GRIFFITH UNIVERSITY

Handling Low Homophily in Recommender Systems with Partitioned Graph Transformer

Author

Nguyen, Thanh Tam, Nguyen, Thanh Toan, Weidlich, Matthias, Jo, Jun, Nguyen, Quoc Viet Hung, Yin, Hongzhi, Liew, Alan Wee-Chung

Published

2024

Journal Title

IEEE Transactions on Knowledge and Data Engineering

Version

Accepted Manuscript (AM)

DOI

10.1109/tkde.2024.3485880

Rights statement

This work is covered by copyright. You must assume that re-use is limited to personal use and that permission from the copyright owner must be obtained for all other uses. If the document is available under a specified licence, refer to the licence for details of permitted re-use. If you believe that this work infringes copyright please make a copyright takedown request using the form at https://www.griffith.edu.au/copyright-matters.

Downloaded from

https://hdl.handle.net/10072/433916

Funder(s)

ARC

Grant identifier(s)

DP240101108

Griffith Research Online

https://research-repository.griffith.edu.au

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

Handling Low Homophily in Recommender Systems with Partitioned Graph Transformer

Thanh Tam Nguyen, Thanh Toan Nguyen, Matthias Weidlich, Jun Jo, Quoc Viet Hung Nguyen, Hongzhi Yin, and Alan Wee-Chung Liew

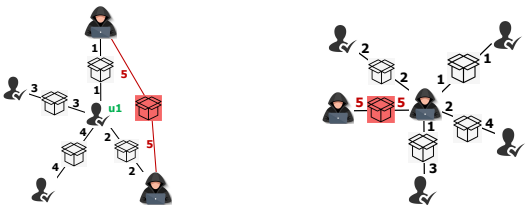
Abstract—Modern recommender systems derive predictions from an interaction graph that links users and items. To this end, many of today's state-of-the-art systems use graph neural networks (GNNs) to learn effective representations of these graphs under the assumption of homophily, i.e., the idea that similar users will sit close to each other in the graph. However, recent studies have revealed that real-world recommendation graphs are often heterophilous, i.e., dissimilar users will also often sit close to each other. One of the reasons for this heterophilia is shilling attacks that obscure the inherent characteristics of the graph and make the derived recommendations less accurate as a consequence. Hence, to cope with low homophily in recommender systems, we propose a recommendation model called PGT4Rec that is based on a Partitioned Graph Transformer. The model integrates label information into the learning process, which allows discriminative neighbourhoods of users to be generated. As such, the framework can both detect shilling attacks and predict user ratings for items. Extensive experiments on real and synthetic datasets show PGT4Rec as not only providing superior performance in these two tasks but also significant robustness to a range of adversarial conditions.

Index Terms—recommender systems, graph transformer, semi-supervised learning, low homophily, heterophilic graphs

1 Introduction

Recommender systems have become an integral part of numerous different online applications, such as promoting sales (Amazon, Taobao) and enhancing multimedia experiences (Youtube, Netflix, Spotify) [1]. In recent years, recommender systems have evolved from models based on *matrix-factorisation* [2] to those leveraging *association rules* [3] and *deep learning* [4]. Now, many of today's state-of-the-art algorithms are relying on graph neural networks (GNNs) as a backbone because of their superior ability to learn effective representations of the kind of graph-structured data [5] that links users with items.

While a large number of GNN-based recommender systems have been designed, most follow the assumption of homophily, i.e., that users with similar features or the same class labels will be linked to items with the same or similar labels and features [5]. Yet real-world recommendation graphs do not always hold to this assumption. Rather, they frequently show heterophily, i.e., they contain indirectly linked users with dissimilar features and different class labels. For example, Fig. 1(a) illustrates the result of a shilling attack [6], where a malicious adversary has given the red item a 5-star rating. However, they have also camouflaged their behaviour by ratings some items in the same way as a genuine user (u_1). As such, in Fig. 1(b), the genuine user will be negatively affected by receiving recommendations for items falsely promoted by the adversary. This is an indirect influence that is highly problematic. As a reminder, a shilling attack in recommender systems refers to a type of system manipulation or sabotage where malicious user profiles and fabricated ratings are deliberately submitted to influence the recommendation output [7]. These attacks are typically executed with the intent to either promote or demote certain items in the recommendation lists.



(a) Target user is genuine. (b) Target user is malicious.

Fig. 1: Low homophily in a recommender system due to a shilling attack.

In general, low homophily in graphs can be addressed with dedicated GNN architectures for heterophilic graphs [8]. However, such techniques cannot be made to work directly with recommender systems since the traditional ratios of heterophily versus homophily do not apply. Unlike the case with other graph-based tasks, in rating graphs, one user does not directly link to another. Rather, users are indirectly linked through items. In other words, existing GNNs for heterophilic graphs that are based on neighbourhood aggregation will fail with rating graphs because the users are connected to items not to other users.

- *Thanh Tam Nguyen, Jun Jo, Quoc Viet Hung Nguyen, and Alan Wee-Chung Liew are with Griffith University Gold Coast, QLD 4215, Australia.*
  E-mail: {t.nguyen19,j.jo,quocviethung.nguyen,a.liew}@griffith.edu.au
- *Thanh Toan Nguyen is with Faculty of Information Technology, HUTECH University, Ho Chi Minh City 70000, Vietnam.*
  E-mail: nt.toan@hutech.edu.vn
- *Matthias Weidlich is with Humboldt-Universität zu Berlin, Berlin 10117, Germany.*
  E-mail: matthias.weidlich@hu-berlin.de
- *Hongzhi Yin is with The University of Queensland, QLD 4072 St Lucia, Australia.*
  E-mail: h.yin1@uq.edu.au

Corresponding author: Thanh Toan Nguyen.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

In this way, rating graphs are bipartite, and this calls for an alternative, purpose-built aggregation strategy.

In short, three challenges have to be addressed in order to build a robust GNN-based recommender system:

Challenge 1: Untruthful ratings obscure the authentic connections in rating graphs. For example, in Fig. 1(b), a malicious user has provided ratings that are inconsistent with those of genuine users so as not to promote items that are not part of the attack. The challenge is that models need to be robust against such attacks.

Challenge 2: Low homophily impairs neighbourhood aggregation, by commonly neglecting the labels assigned to nodes. The challenge, therefore, is to define aggregation strategies that incorporate the node classes.

Challenge 3: Models based on GNNs typically learn representations based on the features assigned to nodes, only exploiting node labels as signals in the training phase. Hence, the challenge is to integrate labels into the actual representations as a way of augmenting the features associated with the original node.

In this paper, we tackle the above challenges with a Partitioned Graph Transformer-based Recommendation model (PGT4Rec). Technically, the model includes a novel neighbourhood aggregation strategy, called partitioned aggregation (PA), that derives discriminative neighbourhood information from the labels. Moreover, PGT4Rec includes an embedding mechanism that embeds information on features, graph structures, and labels, which sits on top of a Transformer encoder. Compared to existing models [9], PGT4Rec's Transformer encoder is more light-weight, but it still incorporates a multi-head self-attention mechanism. Further, the partitioned aggregation preprocessing step addresses the lack of neighbourhood information common to most Transformer encoders.

Our contributions are summarised as follows.

- *Multi-task learning:* To address *Challenge 1*, we have developed a multi-task learning process that simultaneously predicts ratings and detects shilling attacks. Ratings information allows the model to detect the type of inconsistent behaviour associated with malicious users, which ultimately leads to more truthful recommendations.
- *Partitioned aggregation:* Our partitioned aggregation approach addresses *Challenge 2* by partitioning the nodes in a rating graph based on class labels. Partitioned aggregation also generalizes the links between users and items via multi-hop aggregation to capture higher-order neighbourhood information.
- *Threefold embeddings with graph transformer: Challenge 3* is addressed by capturing information on features, graph structures, and labels within the respective embeddings. These embeddings are then integrated into the Transformer encoder via a multi-head self-attention mechanism.

Experiments on a diverse range of datasets show the effectiveness of PGT4Rec at predicting ratings and detecting shilling attacks. The results also demonstrate the model's substantial robustness against adversarial conditions.

2 Problem Formulation

Below, we introduce the PGT4Rec model and then formalise the problem to be addressed.

2.1 Model

Consider a recommendation graph as weighted bipartite graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ (aka rating graph [6]), where $\mathcal{U} = \{u_1, \ldots, u_n\}$ represents a sets of n users, $V = \{v_1, \ldots, v_m\}$ and represents a set of m items. An undirected edge $(u, v, r_{uv}) \in \mathcal{E}$ indicates that user u has rated item v with a score r_{uv}. $\boldsymbol{R} \in \mathbb{R}^{n \times m}$ denotes the user-item rating matrix. Notably, some users $u' \in \mathcal{U}$ are malicious; they are shills that have input untruthful ratings to the platform $r_{u'v} \in \boldsymbol{R}$.

Now consider $\mathcal{F} = \{\boldsymbol{f}_{u_1}, \ldots, \boldsymbol{f}_{u_n}, \boldsymbol{f}_{v_1}, \ldots, \boldsymbol{f}_{v_m}\}$, which is a set of features assigned to each user u and each item v. Here, $\boldsymbol{f}_u$ is a $d_{\mathcal{U}}$-dimensional feature vector of a user , and $\boldsymbol{f}_v$ is a $d_{\mathcal{V}}$-dimensional feature vector of an item. For convenience, all feature vectors are modelled at the same length $d = d_{\mathcal{U}} + d_{\mathcal{V}}$, where padding is used if needed.

