Compute users embeddings at layer $l$:
       $\mathbf{z}_u^{(l)} \leftarrow g\Big(\boldsymbol{R}_{:,:t-1} \cdot \boldsymbol{z}_i^{l-1}[:t-1,:] + \boldsymbol{R}_{:,t} \cdot \boldsymbol{z}_i^{l-1}[t,:] + \boldsymbol{R}_{:,t:N} \cdot \boldsymbol{z}_i^{l-1}[t:N,:]\Big)$
       ;
6      Compute items embeddings at layer $l$:
       $\mathbf{z}_i^{(l)} \leftarrow \Big[g(\boldsymbol{R}_{:,t}^T \cdot \boldsymbol{z}_u^{l-1}); g(\boldsymbol{R}_{:,t}^T \cdot \boldsymbol{z}_u^{l-1}); g(\boldsymbol{R}_{:,t:N}^T \cdot \boldsymbol{z}_u^{l-1})\Big]$ ;
7      $l \leftarrow l + 1$;
8  end
9  Compute final user and item embeddings using Eq. (2);
// Backward for gradient computation
10 Compute masked targeted loss $\mathcal{L}_{atk}^m$ using Eq. (10) ;
11 Compute $\nabla_{\boldsymbol{R}_{:,t}} \mathcal{L}_{atk}^m$ using autograd, and set the rest gradients in $\nabla_{\boldsymbol{R}}$ as 0 ;
12 Find top-$\Delta$ largest values in $\nabla_{\boldsymbol{R}}$, and compute binary mask $\boldsymbol{M}$ using Eq. (7);
13 Compute $\boldsymbol{R}^{atk}$ using gradient ascent in Eq. (6);
14 Return: The perturbed $\boldsymbol{R}^{atk}$.
```

on two real-world datasets for empirical validation under different settings.

4.1 Experimental Setup

Datasets: We conduct experiments on Gowalla [2] and Yelp2018 [30], two commonly-used datasets for recommendation [9, 23]. For both datasets, we use the pre-processed dataset with train/test split following work [9]. Gowalla contains 29,858 users and 40,981 items, with an overall number of user-item interactions as 1,027,370. Yelp2018 includes 31,667 users and 38,047 items, and has 1,561,406 user-item interactions in total.

Models: We evaluate our method on the LightGCN and variant models, the state-of-the-art GNN-based collaborative filtering recommenders. LightGCN is trained on Gowalla and Yelp2018 datasets, respectively, with PyTorch implementations officially released by [9]. We adopt default hyperparameters as shown in the official implementation. After sufficient training, LightGCN achieves good performances on both datasets. The recommendation performances on the clean datasets are reported in Appendix.

Evaluation Protocols: We demonstrate the attack ability to promote a target item on low popular items on Gowalla [2] and Yelp2018 datasets. For a well trained collaborative filtering model, an item with fewer positive ratings in $\boldsymbol{R}$ will be less popular than another that has more positive feedbacks. Therefore, we use the degree of an item to quantify the popularity. To be specific, we compose three low popular target item sets based on an item's original degree. The percentile of the three item sets are: Q_{10}, Q_{30}, Q_{50}, respectively. For each item from the three item sets, two perturbation budgets are defined: $\Delta_s^1 = deg(Q_{65}) - deg(Q_s)$ and $\Delta_s^2 = \bar{d} - deg(Q_s)$, where $\bar{d}$ is the mean degree, $deg(q)$ denotes the degree of an item that lies in a percentile q, and $s \in \{10, 30, 50\}$. To better reflect the trend of item promotion improvements, we also adopt a continually varying

Dataset	Δ_{10}^1	Δ_{10}^2	Δ_{30}^1	Δ_{30}^2	Δ_{50}^1	Δ_{50}^2
Gowalla	7	12	5	10	3	8
Yelp2018	17	25	13	21	8	16

Table 1: Perturbation budgets for topological attacks.

number of perturbation budgets. The perturbation budgets are shown in Table 1.