The users are assigned labels, modelled as a set $\mathcal{Y}$, but the setting considered is semi-supervised, so only a small number of users, denoted as $\hat{\mathcal{U}}$, are labelled, denoted as $\hat{\mathcal{Y}}$. Each user $u \in \hat{\mathcal{U}}$ has a scalar label $y_u \in \{1, \ldots, C\}$. While PGT4Rec generalises to any number of user classes C (e.g. minority users, majority uses, hardcore users as in [10]), for brevity's sake, only 2 classes are considered in this paper: genuine ($y_u = 1$) and malicious ($y_u = 2$). Hence, $C = 2$.

Taking our rating graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, each (undirected) triple $t = (u, v, u')$, with $u, u' \in \mathcal{U}$, $v \in \mathcal{V}$, (u, v) and $(v, u') \in \mathcal{E}$, is homophilic if u and u' belong to the same class. Otherwise, t is heterophilic. Most rating graphs will contain both homophilic and heterophilic triples. The degree of homophily in the rating graph is measured by the following equation:

$$\mathcal{H} = \frac{\sum_{(u,v,u') \in \mathcal{T}, y_u \neq y_{u'}} |r_{uv} - r_{vu'}|}{\sum_{(u,v,u') \in \mathcal{T}} |r_{uv} - r_{vu'}|} \tag{1}$$

Here, $\mathcal{T} = \{(u, v, u') : (u, v) \in \mathcal{E}, (v, u') \in \mathcal{E}\}$ is the set of all triples in $\mathcal{G}$, $\mathcal{N}^2(u) = \mathcal{N}(\mathcal{N}(u))$ is the two-hop neighbourhood of u, and $\Delta r(u, u') = \sum_{v \in \mathcal{V}, (u,v,u') \in \mathcal{T}} |r_{uv} - r_{vu'}|$ is the total differences in ratings made by u and u' for the common items between them. Note that $0 \leq \mathcal{H} < 1$. Some works studied local homophily measures [11], [12] but global homophily measures turn out to be more easily controlled via random graph processes for experimental purpose [13], [14], [15].

The novelty of our measure of homophily is that traditional homophily measures do not consider edge weights [13], [16], [17]. The motivation is to capture the nuanced relationships between user preferences and their connections. Traditional homophily measures consider only the similarity of labels, overlooking the complexity of user behavior in their ratings. By integrating rating-based similarities, the new measures provide a more accurate assessment of homophily, reflecting how aligned users' tastes are. This addresses *Challenge 1* by leveraging the detailed data in user ratings to identify genuine relationships and detect potential fraud, leading to more robust recommendations.

For example, in Fig. 1b, if a malicious user gives a high rating to a highly rated item, the item might end up being

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

rated more highly than the item they are trying to promote. To capture the two different cases shown in Fig. 1a and Fig. 1b, our measure of homophily considers the difference in ratings between genuine and malicious users.

2.2 Multi-task Problem Statement

Using the above model, we defined two tasks that the recommender system needs to perform.

Task 1 (Ratings Prediction)**.** *Given a rating graph $\mathcal{G}$ with an accompanying feature set $\mathcal{F}$, the model must predict the rating that a non-malicious user would assign to an item that they have not yet rated.*

Task 2 (Shilling Attack Detection)**.** *Given a rating graph $\mathcal{G}$ with an accompanying feature set $\mathcal{F}$ and a set of users $\mathcal{U}$, the model must classify the users into genuine users and malicious adversaries.*

These tasks are two main approaches in contemporary research for developing robust recommender systems [18]. However, these tasks have been studied separately, but integrating them into a unified framework remains largely unexplored [5]. Existing works on Task 1 focus on incorporating robust statistical methods, like M-estimators, into matrix factorization optimisation [19]. However, this approach often struggles to integrate rich user side information [6]. On the other hand, existing works on Task 2 aim to identify and remove fraudsters from rating datasets, but this can mistakenly exclude genuine users, leading to counterproductive results [20], [21]. These limitations motivate the design of our framework that fully utilises user reliability features to improve robustness against noisy data and functions as a unified, end-to-end model for both robust recommendations and fraudster detection.

TABLE 1: Summary of important notations.

Symbols	Definition
$\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$	rating graph (weighted bipartite graph)
$u \in \mathcal{U}$	a user node u in the set of all user nodes $\mathcal{U}$
$v \in \mathcal{V}$	an item node v in the set of all item nodes $\mathcal{V}$
r_{uv}	predicted rating of a user u on item v
$\hat{r}_{uv}$	groundtruth rating of a user u on item v
$i \in \mathcal{U} \cup \mathcal{V}$	a node (either a user or an item)
$\boldsymbol{f}_u$, $\boldsymbol{f}_v$, $\boldsymbol{f}_i$	feature vector of a user node u, an item node v, a node i
$\boldsymbol{H}(u)$	one-hop partitioned aggregation of a user node u
$\boldsymbol{H}(v)$	one-hop partitioned aggregation of an item node v
$\boldsymbol{H}^K(u)$	K-hop partitioned aggregation of a user node u
$\boldsymbol{H}^K(v)$	K-hop partitioned aggregation of an item node v
$\boldsymbol{H}_i$	input feature sequence of node i in the transformer encoder

3 Methodology

3.1 Approach Overview

As illustrated in Fig. 2, PGT4Rec consists of three steps: partitioned aggregation, the Transformer encoding, and multi-task learning. The first step is to pre-compute the multi-hop neighbourhood of each target node and perform partitioned aggregation, which generates a sequence of partition vectors. Next, this sequence is encoded through three types of embeddings that retain information about: a) the features, b) the graph structure, and c) the labels. A multi-head self-attention mechanism then adaptively re-weights

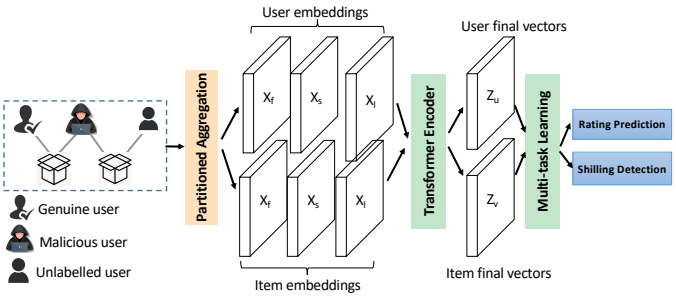


Fig. 2: Framework Overview.

the partition vectors while capturing neighbourhood information from the labels. The output vectors generated are then combined to produce a final representation of the target node. Note here that, although the user and item nodes are represented separately, mathematically, they are both treated in the same way. Last, we come to the multi-task learning step. Here, the user representations are used to detect shilling attacks, while both the user and item representations are used to predict the ratings.

It should be emphasized that this approach is not strictly based on a GNN. Rather, this approach simply overcomes the limitation of homophily in GNN-based recommender systems by using graph-based Transformer and a novel neighbourhood aggregation mechanism. Significantly, this approach is also highly efficient, as the partitioned aggregation procedure is performed only once and does not involve a full-graph Transformer; only a Transformer encoder is needed [9]. This addresses *Challenge 3* – augmenting a node's features with class labels.

It should be said that it is not easy to capture low homophily with a recommendation algorithm based on a two-layer graph [8]. As mentioned, malicious users can indirectly affect the recommendations a genuine user receives if the user and the adversary have rated common items.

3.2 Partitioned Aggregation (PA)

Traditional aggregation. In traditional aggregation, graph data is aggregated into neighbourhoods using the following common formulation:

$$agg(\{\boldsymbol{f}_v \mid \forall\, v \in \mathcal{N}(u)\}) = \frac{1}{\phi(.)} \sum_{v \in \mathcal{N}(u)} \boldsymbol{f}_v \qquad (2)$$

where $\phi(.)$ is a differentiable normalisation function (e.g., $\phi(.) = |\mathcal{N}(u)|^{\alpha}$ with α as the constant), and $\mathcal{N}(u)$ are the neighbours of node u. However, this form of traditional aggregation works poorly with low-homophily graphs, and especially poorly when nodes with different labels are connected to each other. Moreover, this approach to aggregation cannot be generalised to rating graphs because each user's neighbours are items and not other users. In short, traditional methods of aggregating neighbourhoods do not consider whether the node is a user or an item. In our case, however, our partitioned aggregation recognises that rating graphs contain two different types of nodes – users and items – and, hence, each needs to be treated differently. Additionally, the indirect impact of malicious users on genuine

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

users who have rated the same items means that directly aggregating neighbourhoods would not work.

Example 1. Let us consider a toy recommendation graph with an item v connected to a genuine user u_1, a malicious user u_2, and an unlabelled user u_3. The neighbourhood of u_1 is $\mathcal{N}(u_1) = \{v\}$, of u_2 is $\mathcal{N}(u_2) = \{v\}$, of u_3 is $\mathcal{N}(u_3) = \{v\}$, of v is $\mathcal{N}(v) = \{u_1, u_2, u_3\}$. The neighbourhood aggregation of u_1 is $agg(\{\boldsymbol{f}_v\}) = \boldsymbol{f}_v$, of u_2 is $agg(\{\boldsymbol{f}_v\}) = \boldsymbol{f}_v$, of u_3 is $agg(\{\boldsymbol{f}_v\}) = \boldsymbol{f}_v$, of v is $agg(\{\boldsymbol{f}_{u_1}, \boldsymbol{f}_{u_2}, \boldsymbol{f}_{u_3}\}) = \frac{1}{2}(\boldsymbol{f}_{u_1} + \boldsymbol{f}_{u_2} + \boldsymbol{f}_{u_3})$ with $\alpha = 1$.

One-hop partitioned aggregation. Motivated by this observation, we have devised a partitioned aggregation strategy that considers information about different classes. Inspired by Wang et al. [22], the idea is to integrate partially observed labels into the aggregation process and to re-weight the neighbourhood information for different classes. However, unlike Eq. 2, partitioned aggregation provides multiple outputs. Nodes with the same class label join the same partition and each partition is aggregated separately. This addresses *Challenge 2* – which implied that low homophily impairs neighbourhood aggregation.

Note that, during training, if a node is a target node for optimisation, its label (if available) will be masked to avoid label leakage, i.e., it will be considered as unlabelled. This is because, due to multi-hop propagation, the target node may also form its own neighbourhood. More specifically, consider a target item v and its one-hop neighbourhood of users $\mathcal{N}(v)$. The users $\mathcal{N}(v)$ are then divided into 3 partitions $\{U_1, U_2, U_*\}$, where U_1 contains all neighbouring genuine users, U_2 contains all neighbouring malicious users, and the last partition U_* contains all the unlabelled users. In this situation, the partitioned aggregation will return a sequence of 3 partition vectors:

$$\begin{aligned} \boldsymbol{H}(v) &= [\boldsymbol{h}_1(v), \boldsymbol{h}_2(v), \boldsymbol{h}_*(v)] \\ \text{where } \boldsymbol{h}_1(v) &= agg(\{\boldsymbol{f}_u \mid \forall\, u \in U_1\}), \\ \boldsymbol{h}_2(v) &= agg(\{\boldsymbol{f}_u \mid \forall\, u \in U_2\}), \\ \boldsymbol{h}_*(v) &= agg(\{\boldsymbol{f}_u \mid \forall\, u \in U_*\}). \end{aligned} \tag{3}$$

$\boldsymbol{H}(v)$ is a sequence of partition vectors. The aggregation function $agg(.)$ is defined by Eq. 2, and $\boldsymbol{h}_*(u)$ is the result of aggregating the masked and unlabelled users, noting that zeros will be assigned as default values to empty partitions.

Example 2. Continuing the toy example in Example 1, we have $U_1 = \{u_1\}$, $U_2 = \{u_2\}$, and $U_* = \{u_3\}$. Then, $\boldsymbol{h}_1(v) = \boldsymbol{f}_{u_1}$, $\boldsymbol{h}_2(v) = \boldsymbol{f}_{u_2}$, and $\boldsymbol{h}_*(v) = \boldsymbol{f}_{u_3}$. As a result, $\boldsymbol{H}(v) = [\boldsymbol{f}_{u_1}, \boldsymbol{f}_{u_2}, \boldsymbol{f}_{u_3}]$.

Using the above operations, one-hop aggregation is modelled for the users by swapping the roles of the item and user nodes, with the only difference being that the items do not have labels. Notably, this framework can be extended to classify items by partitioning the items according to their labels. This is how shilled items might be detected [23]. Specifically, consider the following vector for a target user:

$$\boldsymbol{H}(u) = [h_*(u)] \text{ where } h_*(u) = agg(\{\boldsymbol{f}_v \mid \forall\, v \in \mathcal{N}(u)\}) \tag{4}$$

Without loss of generality, one can actually define the same aggregation function for both users and items by partitioning the neighbours of any target node into 3 partitions $\{P_1, P_2, P_*\}$ (whether that be a user or an item). Here, P_1 contains the neighbouring nodes with label 1 (i.e. genuine users), P_2 contains the neighbouring nodes with label 2 (i.e. malicious users), and P_* contains the unlabelled nodes (item nodes or unlabelled user nodes). Note that, items are always in P_* as they do not have labels.

Example 3. Back to the toy example in Example 1, we have $\boldsymbol{H}(u_1) = \boldsymbol{H}(u_2) = \boldsymbol{H}(u_3) = [\boldsymbol{f}_v]$.

Multi-hop partitioned aggregation. To incorporate distant neighbours and capture higher-order semantics, the next step is to generalise the partitioned aggregation method to a multi-hop neighbourhood. In this variation, the K-hop neighbours $\{\mathcal{N}^k(i) \mid k = 1, \ldots, K\}$ are pre-computed for each target node $i \in \mathcal{U} \cup \mathcal{V}$, and the k-hop neighbourhood $\mathcal{N}^k(i) = \underbrace{\mathcal{N}(...(\mathcal{N}(i)))}_{k \text{ times}}$ is partitioned into 3 partitions $\{P_1^{(k)}, P_2^{(k)}, P_*^{(k)}\}$, where $P_1^{(k)}$ contains the genuine user nodes, $P_1^{(k)}$ contains the malicious user nodes, and $P_*^{(k)}$ contains the items and unlabelled users. Partitioned aggregation is then performed per hop:

$$\begin{aligned} \boldsymbol{H}_{\oplus}^{K}(i) &= \bigoplus_{k=1}^{K} \boldsymbol{H}^{(k)}(i), \\ \text{where } \boldsymbol{H}^{(k)}(i) &= [\boldsymbol{h}_1^{(k)}(i), \boldsymbol{h}_2^{(k)}(i), \boldsymbol{h}_*^{(k)}(i)] \\ \text{and } \boldsymbol{h}_c(i)^{(k)} &= agg(\{\boldsymbol{f}_{i'} \mid \forall\, i' \in P_c^{(k)}\}) \text{ with } c = \{1, 2, *\} \end{aligned} \tag{5}$$

Here, $\boldsymbol{H}_{\oplus}^{K}(i)$ is the discriminative aggregation result within K hops, $\boldsymbol{H}^{(k)}(i)$ is a sequence of the partition vectors of the k-th hop, and $\oplus$ denotes the concatenation operation. To apply self-attention to the target node and and classify it, the raw feature $\boldsymbol{x}_i$ and the partition vectors $\boldsymbol{H}_{\oplus}^{K}(i)$ are combined into a single sequence $\boldsymbol{H}_i = [\boldsymbol{f}_i]||\boldsymbol{H}_{\oplus}^{K}(i)$. This input feature sequence has $I = (C+1) \times K + 1$ vectors, where I is the sequence length.

Example 4. Combining the previous examples, let us consider the 2-hop aggregation of the user u_3. We have $\mathcal{N}^2(u_3) = \{v, u_1, u_2, u_3\}$, which is partitioned into $P_1^{(2)} = \{u_1\}$, $P_2^{(2)} = \{u_2\}$, $P_*^{(2)} = \{u_3, v\}$. Then $\boldsymbol{h}_1^{(2)}(u_3) = \boldsymbol{f}_{u_1}$, $\boldsymbol{h}_2^{(2)}(u_1) = \boldsymbol{f}_{u_2}$, and $\boldsymbol{h}_*^{(2)}(u_3) = \frac{1}{2}(\boldsymbol{f}_{u_3} + \boldsymbol{f}_v)$. Then, $\boldsymbol{H}^{(2)}(u_3) = [\boldsymbol{f}_{u_1}, \boldsymbol{f}_{u_2}, \frac{1}{2}(\boldsymbol{f}_{u_3} + \boldsymbol{f}_v)]$. As a result, $\boldsymbol{H}_{\oplus}^{2}(u_3) = [\boldsymbol{f}_v, \boldsymbol{f}_{u_1}, \boldsymbol{f}_{u_2}, \frac{1}{2}(\boldsymbol{f}_{u_3} + \boldsymbol{f}_v)]$. And finally, $\boldsymbol{H}_{u_3} = [\boldsymbol{f}_{u_3}, \boldsymbol{f}_v, \boldsymbol{f}_{u_1}, \boldsymbol{f}_{u_2}, \frac{1}{2}(\boldsymbol{f}_{u_3} + \boldsymbol{f}_v)]$.

3.3 Embeddings with Transformer Encoder

In effect, the partitioned aggregation procedure transforms a high-dimensional graph structure into sequential data. In the learning process, three types of embeddings capture: a) the features, b) the graph structure, and c) the labels, each of which is treated as an input feature sequence to the Transformer encoder. The vectors from different hops and partitions are re-weighted automatically through multi-head self-attention. Choosing a Transformer instead of an LSTM or RNN means signals can be captured from different hops across the labels simultaneously, as derived by the partitioned aggregation.

Feature embedding. The first step in generating the input sequence for the Transformer encoder is to linearly embed

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

the partition vector of each node i: $\boldsymbol{X}_f(i) = \sigma(\varphi(\boldsymbol{H}_i))$, where $\varphi : \mathbb{R}^{I\times d} \rightarrow \mathbb{R}^{I\times d_X}$ is a projection function, d_X represents the embedding dimension, and σ is the activation function.

Structure embedding. The Transformer encoder treats each partition vector separately. Hence, an embedding is required to include the multi-hop structural information. For given a target node i, this is formulated as:

$$\boldsymbol{X}_s = [\boldsymbol{E}_s(0), \overbrace{\boldsymbol{E}_s(1), \ldots, \boldsymbol{E}_s(1)}^{3 \text{ embeddings}}, \ldots, \overbrace{\boldsymbol{E}_s(K), \ldots, \boldsymbol{E}_s(K)}^{3 \text{ embeddings}}] \quad (6)$$

where $\boldsymbol{E}_s(.)$ denotes a learnable embedding that is stored as a $(K+1)\times d_X$ matrix. In other words, the partition vector of the k-th hop in the input sequence is assigned by a learnable embedding $\boldsymbol{E}_s(k) = [\mathbb{1}_k(j)]^K \times \boldsymbol{W}_{(K+1)\times d_X}$, and $[\mathbb{1}_k(j)]^K$ is a one-hot vector of length $K+1$ (i.e., $\mathbb{1}_k(j) = 1$ if $j = k$, and $\mathbb{1}_k(j) = 0$) otherwise. As an example, $[\mathbb{1}_1(j)]^2 = [0, 1, 0]$, where the first position is for the target node.