For the quantitative measure, we utilize hit number (HN) to evaluate the item promotion ability. For a specific target item, $HN@50$ is defined as the frequency that users have this item to be displayed in their top-50 recommendation list. To be more accurate, we define a pruned hit number ($PHN@50$) metric that removes the number of newly-added users from the $HN@50$ metric. For reproducibility, we report the averaged $HN@10$ and $PHN@K$ over 30 number of randomly selected target items from three low popular item sets individually. All reported results utilize fixed hyperparameters $\lambda = 0.5$, $\gamma = 0.95$.

Comparison Methods: We utilize and modify existing model-agnostic attacks on collaborative filtering models and use them as our baseline methods (e.g., random attack [17]). Please note that we cannot compare with recent model-specific attacks (*e.g.,* [6, 18, 22]), because they were developed for MF-based CF, and they do not apply for attacking GNN-based CF models. Besides, our considered settings are dramatically different from model-specific attacks in that these methods require injecting a set of fake users into training data, while our method focuses on selecting and modifying a small number of existing users. Besides RandFilter, we also design two other heuristic attacks as our baseline attacks. The compared methods are:

- **RandFilter**: RandFilter was originally used for explicit ratings [17], and we modify it for implicit rating. We randomly select Δ users and asked them to give positive ratings to the target item t.
- **IUFilter**: From the users perspective, users that have frequent purchasing history tend to be more influential in positive rating. Therefore we choose top-Δ such users and let them to rate positively to the target items.
- **RUFilter**: RUFilter selects users that have top-Δ predicted rating scores for item t and put corresponding entries in $\boldsymbol{R}$ as 1 in the implicit recommendation setting.

4.2 Promoting Items in White-box Scenes

An attacker is first assumed to have a white-box access to the pre-trained GNN-based CF model. The attacker can use three baseline attacks and the proposed attack (in Algorithm 1) to conduct item promotion attacks. Three sets of low popularity items (i.e., Q_{10}, Q_{30}, Q_{50}) will be evaluated with different perturbation budgets. The comparison results are reported in Table 2.

From Table 2, we can observe that our proposed method achieves the highest averaged $PHN@50$ for all settings, substantially outperforming baseline methods. For example, when target items are from Q_{10} and a perturbation budget as Δ_{10}^1 on Gowalla, the $PHN@50$ values are 0.7, 0.5, 9.1 for RandFilter, IUFilter and RUFilter, respectively; while our method achieves a $PHN@50$ as $\mathbf{41.4}$, which is $\mathbf{4.5\times}$ larger than the second best method. The superiority of our method is

Dataset	Attack	Low popularity items					
		Q_{10}		Q_{30}		Q_{50}	
		$PHN@50$ (Δ_{10}^1)	$PHN@50$ (Δ_{10}^2)	$PHN@50$ (Δ_{30}^1)	$PHN@50$ (Δ_{30}^2)	$PHN@50$ (Δ_{50}^1)	$PHN@50$ (Δ_{50}^2)
Gowalla	RandFilter	0.7	0.5	2.2	0.8	3.9	1.8
	IUFilter	0.5	0.3	1.9	0.7	4.5	1.5
	RUFilter	9.1	15.6	14.2	25.7	17.1	29.0
	Proposed	**41.4**	**117.6**	**39.1**	**106.3**	**28.9**	**87.7**
Yelp2018	RandFilter	0	0	0.2	0.1	1.2	0.4
	IUFilter	0	0	0.2	0.1	0.8	0.2
	RUFilter	7.4	19.1	4.8	10.5	7.6	16.5
	Proposed	**60.6**	**140.8**	**32.6**	**106.7**	**20.7**	**75.0**

Table 2: Performance comparisons of different attacks in improving a target item's popularity on Gowalla and Yelp2018 datasets. Three low popularity item sets (Q_{10}, Q_{30}, Q_{50}) are used for performance evaluation with perturbation budgets as Δ_s^1 and $\Delta_s^2 (s = 10, 30, 50)$. $PHN@50$ is averaged over 30 randomly selected target items at each item set. The best performances are marked in bold.