Note that there are 3 embeddings for each hop $\boldsymbol{E}_s(k)$ ($k = 1..K$) to capture the characteristics of each class (one for genuine users, one for malicious users, and one for unlabelled nodes), which will easily bypass any downside of low homophily. Also, $\boldsymbol{E}_s(0)$ is placed at the beginning of $\boldsymbol{X}_s$ to enable self-attention.

Label embedding. Unlike existing techniques that only incorporate labels as signals during training, the `PGT4Rec` framework uses these labels to augment the feature space. Inspired by the label encodings outlined in UniMP [24], `PGT4Rec` adds a learnable embedding to the input sequence of each target node. In the partitioned aggregation, neighbours are partitioned according to the class labels such that the indices of the label embeddings correspond to these labels. The label embedding $\boldsymbol{X}_l$ is:

$$\boldsymbol{X}_l = [\boldsymbol{E}_l(*), \overbrace{\boldsymbol{E}_l(1), \boldsymbol{E}_l(2), \boldsymbol{E}_l(*)}^{1\text{st hop}}, \ldots, \overbrace{\boldsymbol{E}_l(1), \boldsymbol{E}_l(2), \boldsymbol{E}_l(*)}^{K\text{-th hop}}] \quad (7)$$

where $\boldsymbol{E}_l(c)$ (with $c = 1, 2, *$) denotes a learnable embedding that is stored as a $3 \times d_X$ matrix (one dimension for class $c = 1$, one for class $c = 2$, and one for the unlabelled $c = *$). Note that the target node is encoded with $\boldsymbol{E}_l(*)$ to avoid label leakage. Again, placing $\boldsymbol{E}_l(*)$ at the beginning of $\boldsymbol{X}_l$ enables self-attention.

Transformer encoder. These three embedding types, which are all of equal sizes, are then fused to construct an input sequence for the Transformer encoder:

$$\boldsymbol{X} = \boldsymbol{X}_f + \boldsymbol{X}_s + \boldsymbol{X}_l. \quad (8)$$

Here, the multi-head attention modules facilitate deep interactions among the vectors, after which the hidden vector $\boldsymbol{x}_i$ of the i-th partition vector in $\boldsymbol{X}$ is updated from the layer l to the layer $l+1$:

$$\begin{aligned} &\boldsymbol{x}_i^0 = \boldsymbol{x}_i, \boldsymbol{x}_i^{l+1} = \left(\bigoplus_{m=1}^{M} head_m\right)\boldsymbol{O}^l, \\ &head_m = Attention(\boldsymbol{Q}^{m,l}\boldsymbol{x}_i^l, \boldsymbol{K}^{m,l}\boldsymbol{x}_j^l, \boldsymbol{V}^{m,l}\boldsymbol{x}_j^l) \\ &- \sum_{j\in S} w_{ij}(\boldsymbol{V}^{m,l}\boldsymbol{x}_j^l), w_{ij} = softmax_j\left(\frac{\boldsymbol{Q}^{m,l}\boldsymbol{x}_i^l \cdot \boldsymbol{K}^{m,l}\boldsymbol{x}_j^l}{\sqrt{d_X}}\right), \end{aligned} \quad (9)$$

where $\boldsymbol{Q}^{m,l}$, $\boldsymbol{K}^{m,l}$, $\boldsymbol{V}^{m,l}$ are the learnable weights of the m-th attention head, $\boldsymbol{x}_j^l$ is the j-th hidden vector of layer l, and $\boldsymbol{O}^l$ is a projection to match the dimensions between adjacent layers. Additionally, Layer Normalisation and a position-wise 2-layer MLP is applied after the multi-head attention module to avoid overfitting [25].

3.4 Multi-task Learning

Finally, it is time to consider the output sequence of the last layer of the encoder, where the output vectors $\boldsymbol{z}_u$ per user u and $\boldsymbol{z}_v$ per item v are extracted. The user vectors are used to detect shilling attacks, while both the item and user vectors are used to predict ratings.

Shilling Attack Detection. Following the embedding step and the Transformer encoder, an MLP classifier is appended as a new layer. This classifier derives the class probabilities per user:

$$p_u = sigmoid(MLP(z_u)) \quad (10)$$

As a result, $p_u(c)$ represents the probability that user u has the label c (1 for genuine, 2 for malicious). The optimisation objective for detecting a shilling attack is then defined via a regularized cross-entropy:

$$\mathcal{L}_{fraudster} = \sum_{u\in\hat{U}} -\sum_{c=1}^{2} \mathbb{1}_{y_u}(c)\log p_u(c) + \beta||\theta||_2^2, \quad (11)$$

where $\mathbb{1}_{y_u}(c)$ is an indicator function that is equal to 1 if $c = y_u$, and 0 otherwise. θ is `PGT4Rec`'s parameter set, while β is a regularisation hyperparameter to avoid overfitting.

Rating prediction. The representations generated for the users and items, i.e., $\boldsymbol{z}_u$ and $\boldsymbol{z}_v$, contain comprehensive information on both the corresponding users and items as well as the nodes connected to them. Further, not only do they capture directly observed (first-order) interactions as is the case with most existing models [5], they also capture the (higher order) multi-hop interactions to address the issue of low homophily. As a bonus, this avoids the need for two GNNs, one for users and one for items, which Zhang et al. implemented [6]. This helps to increase efficiency.

The rating scores r_{uv} for each user-item pair are then predicted using the learned representations. To this end, these latent representations are concatenated prior to using another layer l in the MLP $g(.)$ coupled with a projection layer to regress the ratings:

$$r_{uv} = \boldsymbol{w}_{project}^T MLP(\boldsymbol{z}_u||\boldsymbol{z}_v) \quad (12)$$

where r_{uv} denotes the predicted rating. Measuring the error of r_{uv} against the ground truth rating $\hat{r}_{uv}$ optimises the MLP-based regressor based on the squared error:

$$\mathcal{L}_{rating} = \frac{1}{\mathcal{E}} \sum_{\forall(u,v)\in\mathcal{E}} (r_{uv} - \hat{r}_{uv})^2 \quad (13)$$

Model Training. Significantly, the above tasks are not trained separately. Rather, their loss functions are combined:

$$\mathcal{L} = \mathcal{L}_{rating} + \lambda\mathcal{L}_{fraudster} \quad (14)$$

where λ is a hyper-parameter used to balance both losses.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

3.5 Further Optimizations

Enhanced user representations (optim1). Malicious users is often less stable than genuine users [6]. So, to help detect them, we include the mean square of all prediction errors for a given user u and the items in $\mathcal{N}(u)$ as a feature δ_u:

$$\delta_u = \frac{1}{|\mathcal{N}(u)|} \sum_{\forall\ v \in \mathcal{N}(u)} |r_{uv} - \hat{r}_{uv}|^2 \tag{15}$$

The enhanced embedding is then derived as $\boldsymbol{z}'_u = \boldsymbol{z}_u \oplus \delta_u$.

Influence-based recommendation (optim2). Shilling attacks often result in untruthful ratings that can distort any subsequent predictions based on that data. However, directly optimising those recommendations through a least squares measure, which has an unbounded influence function, is not a robust solution to the problem [6]. For this reason, Eq. 10 considers the probability that user u is a genuine user with $p_u(genuine)$. If that probability is low, it reduces the influence of that user on the prediction, as captured by a modified loss function used for ratings prediction:

$$\mathcal{L}'_{rating} = \frac{1}{\mathcal{E}} \sum_{\forall\ (u,v) \in \mathcal{E}} p_u(genuine)(r_{uv} - \hat{r}_{uv})^2 \tag{16}$$

The predicted probability in Eq. 11 guides the ratings prediction optimisation. In turn, the predicted ratings serve as an auxiliary feature for detecting shilling attacks. Here, performance is optimised via Eq. 14. In this way, both tasks are interlinked within PGT4Rec's training process. We studied the impact of adding such "dependencies" between the loss functions through an ablation experiment where we compared two versions of framework - one with these joint optimisations and one without. The results are discussed in §4.4.

Scalability. As mentioned, the partitioned aggregation step is conducted only once. With a large rating graph, the aggregation can be parallelised per target node, as it only uses the raw feature vectors. Moreover, the multi-hop neighbourhoods can be computed with dynamic programming, as they concatenate the respective vectors.

The partitioned aggregation procedure is similar to a flat aggregation version of GraphSAGE [26]. Since the K-hop neighbourhood is bounded by $\mathcal{O}(a^K)$, where a is the average degree of the rating graph. Thus, the time complexity of PA is $\mathcal{O}(|\mathcal{U} \cup \mathcal{V}|a^K d)$, where d is the length of feature vectors. In terms of the embedding step, all embedding dimensions are considered to be of size d. The complexity of the transformer encoder is $\mathcal{O}(L(I^2 d + I^2 d^2))$, where L is the number of layers. Hence, the overall time complexity of PGT4Rec is $\mathcal{O}(|\mathcal{U} \cup \mathcal{V}|a^K d + eL(I^2 d + I^2 d^2))$ $= \mathcal{O}(|\mathcal{U} \cup \mathcal{V}|a^K d + eL \times constant)$, where e is the number of training epochs. With existing GNN-based models (e.g., a GCN or GraphTransformer), the overall time complexity would be about $\mathcal{O}(eL|\mathcal{U} \cup \mathcal{V}|a^K d^2)$, which is higher.

4 Experiments

Our experiments answer the following questions:

(RQ1) Does PGT4Rec outperform a range of today's baseline techniques in both ratings prediction and shilling attack detection?

(RQ2) What is the importance of each framework's component?

(RQ3)) What influence does label information and its availability have on our model?

(RQ4) What is the robustness of PGT4Rec to different shilling attacks?

(RQ5) How do different data splitting strategies affect recommendation performance?

(RQ6) How sensitive is PGT4Rec to changes in the hyperparameters?

(RQ7) How efficient is PGT4Rec compared to graph-based methods?

4.1 Experiment Setup

Datasets. We used two real-world public datasets in our experiments, both of which include ground-truth information. These datasets have been used as the primary evaluation in many researches on shilling attacks in recommender systems [6], [27], [28]. Dataset statistics are given in Table 2. *Class Ratio* is the ratio between genuine and malicious users.

- **Yelp [29]:** The dataset comprises hotel and restaurant reviews. All spam reviews are labelled, making it easy to divide malicious users from genuine users. The Yelp dataset was built by [30] and used in many previous works [6], [29], [31], [32]. While the details can be found in these works, we summarise the ground truth as follows. The dataset includes reviews for various restaurants and hotels in Chicago, New York, New Jersey, Vermont, Connecticut, Pennsylvania, etc. Yelp employs a filtering algorithm to identify fake or suspicious reviews, placing them into a filtered list. Both recommended and filtered reviews are publicly accessible: recommended reviews are displayed on the business's main Yelp page, while filtered/unrecommended reviews can be viewed via a link at the bottom of the page. Although the Yelp anti-fraud filter is not flawless (thus providing "near" ground truth), it has been shown to yield accurate results [31]. As a result, many subsequent works [6], [29], [32] define the ground truth for genuine users if they do not have fake reviews.
- **Amazon [6]:** This dataset contains a series of movie reviews (including ratings). The users have voted on each review for its helpfulness. A user is classified as genuine if $\geq$ 70% of the votes their review received were helpful, and as malicious if only $\leq$ 30% of the votes received were helpful. The Amazon dataset was built by [33], [34] and was used in many previous works [6], [35]. While the details can be found in these works, we summarise the ground truth as follows. On Amazon.com, reviews are given a star rating from 1 to 5, and users can also mark reviews as either "helpful" or "not helpful". Fayazi et al. [34] calculates the helpfulness ratio by dividing the number of helpful votes by the total number of votes:

  $$\text{Helpfulness Ratio} = \frac{\text{helpful votes}}{\text{helpful votes} + \text{not helpful votes}}.$$

  They then grouped the reviews by their star ratings, differentiating between deceptive and legitimate reviewers. In their experiments, Fayazi et al. [34] found out

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

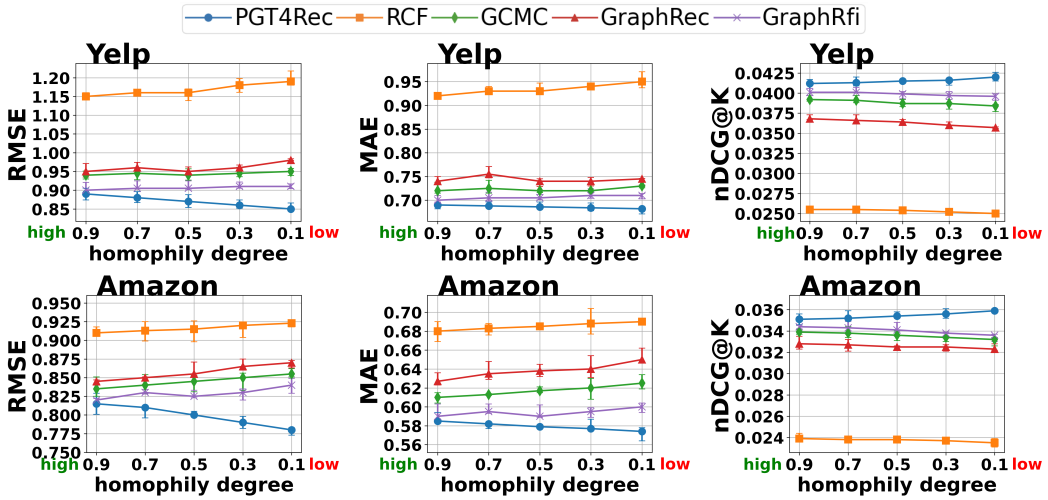


Fig. 3: Performance on robust rating prediction.

that helpfulness ratio is a good indicator of malicious bbehaviour. As a result, many subsequent works [6], [35] define the ground truth for genuine users based on their helpfulness ratio.

As such, giving a deceptive, non-helpful review makes a user malicious. Additionally, similar to Shi et al. [36], we evaluated the robustness of our approach with both the original dataset and with different levels of homophily.

TABLE 2: Dataset Statistics.

Dataset	#Users	Class Ratio	#Products	#Edges/Ratings	Homophily
Yelp	32393	7/3	4670	293936	0.4902
Amazon	12630	7/3	4746	250423	0.5178

Node features. In a recommender system, a user's behavioural traits offer valuable insights into their preferences. These traits can also serve as vital indicators for detecting fraudulent activity. Hence, for good recommendation performance and to help detect shilling attacks, it is wise to harness these traits as features, and use them when generating latent user representations. Common behavioural traits encompass metrics like *the number of rated products*, *the ratio of positive to negative ratings*, *rating entropy*, *feedback summary length*, and *sentiment analysis of review text*. For a comprehensive list of these features and their computation, interested readers can consult our previous work [6].

Homophily control in recommender systems. To control the degree of homophily, we followed [16], [37] and generated synthetic edges on top of the original graph via a modified preferential attachment process for weighted bipartite graphs. In this process, malicious users are incrementally added to the graph until their number reaches a preset level. At each step, a new malicious user u' is indirectly connected to an existing genuine user u (via items) according to a probability of $p_{uu'}$, which is proportional to the node degree of u' and the class ratio H:

$$H = \frac{|\{(u_1, v, u_2) \in \mathcal{T} : y_{u_1} = \text{genuine}, y_{u_2} = \text{malicious}\}|}{|\{u_1, v, u_2) \in \mathcal{T} : y_{u_1} = \text{genuine}\}|}.$$

The rating score of each new edge is initially set to a fixed value r_0, which should be set to the minimum rating given if a malicious user wanted to promote a target item. Each time new malicious users are added, all existing untruthful scores are updated by $r_{uv} = r_{uv} + \delta_u \frac{r_v - r_{uv}}{d_u}$, where r_v is the average of the truthful ratings for item v, d_u is the node degree of the malicious user u, and $\delta_u \in (0, 1)$ is an offset parameter (e.g., set by a Gaussian distribution). Here, the idea is to decrease the homophily gradually, without creating suspicious behaviour by mirroring the connectivity between genuine and malicious users or any ratings made by genuine users.

Baselines. We evaluated `PGT4Rec`'s predictions against the following baselines:

- **RCF** [38]: The Robust Collaborative Filtering model is a shilling-resistant matrix factorisation algorithm based on robust M-estimators.
- **GCMC** [39]: GCMC leverages graph auto-encoding with message passing on bipartite graphs.
- **GraphRec** [40]: GraphRec is a GNN-based approach that derives recommendations based on user-item interactions.
- **GraphRFI** [6]: GraphRFI uses a graph convolution network and a neural random forest for recommendation and shilling attack detection.

As benchmark comparators for detecting shilling attacks, we compared `PGT4Rec` against the following techniques:

- **PCA-VarSelect** [41]: PCA-VarSelect is an unsupervised detection approach that assumes that the number of malicious users is known.
- **SVM-TIA** [42]: This is an SVM-based technique for attack detection.
- **GraphRFI** [6]: Again, GraphRFI, as mentioned above, is a baseline technique.
- **H^2-FDetector** [36]: This detector considers the effect of homophilic and heterophilic connections.

Notably there are several other GNN-based models that work in a similar fashion to H^2-FDetector, such as RioGNN [43], PC-GNN [28], and FRAUDRE [44]. However, as we are particularly interested in reporting the results from a wide range of different model types, we conducted some experiments with this set of models. The results confirmed that H^2-FDetector is the state-of-the-art for GNN-based models,

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

and so we selected this as the representative baseline.

Recently, some generic GNNs have been put forward for handling heterophilic graphs, including GPNN [45], HOG-GNN [46], and BM-GCN [47]. However, these are not as specialised as H^2-FDetector. Another relevant technique is relation-aware GNNs [22] but these are only applicable to graphs with multiple different types of edges.

Evaluation Metrics. The ratings prediction were evaluated in terms of accuracy using mean absolute error (**MAE**) and root mean square error (**RMSE**). Additionally, we utilized Normalised Discounted Cumulative Gain (**nDCG@K**), a standard metric for quantifying the positive relevance of results within the top-K recommendation list, where $K = 20$ was set for all experiments [6].

To assess how effective PGT4Rec is at detecting shilling attacks, we calculated the Precision, Recall, and F1-scores. We considered a true positive to be a correctly detected malicious user, and a false positive to be a genuine user misclassified as a malicious user. Other contextual metrics are available [48], but considered irrelevant as we do not use any contextual information.

We followed the same experimental design as outlined in [6], where unlabelled users were not included when computing the evaluation metrics, but they were included in the training. In terms of ratings predictions, we randomly chose 20% of the truthful ratings as the test set. The remaining truthful ratings were used as the initial training set, which was gradually enlarged with malicious users and their ratings. To detect shilling attacks, we randomly chose 80% of the labelled users for the training set and used the remaining labelled users for the test set.

Hyperparameter tuning. We observed that the effect of changing the constant α and the regularisation hyperparameter β was negligible, so we set them to 1 and 2, respectively. The sensitivity of the embedding dimension d_X and the balancing factor λ were set to the optimal value, which is studied and discussed in greater detail in §4.6.

Environment. We implemented our model in Python / Pytorch and ran the experiments on a 2.4GHz CPU, 24GB RAM workstation, reporting averages over 20 runs.