even more prominent for target items from Q_{10} with the perturbation budget as Δ_{10}^2, *i.e.,* 7.5× stronger than RUFilter. Although the item promotion ability tends to decrease from Q_{10} to Q_{50}, the performance of our method is still significantly better than all baseline methods. We can arrive at a same conclusion for experiments on Yelp2018.

In addition, we vary the perturbation budgets gradually and show in Fig. (2) the comparison results. Fig. (2) reveals that as the perturbation budgets increase, the promotion ability of our method increase dramatically, and the performance gap becomes larger compared to baseline methods. This observation indicates that GNN-based CF model is vulnerable to the proposed masked topological attack, particularly with a relatively large adversarial perturbation budget.

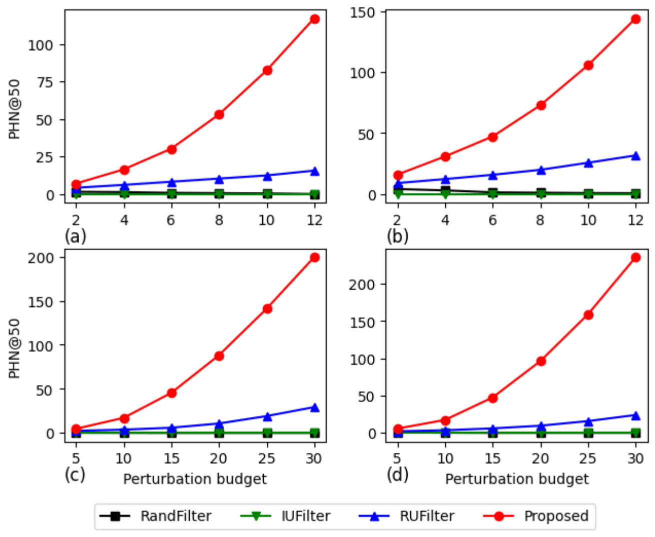


Figure 2: Performance comparisons with a gradual varying number of budgets on low-popular items from Gowalla and Yelp2018 datasets. (a) and (b) display $PHN@50$ results for target items from Gowalla with Q_{10} and Q_{30}, and (c) and (d) show $PHN@50$ from Yelp2018 with Q_{10} and Q_{30}, respectively.

4.3 Promoting Items in Black-box Scenes

In addition to white-box attacks, we study the effectiveness of our method in a black-box setting, in which an attacker is assumed to have no knowledge to the victim models. In this setting, an attacker first adopts the pertrained model as a substitute model (sub. model) and creates a perturbed graph for a target item. The attacker then attempts to promote the target item on a unknown collaborative filtering model. Based on [9], we obtain three victim models by setting different number of layers and the length of embeddings. Please refer to Appendix for detailed setup.

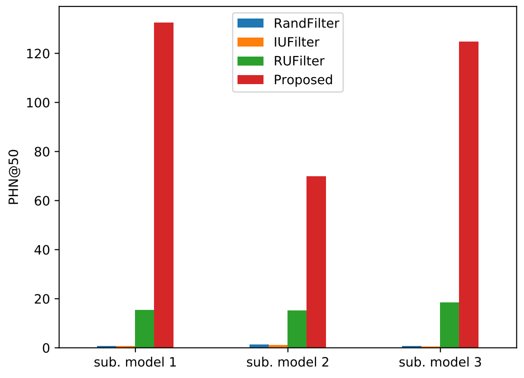


Figure 3: Visualization of attack performance comparisons on three different substitute models on Gowalla.

In Fig. 3, we compare and visualize the attack performance of different methods on three substitute models (i.e., sub. models 1–3) on Gowalla. Although three substitute models are different from the source model, clearly, we can conclude that the proposed method still achieves satisfactory attack performances.

4.4 Effectiveness of Node Masking Mechanism

This section evaluates the performance of our method with different choices of parameter γ. Specifically, we vary γ from 0.05 to 0.95 and compare $PHN@50$. Figure 4 displays the performance curve using target items from Q_{10} item set.

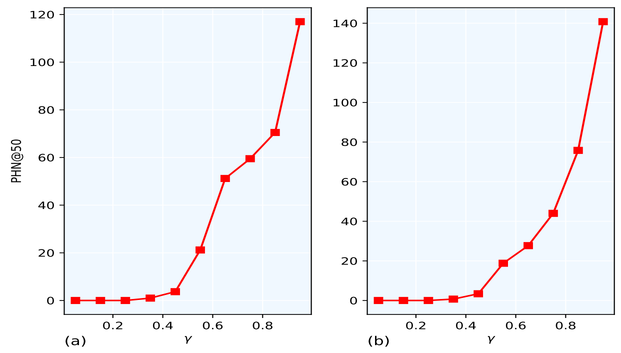


Figure 4: Performance variations versus masking thresholds γ. (a) and (b) show $PHN@50$ results vs γ for target items from Q_{10} on Gowalla and Yelp2018 datasets, respectively.

Dataset	Attack	Low popularity items		
		Q_{10}	Q_{30}	Q_{50}
		$PHN@50$ (Δ^1_{10})	$PHN@50$ (Δ^1_{30})	$PHN@50$ (Δ^1_{50})
Gowalla	RandFilter	8.8	12.4	14.9
	IUFilter	1.9	5.0	7.8
	RUFilter	5.2	11.3	13.6
	Proposed	**23.4**	**23.3**	**20.5**
Yelp2018	RandFilter	5.0	6.4	8.3
	IUFilter	1.2	1.7	3.9
	RUFilter	5.1	3.6	6.4
	Proposed	**23.1**	**16.7**	**13.0**

Table 3: Performance comparisons of different attacks in improving a target item's popularity on Gowalla and Yelp2018 datasets with retraining. Three low popularity item sets (Q_{10}, Q_{30}, Q_{50}) are used for performance evaluation with perturbation budgets as Δ^1_s ($s = 10, 30, 50$). $PHN@50$ is averaged over 30 randomly selected target items at each item set. The best performances are marked in bold.

In Figure 4, we can see that the item promotion ability of our method is low when we use all users to establish the attack objective (i.e., γ as 0). As we increase the masking threshold γ, $PHN@50$ increases remarkably. This observation confirms the effectiveness of the node masking mechanism.

4.5 Promoting Items in Retraining Scenes

In real deployment scenarios, a collaborative filtering model may be retrained from scratch to capture the dynamics of an input graph. We simulate such an item promotion scene by first perturbing a user-item graph followed by retraining a new model on the perturbed graph. We keep all experimental setting the same as that used for the source model.

Table 3 reveals that, compared with baseline methods, our method consistently achieves the highest $PHN@50$ with a same perturbation budget. On both datasets, we have about 1.4× to 4.6× larger attack ability than the second best method. Therefore, the proposed attack method successfully maintains a strong promotion ability even when we retrain the collaborative filtering model after topological perturbations. Please refer to the Appendix for more results.

5 CONCLUSION

In this work, we have proposed a novel view on promoting items in GNN-based collaborative filtering models based on topological attacks. Our formulation identifies users that play pivotal roles to promote a target item. We then propose a node masking mechanism to effectively improve vanilla gradient-based solutions. A resource-efficient approach is developed to make our method scalable to large-scale graphs. Experimental results demonstrate that the proposed method can significantly promote a target item. Our findings raise the security and adversarial robustness issues of GNN-based collaborative filtering models in practical scenes.

APPENDICES

A. Detailed Training Setups & Performances of The Main Model and Three Victim Models

For the main model evaluated in Section 4.2, we utilized the default parameter settings as in the official PyTorch implementation of work [9]. We also craft three victim models by setting different training hyparameters.

To be more specific, four hyperparameters are changed to obtain three different victim models, i.e., sub. model 1, sub. model 2 and sub. model 3. The hyperparameters are respectively as 1) the number of layers L; 2) the length of a user or item embedding d; 3) the number of training epochs (i.e., #epochs); and 4) the random seed used for model randomization. The hyperparameters of the main model and three other victim models are listed in Table 4.