4.2 Robust Recommendation

The performance of recommendation algorithms is critically dependent on how well they can predict the ratings. Fig. 3 shows the RMSEs, MAEs and nDCG@K for all the baselines. A random 20% of truth ratings was selected as the test set. The remaining truthful ratings along with 80% of the malicious users and corresponding ratings were used as the training set [6]. PGT4Rec shows superior performance over the competitors. PGT4Rec's dominance is more pronounced as the homophily ratio decreases, which suggests that PGT4Rec is particularly adept at handling diverse and less similar interactions, even to the extent that it may be pushing the boundaries of what we typically expect from a recommender system in terms of handling dissimilar data.

The GNN-based models, which include GraphRFI, GraphRec, and GCMC, also display commendable performance, outshining other methods. This performance is indicative of the intrinsic advantage GNNs possess in recommendation tasks due to their ability to incorporate the topology of the rating graph. By considering the connections and the structure of the user-item interactions, GNNs can effectively leverage relational information which is often pivotal for accurate recommendations.

Yet, in low homophily settings, the neighbourhoods did include mixed nodes of different classes. As a result, the GNN-based baselines do not perform well in low homophily as our method. Moreover, when the homophily degree is reduced (due to our homophily control mechanism), sometimes it actually provides more information (e.g. more interactions between malicious users and genuine users). Only PGT4Rec can leverage these interactions via the proposed partitioned aggregation to dissect the influence of malicious users.

It is worth noting that the trend for PGT4Rec might seem to be counter-intuitive at first. But in fact, homophily degree is varied via a specialised random graph process as described in the "homophily control" discussion (Section 4.1). As such, the graph data with different homophily degrees can be slightly different. Moreover, as the homophily degree is reduced with this control, new edges (ratings) might be created. As a result, the overall performance with more number of ratings might be better than the case with less number of ratings if the recommender keeps predicting correct ratings and avoids previous mistakes, as more higher-order information is available to be captured via our multi-hop partitioned aggregation.

4.3 Shilling Attack Detection

The capability of PGT4Rec to discern shilling attacks within the intricate scenarios of low and high homophily is central to understanding its robustness as a recommendation system. In testing this ability, we utilised known labels of malicious users within our datasets to evaluate performance. We selected 80% of the labeled users at random for the training set and used the remaining labeled users for the test set [6]. As indicated in Table 3, PGT4Rec demonstrated superior performance across all metrics – precision, recall, and F1-scores – outshining the baseline methods.

TABLE 3: Performance on shilling attack detection.

	Method	Low Homophily			High Homophily		
		Prec.	Rec.	F1	Prec.	Rec.	F1
Yelp	PCA-VarSelect	0.765	0.742	0.753	0.805	0.813	0.809
	SVM-TIA	0.776	0.782	0.779	0.823	0.821	0.822
	H2-FDetector	0.734	0.742	0.738	0.781	0.793	0.787
	GraphRFI	0.876	0.892	0.884	0.962	**0.971**	0.966
	PGT4Rec	**0.923**	**0.919**	**0.921**	**0.968**	0.969	**0.968**
Amazon	PCA-VarSelect	0.741	0.739	0.74	0.809	0.802	0.805
	SVM-TIA	0.779	0.762	0.77	0.819	0.827	0.823
	H2-FDetector	0.736	0.743	0.739	0.798	0.81	0.804
	GraphRFI	0.897	0.871	0.884	**0.934**	0.932	0.933
	PGT4Rec	**0.909**	**0.914**	**0.911**	0.929	**0.94**	**0.934**

It is imperative to underscore the differing approaches to shilling attack detection among the methods compared. The initial three baselines, PCA-VarSelect, SVM-TIA, and H2-FDetector, tackle the problem of shilling attack detection as an isolated issue, independent of the nuances of rating predictions. They do not integrate the attack detection with the recommendation process, which could be a reason for their relatively lower performance. Their methodology likely revolves around identifying outliers or anomalies in

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

user behaviour without considering the contextual information provided by the rating predictions.

In contrast, GraphRFI and PGT4Rec adopt a more holistic approach by incorporating shilling attack detection as part of the recommendation process, leveraging the connections and patterns within the rating data. PGT4Rec's superior results in both low and high homophily settings suggest that it is not just the integration of detection with prediction that makes it effective, but the specific manner in which PGT4Rec processes and utilises the information. In low homophily contexts, where user similarities are scarce and patterns are less pronounced, PGT4Rec still manages to identify malicious behaviour with high accuracy. This implies a significant resilience to noise and an ability to discern subtle patterns indicative of shilling.

4.4 Ablation Study

As default, the train/test ratio for labeled users is 80/20. A random 20% of true ratings was selected as the test set. The remaining true ratings along with 80% of the malicious users and corresponding ratings were used as training [6].

Step-wise Ablation. Table 4 shows the results for each step of the ablation study. First, we replace the partitioned aggregation with a simple mean aggregator but kept the Transformer encoder (SA+TE). With the variant, the F1-score dropped by 5.4% on Yelp and 5.2% on Amazon. This highlights that partitioned aggregation can mitigate the issues stemming from low homophily. Second, we combined the partitioned aggregation procedure with other encoders (PA+RNN and PA+LSTM). This also resulted in lower F1 scores with both datasets. From this, we concluded that the Transformer encoder is better at capturing neighbourhoods, which translates to a better re-weighting of the partition vectors for different labels. To study the effectiveness of our optimisations from §3.5, we included a plain approach (PGT4Rec w/o opts) and each separate optimisation (optim1 and optim2), which yielded slightly worse results.

TABLE 4: Ablation study.

	Method	Recommend.			Shilling Detection		
		RMSE	MAE	nDCG@K	Pre.	Rec.	F1
Yelp	PA + RNN	0.892	0.734	0.0344	0.918	0.921	0.919
	PA + LSTM	0.873	0.712	0.0385	0.949	0.951	0.950
	SA + TE	0.881	0.721	0.0381	0.927	0.906	0.916
	PGT4Rec w/o optims	0.864	0.693	0.0408	0.952	0.948	0.950
	PGT4Rec optim1 only	0.853	0.684	0.0420	0.963	0.964	0.963
	PGT4Rec optim2 only	0.856	0.682	0.0414	0.959	0.960	0.959
	PGT4Rec	**0.846**	**0.677**	**0.0420**	**0.968**	**0.969**	**0.968**
Amazon	PA + RNN	0.834	0.623	0.0298	0.890	0.903	0.896
	PA + LSTM	0.792	0.604	0.0324	0.913	0.921	0.917
	SA + TE	0.784	0.584	0.0341	0.881	0.889	0.885
	PGT4Rec w/o optims	0.781	0.582	0.0347	0.908	0.932	0.920
	PGT4Rec optim1 only	0.778	0.579	0.0359	0.921	0.937	0.929
	PGT4Rec optim2 only	0.779	0.580	0.0353	0.916	0.934	0.925
	PGT4Rec	**0.775**	**0.576**	**0.0359**	**0.929**	**0.940**	**0.934**

Embedding-wise Ablation. Fig. 4 illustrates the contribution of the different types of embeddings, including the feature embeddings ($\boldsymbol{X}_f$), the structure embeddings ($\boldsymbol{X}_s$), the label embeddings ($\boldsymbol{X}_l$), and various combinations of the three. Here, combining all types of embeddings produced the best results. The model also converged faster with both tasks: robust recommendation (F1 Score) and shilling attack detection (RMSE). we observed similar results with the Amazon dataset, omitted for the sake of brevity.

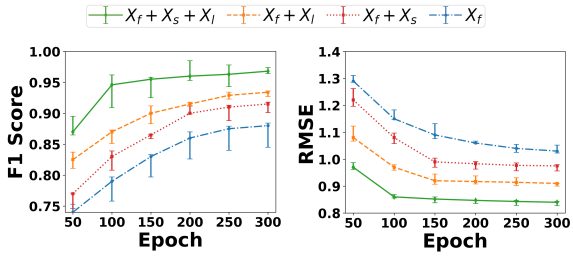


Fig. 4: Effect of learnable embeddings (Yelp).

Effects of Multi-hop Embeddings. To assess the efficacy of learnable embeddings (§3.3), we analyse different combinations of X_l (label embedding), X_s (structure embedding), and X_f (feature embedding). According to Table 5, PGT4Rec achieves optimal performance when utilising all learnable embeddings (X_l, X_s, X_f). Notably, the label embedding X_l alone boosts the performance by up to 4%. Other variants without partitioned aggregation tend to perform poorly, highlighting the crucial role of both partitioned aggregation and label embedding.

TABLE 5: Effects of multi-hop embeddings.

Embeddings (n_hops=4)		Yelp		Amazon	
		RMSE	F1 Score	RMSE	F1 Score
w/ PA	X_l, X_s, X_f	0.846	0.968	0.775	0.934
	X_s, X_f	0.825	0.945	0.748	0.912
	X_l	0.838	0.952	0.762	0.926
	X_s	0.822	0.940	0.741	0.904
	X_f	0.817	0.936	0.734	0.897
w/o PA	X_s, X_f	0.820	0.932	0.735	0.899
	X_s	0.816	0.925	0.728	0.895
	X_f	0.804	0.930	0.731	0.892

Quality of Multi-hope Embeddings. To illustrate the effectiveness of PGT4Rec, we provide visualisations of the attention scores for users in different classes (malicious and genuine). Fig. 5 highlights attention scores across various hops. This illustrates the importance of multi-hope embeddings and PGT4Rec utilises higher-order neighbourhood information. Due to the complex structure between users and items in the rating graphs, higher hops tend to reveal more information about user behaviour.