	L	d	#epochs	seed
main model	3	64	1000	2020
sub. model 1	2	64	200	2022
sub. model 2	2	128	200	2022
sub. model 3	3	128	200	2022

Table 4: The training setups of the main victim model (for white-box attacks) and three different victim models (for black-box attacks).

The main victim model and three victim models adopt a same f_{comb} (defined in Eq. (2)) to combine user or item embeddings as for the main evaluation model, following existing studies [9, 26]. Namely, the final embedding is formed by a weighted sum of embeddings output from each layer,

$$\mathbf{z}_u = \sum_{l=0}^{L} \alpha_l \, e_u^{(l)}, \quad \mathbf{z}_i = \sum_{l=0}^{L} \alpha_l \, e_i^{(l)} \tag{11}$$

where α_l is set uniformly as $1/(L+1)$.

After sufficient training of the main model and three victim models, we report in Table 5 their recommendation performances on clean Gowalla and Yelp2018 datasets. Similar as the main model, all three victim models can achieve satisfactory recommendation performances, indicating that they may also be deployed in real recommendation scenes. Therefore, we assume these three sub. models as victim models, and we evaluate the attacking performance of our method on them in a black-box attack setting.

	Model	*Precision*	*Recall*	*NDCG*
Gowalla	main model	0.056	0.182	0.154
	sub. model 1	0.053	0.173	0.148
	sub. model 2	0.055	0.180	0.153
	sub. model 3	0.055	0.181	0.153
Yelp2018	main model	0.028	0.063	0.052
	sub. model 1	0.026	0.582	0.048
	sub. model 2	0.028	0.062	0.051
	sub. model 3	0.029	0.064	0.053

Table 5: The recommendation performances of the pretrained main model and three victim models on clean Gowalla and Yelp2018 datasets.

	Model	*Precision*	*Recall*	*NDCG*
Gowalla	main model	0.056	0.182	0.154
	sub. model 1	0.052	0.173	0.148
	sub. model 2	0.055	0.180	0.153
	sub. model 3	0.055	0.181	0.153
Yelp2018	main model	0.028	0.063	0.052
	sub. model 1	0.026	0.580	0.048
	sub. model 2	0.028	0.062	0.051
	sub. model 3	0.029	0.064	0.053

Table 6: The recommendation performances of the pretrained main model and three victim models on perturbed graphs on Gowalla and Yelp2018 datasets. The perturbed graphs are generated using the proposed method with low popularity items Q_{10} and Δ as 12 and 25, respectively on Gowalla and Yelp2018 datasets.

B. Perturbed Graphs Impose Minimal Influences on Overall Recommendation Performances

In this section, we intends to evaluate whether perturbations on the user/item graphs can influence the overall recommendation performances of a GNN-based collaborative filtering model. On Gowalla and Yelp2018 datasets, the evaluated graphs are generated on Q_{10} items with the number of perturbation budgets set as 12 and 25, respectively.

The recommendation performances of four models are reported in Table 6. Compared with those in Table 5, the recommendation performances almost stay unchanged by using perturbed graphs generated by our method. This is because we are utilizing a very small perturbation budget compared to the (large) number of users in the recommendation system, thus leading to minimal influence to the overall recommendation performances. For example, on Gowalla, we perturb 12 users which only takes about 0.04% of all users. The minimal influences on overall recommendation performances imply that the model will perform almost the same to recommend all other items, which also indicates difficulties to detect the existence of adversarial perturbations by our method.

Dataset	Attack	Low popularity items		
		Q_{10}	Q_{30}	Q_{50}
		PHN@50 (Δ^2_{10})	*PHN@50* (Δ^2_{30})	*PHN@50* (Δ^2_{50})
Gowalla	RandFilter	11.6	11.2	14.3
	IUFilter	1.7	3.8	7.8
	RUFilter	9.6	15.9	19.0
	Proposed	**47.9**	**49.5**	**46.6**
Yelp2018	RandFilter	6.3	9.9	11.9
	IUFilter	1.2	2.0	3.7
	RUFilter	6.6	6.0	8.3
	Proposed	**48.5**	**35.2**	**25.8**

Table 7: Performance comparisons of different attacks in improving a target item's popularity on Gowalla and Yelp2018 datasets with retraining. Three low popularity item sets (Q_{10}, Q_{30}, Q_{50}) are used for performance evaluation with perturbation budgets as $\Delta^2_s (s = 10, 30, 50)$. $PHN@50$ is averaged over 30 randomly selected target items at each item set. The best performances are marked in bold.

C. More Results for Retraining Scenes

In Table 7, we present the item promotion results on Yelp2018 of low popularity items, i.e., Q_{10}, Q_{30}, Q_{50} with perturbations as $\Delta^1_{10}, \Delta^1_{30}, \Delta^1_{50}$ and $\Delta^2_{10}, \Delta^2_{30}, \Delta^2_{50}$ by retraining models on perturbed user and item graphs. Consistent with results reported for Gowalla in Table 3, the proposed method outperforms baseline methods by a large margin in terms of the $PHN@50$ metric.

In addition, we evaluate and compare the $PHN@50$ values of the retrained GNN-based collaborative filtering model for two sets of items: the first set of items are clean items (i.e., no perturbations), while the second set are target items with certain perturbations. For each set of items, we randomly selected 30 items and report their averaged $PHN@50$ values.

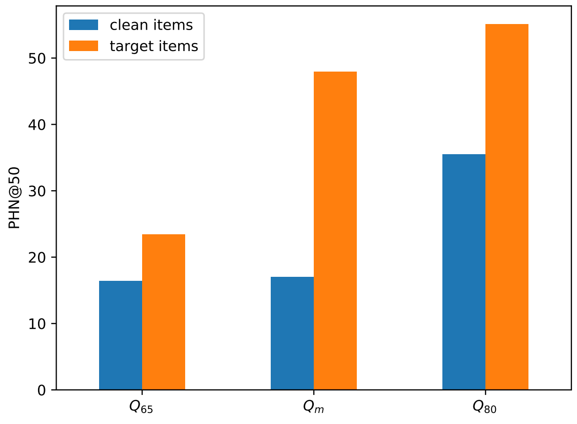
In Fig. 5, we report the comparison results on Gowalla and Yelp2018, where the target items are from the Q_{10} set. The clean items are from Q_{65}, Q_m, and Q_{80}, respectively. The perturbation budgets are computed by subtracting degrees of clean items from those of Q_{10} items. To be specific, as reported in Table 1, the degree differences are Δ^1_{10} and Δ^2_{10} to promote items from Q_{10} set to Q_{65}, Q_m sets; and the perturbation budgets are 14 and 33 to promote items from Q_{10} to Q_{80}.

From Fig. 5, we observe that target items display clearly higher $PHN@50$ values than their clean counterparts even with the same number of degrees. This phenomena reveals that the proposed attack has indeed identified potential users that will have significant impacts on target items regardless whether the collaborative filtering will be trained or not.

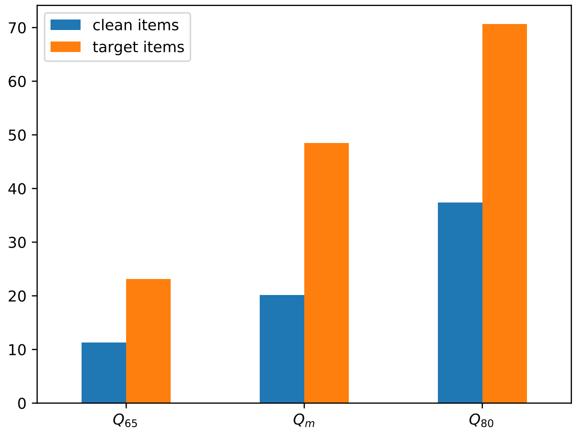
D. Black-box Attacks on Three Victim Models on Yelp2018

In Fig. 6, we show the attack performance comparisons of item promotion methods on three different victim GNN-based collaborative filtering models on Yelp2018 dataset.