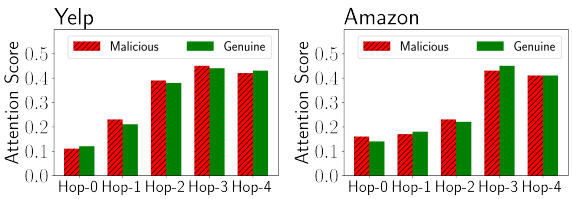


Fig. 5: Attention scores of multi-hop embeddings

Effects of Partitioned Aggregation. The main advantage of PGT4Rec is its partitioned aggregation mechanism. To verify its contribution, we compared PGT4Rec to the state-of-the-art GNN-based models, including a graph convolutional network (GCN [49]), graph attention (GA [50]) and graph Transformer (GT [51])—we compare the performance of these GNN-models with and without label aggregation, i.e., with partitioned aggregation and without. We also compared partitioned aggregation to C&S [33], a recent method

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

that uses label propagation to enhance the power of GNNs. Fig. 6 presents the results in terms of RMSE and F1 Score (again, other metrics are omitted for brevity sake due to similar trends). From the results, we see that C&S helps to improve performance with both tasks for all of these GNN-based models. However, using partitioned aggregation consistently delivered the best results, outperforming the C&S method by a significant margin.

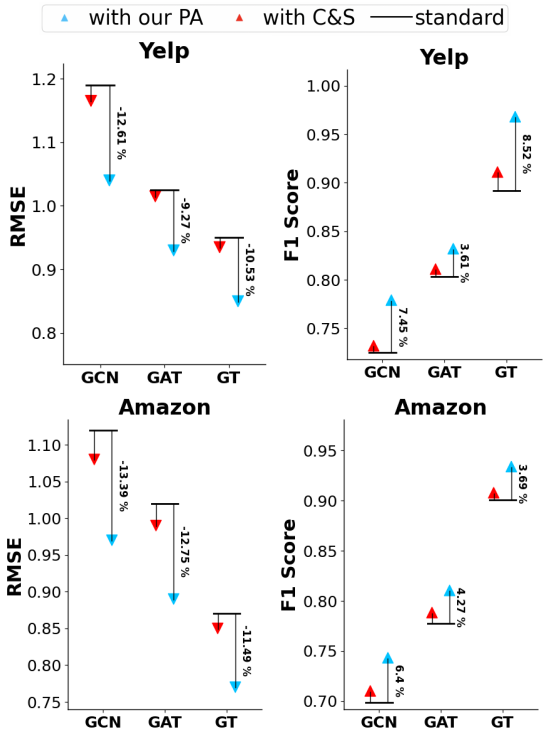


Fig. 6: Effects of Label Aggregation.

4.5 Effects of Semi-Supervision

In this experiment, we demonstrate the sensitivity of the model. As the solution is formulated in a semi-supervised setting, we consider the variety of the training ratio (the percentage of nodes in the training set) and the label ratio (the percentage of labelled nodes). We vary these ratios from 20% to 80% to test the stability of the models. Results are presented in Table 6. For brevity sake, we report RMSE and F1 Score as other metrics show similar trends.

Although performance with both tasks declined as the ratios decreased, `PGT4Rec` still produced competitive results with limited training information. In other words, `PGT4Rec` still performed well with relatively few labelled users. Another interesting finding is that PGT4Rec outperformed the homophily-based shilling attack detector H^2-FDetector [36] (see Table 3) without small label information.

4.6 Robustness against Different Attacks

In general, there are three types of shilling attacks: random, average, and hate attacks [52]. In these attacks, a malicious user will typically assign a high rating to their target items to promote them, while giving low ratings to "filler" items to help make themselves look like normal users. In a random attack, the adversary selects their filler items randomly; further, they assign random ratings. The filler items are also selected randomly in an average attack; however, the ratings assigned are average ratings particular to that platform. In a hate attack, the adversary demotes their target items by assigning them the minimum rating, while the filler items are given a maximum rating. This type of attack is commonly used to take down one's competitors. In all these cases, the attack size is the number of malicious users injected into the system [52].

TABLE 6: Effects of Label Information.

% training	% label	Yelp		Amazon	
		RMSE	F1 Score	RMSE	F1 Score
80	80	0.846	0.968	0.775	0.934
80	60	0.865	0.966	0.779	0.926
80	40	0.868	0.949	0.784	0.915
80	20	0.873	0.930	0.797	0.902
60	80	0.882	0.915	0.802	0.895
60	60	0.890	0.911	0.810	0.889
60	40	0.897	0.902	0.813	0.881
60	20	0.906	0.877	0.824	0.880
40	80	0.919	0.870	0.847	0.877
40	60	0.926	0.866	0.856	0.863
40	40	0.930	0.862	0.860	0.842
40	20	0.937	0.842	0.863	0.828
20	80	0.958	0.833	0.880	0.813
20	60	0.971	0.805	0.894	0.810
20	40	0.974	0.802	0.897	0.804
20	20	0.980	0.796	0.903	0.768

Fig. 7 shows an examination of the resilience of various competing strategies against shilling attacks, conducted on the Yelp and Amazon datasets. As default, the train/test ratio for labeled users is 80/20. A random 20% of truth ratings was selected as the test set. The remaining truthful ratings along with 80% of the malicious users and corresponding ratings were used as the training set [6]. Our solution, `PGT4Rec`, consistently yields the most precise and accurate predicted ratings in terms of RMSE, MAE, and nDCG@K. This achievement underscores the integral role that detecting shills plays in safeguarding the recommendation. Moreover, the analysis reveals an intriguing observation: the three hybrid models (GraphRFI, GCMC, and GraphRec) appear to be more resilient to shilling attacks than the conventional collaborative filtering approaches (see RCF). We believe this is because the hybrid models adopt a more comprehensive strategy. By integrating supplementary information alongside the collaborative mechanisms, such as raw features, these models equip themselves with a broader context beyond the rating patterns.

4.7 A Case Study on Data Split Schemes

In the experiments we conducted, we consistently employed a random split as our primary method of partitioning the data, having determined its superiority empirically. However, in the interest of providing a more comprehensive understanding of alternative partitioning techniques, we also explored two widely recognized strategies within the recommender system domain [53], as outlined below:

1) *Leave Out Last:* This approach involves designating the final interaction of each user for testing, using the second-to-last interaction for validation, and using the remaining interactions for training.
2) *User Split:* The user split strategy, although less prevalent in the realm of recommender systems, holds sig-

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

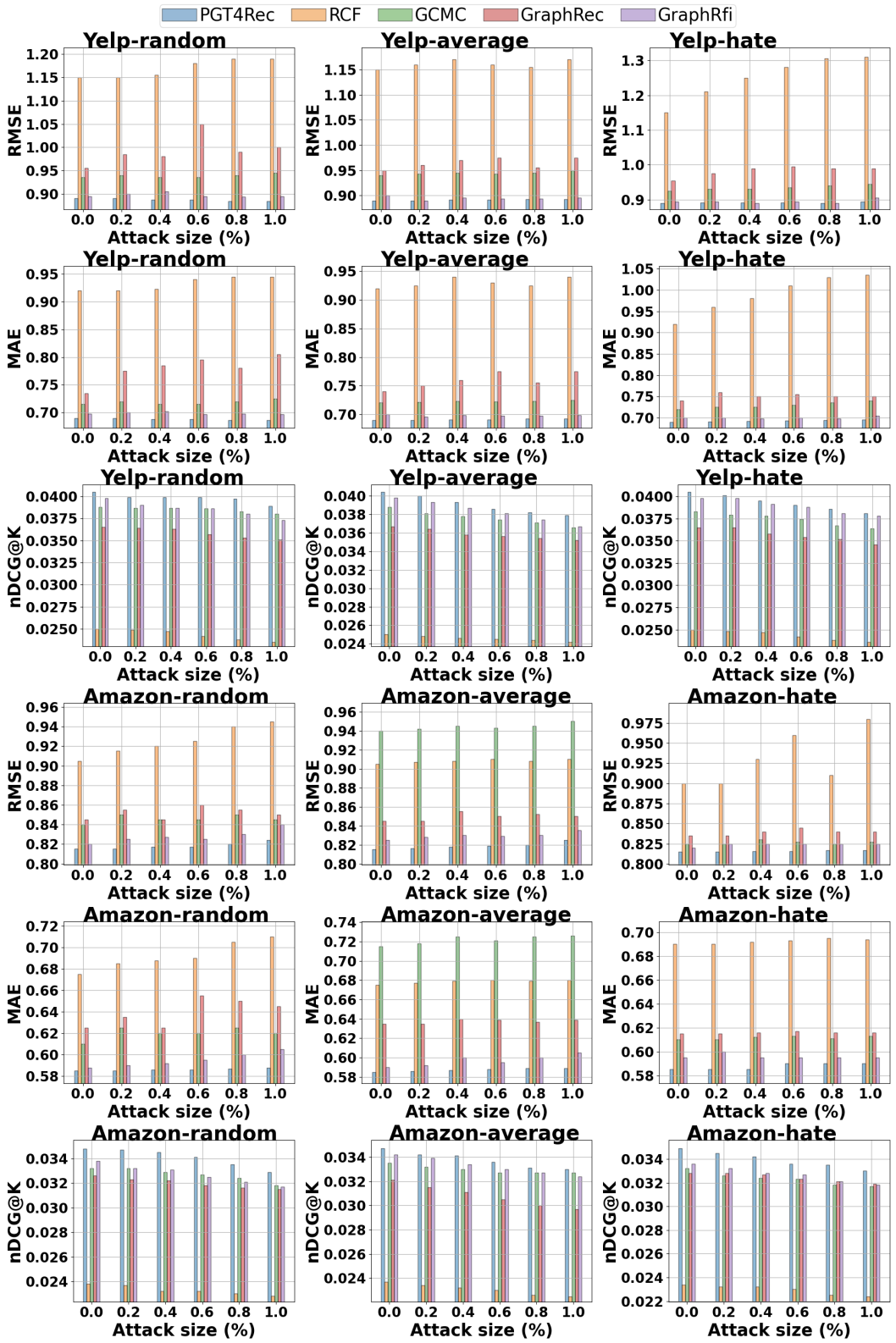


Fig. 7: Robustness against different types and sizes of shilling attacks.

nificance. This method entails reserving a specific set of users and their associated interactions for training, and using another set of users and their interactions for testing and validation.