For three substitute models, our attack method maintains high item promotion ability, while three baseline methods show little promotion ability on Yelp2018. This observation is consistent with that on Gowalla as visualized in Fig. 3.



(a) *PHN*@50 **comparisons on Gowalla**



(b) *PHN*@50 **comparisons on Yelp2018**

Figure 5: Performance comparisons between items from clean and promoted target item sets on Gowalla and Yelp2018 datasets.

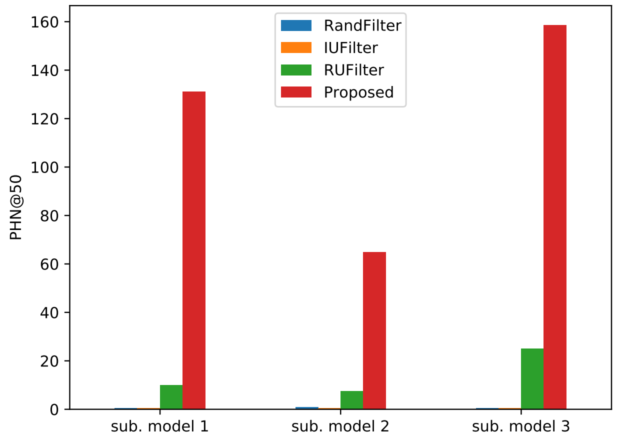


Figure 6: Visualization of attack performance comparisons on three different substitute models on Yelp2018.

E. Evaluation with Different *PHN@K* Metrics

In addition to utilizing the *PHN@K* with *K* set as 50 in previous sections, we also report *PHN*@10 and *PHN*@20 in Table 8. Indeed, we can see that the item promotion ability of our method can also outperform baseline methods substantially on *PHN*@10 and *PHN*@20 metrics.

Dataset	Attack	*PHN@10* (Δ^1_{10})	*PHN@20* (Δ^1_{10})
Gowalla	RandFilter	0	0
	IUFilter	0	0
	RUFilter	3.9	6.7
	Proposed	**18.3**	**41.3**
Yelp2018	RandFilter	0	0
	IUFilter	0	0
	RUFilter	4	6.5
	Proposed	**43.4**	**70.6**

Table 8: Evaluating performances using different *PHN@K* **metrics. The best performances are marked in bold.**

REFERENCES

[1] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization* 20, 4 (2010), 1956–1982.

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

[3] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[4] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.

[5] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction* 4, 2 (2011), 81–173.

[6] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*. 3019–3025.

[7] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems* 34 (2021), 7637–7649.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations* (2015).

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

[10] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.

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

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.

[13] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 665–674.

[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[15] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[17] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*. 393–402.

[18] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* 29 (2016).

[19] Zhuoran Liu and Martha Larson. 2021. Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start. In *Proceedings of the Web Conference 2021*. 3590–3602.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations* (2018).

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[22] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Fourteenth ACM conference on recommender systems*. 318–327.

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

[24] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, Enhong Chen, and Senchao Yuan. 2021. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1074–1083.

[25] Fan Wu, Min Gao, Junliang Yu, Zongwei Wang, Kecheng Liu, and Xu Wang. 2021. Ready for emerging threats to recommender systems? A graph convolution-based generative shilling attack. *Information Sciences* 578 (2021), 683–701.

[26] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[27] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[28] 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* 32, 1 (2020), 4–24.

[29] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214* (2019).

[30] Yelp Challenge Dataset. 2018. (2018). https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset.

[31] Lingzi Zhang, Yong Liu, Xin Zhou, Chunyan Miao, Guoxin Wang, and Haihong Tang. 2022. Diffusion-Based Graph Contrastive Learning for Recommendation with Implicit Feedback. In *International Conference on Database Systems for Advanced Applications*. Springer, 232–247.

[32] Xin Zhou, Aixin Sun, Yong Liu, Jie Zhang, and Chunyan Miao. 2021. SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. *arXiv preprint arXiv:2107.03019* (2021).

[33] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.