The results are provided in Table 7 in terms of RMSE, MAE, and nDCG@K. The ratio of train/test set is still 80/20 as default. These results illustrate that the random splitting strategy consistently yields the most accurate rating predictions. This result can likely be attributed to the effectiveness of random splitting in capturing the inherent stochastic dependencies present in the interactions between users and items across the dataset. In the case of the Leave

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

TABLE 7: Rating prediction under different splitting strategies.

	Method	Leave One Last			User Split			Random Split		
		RMSE	MAE	nDCG@K	RMSE	MAE	nDCG@K	RMSE	MAE	nDCG@K
Yelp	RCF	1.31	0.97	0.0245	1.27	1.13	0.0218	1.19	0.95	0.0250
	GCMC	0.97	0.73	0.0364	1.12	0.8	0.0355	0.95	0.73	0.0384
	GraphRec	0.99	0.79	0.0328	1.04	0.88	0.0302	0.98	0.745	0.0357
	GraphRFI	0.97	0.76	0.0365	0.94	0.8	0.0387	0.91	0.71	0.0396
	PGT4Rec	0.93	0.74	0.0415	1.03	0.69	0.0396	0.846	0.677	0.0420
Amazon	RCF	0.99	0.72	0.0232	0.93	0.76	0.0198	0.923	0.69	0.0235
	GCMC	0.91	0.65	0.0329	0.96	0.74	0.0277	0.855	0.625	0.0332
	GraphRec	0.95	0.66	0.0304	0.96	0.65	0.0313	0.87	0.65	0.0323
	GraphRFI	0.91	0.64	0.0310	0.91	0.66	0.0305	0.84	0.6	0.0336
	PGT4Rec	0.78	0.6	0.0338	0.92	0.62	0.0308	0.775	0.576	0.0359

One Last strategy, we observed instances where certain items exhibit notably high performance. This occurrence could be attributed to the potential overfitting of the training process to specific pairs of interactions. Nonetheless, when considering overall performance, this strategy demonstrates a reasonable level of effectiveness. Conversely, the user split strategy is less common, with this approach delivering comparatively inferior performance across the board. Our tests show this strategy struggles to produce satisfactory results and does not generalize well.

4.8 Sensitivity Analysis

We conducted a comprehensive investigation into the influence of various hyperparameter settings. Specifically, we varied the balancing factor λ across a range of values, namely $\{1, 2, 3, 4, 5\}$, and explored different embedding dimensions, selecting from the set $\{32, 65, 128, 256, 512\}$. The results of these experiments are presented in Fig. 8 for λ and Figure 9 for the embedding dimension. The ratio of train/test set is still 80/20 as default.

Effect of λ hyper-parameter. Analysis of Fig. 8 reveals a noticeable pattern: the rating prediction task exhibits superior performance as λ increases from 1 to 3. However, further elevating λ to 4 and 5 results in a decline in the performance of the rating prediction task. This decline is likely attributed to the fact that $\lambda = 3$ strikes a harmonious balance between minimizing the rating prediction loss and the shilling attack detection loss. It becomes apparent that overly optimizing either of these loss functions leads to a drop in performance for both tasks. Importantly, this observed trend holds across both the Yelp and Amazon datasets. It could be explained by similar neighbourhood structure (e.g. homophily degree) between these datasets. We leave it as future work for optimising hyperparameters based on dataset statistics. On the other hand, *nDCG@K* is less sensitive to λ.

Effect of embedding dimension. For embedding dimension, we observed a distinct phenomenon in Fig. 9: the optimal configuration for the rating prediction task differs from that of shilling attack detection. More precisely, the rating prediction task gains advantages from a larger embedding dimension directly tied to the model's expressiveness. However, the overfitting induced by excess network parameters made it more challenging to identify shilling attacks as the dimension increased. Achieving a trade-off between embedding dimension and model complexity is crucial for attaining peak performance in both rating prediction and shilling attack detection tasks. On the other hand, *nDCG@K* is less sensitive to the embedding dimension.

4.9 End-to-End Efficiency

In this experiment, we assess the complete training performance of several graph-based methods involving neighbourhood aggregation across different datasets. The description of the baselines GCMC, GraphRFI, GraphRec, and H2-FDetector can be found in §4.1. The description of the variants *PA+RNN* and *PA+LSTM* of PGT4Rec can be found in §4.4. We do not include non-graph based methods for a fair comparison.

Table 8 presents the results in terms of training throughput to unify the comparison across different data sizes. It was found that PGT4Rec and its variants has the highest training throughput, essential for managing extensive multi-relational graphs. By decoupling message passing from representation learning, PGT4Rec maintains consistent training throughput regardless of the graph's structure across different datasets. The variants *PA+RNN* and *PA+LSTM* are faster probably because RNN or LSTM is faster than the transformer encoder of PGT4Rec; but they achieve worse accuracy results as in §4.4. On the other hand, the training throughput of the other graph-based methods is slower up to 98%. Detailed cost analysis can be found in §3.5.

TABLE 8: Comparison of training throughput (#samples/s.)

Methods	Yelp	Amazon
PGT4Rec	17480.25	13862.98
PA+RNN	17917.13	14647.56
PA+LSTM	18229.97	14630.97
GCMC	10867.69	2468.53
GraphRFI	7789.01	1567.65
GraphRec	13716.51	2780.69
H2-FDetetor	9120.63	272.33

5 Related Work

Robust Recommender Systems. While there is long history of research on recommender systems [54], their robustness against attacks has been studied only recently. Robust recommender systems rely on adversarial training [55], [56], [56], which adds small perturbations to the inputs such that the model learns to adapt to untruthful ratings. Other work studies the causal effects of user interactions [57] – the aim being to find unbiased ranking metrics, largely based on the differences between ratings. However, such approaches are orthogonal to our work as they mostly focus on analysing causality. Also, simply looking for possible attacks, like poisoning attacks [58], [59], [60] or shilling attacks [7] might indirectly protect a recommendation system.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

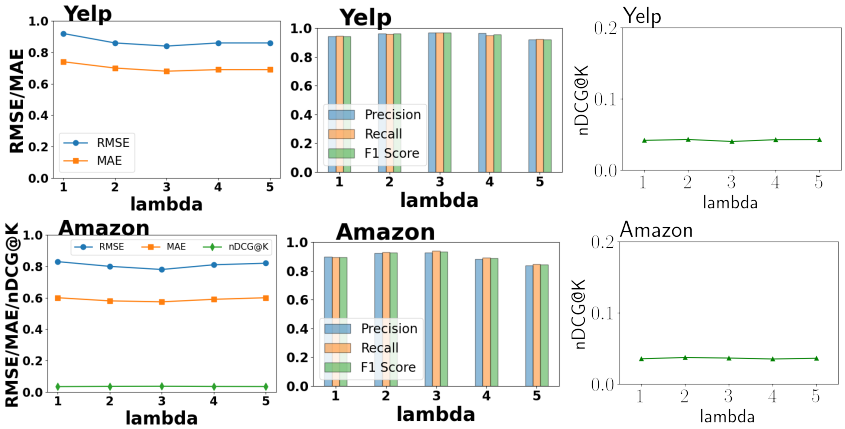


Fig. 8: Effects of the λ parameter.

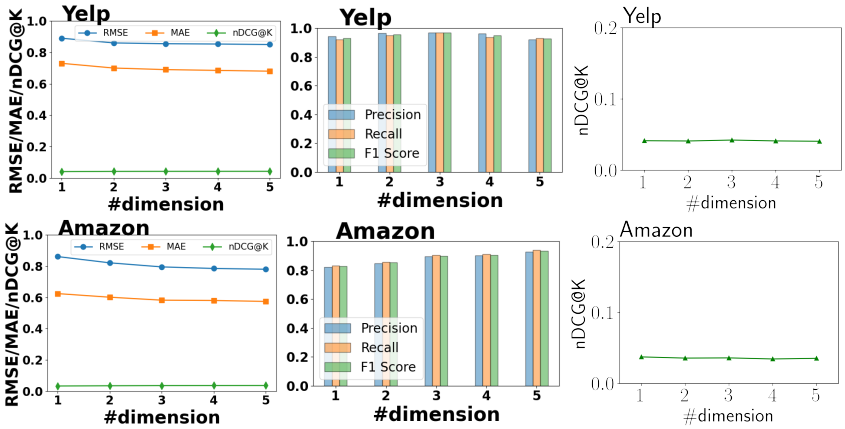


Fig. 9: Effects of the dimension parameter.

However, while GNNs represent the current state-of-the-art in recommender systems [1], their robustness has not yet been studied extensively. In this paper, we show how to handle low homophily in GNN-based recommender systems. We chose to demonstrate our approach in the context of shilling attacks because this is a common reason for low homophily, but our model may be lifted to any user classification. This is because our model accepts any labelling in which a user $u \in \hat{\mathcal{U}}$ has a scalar label $y_u \in \{1, \ldots, C\}$.

Shilling Attack Detection. The world is rife with incentives to conduct a shilling attack [5], [42]. Prior approaches to detecting them tend to be based on either unsupervised training, supervised training, or graph-based training [6], [36], [41], [42]. Similar to work on the attacks [7], Yet studies on detecting shilling attacks are typically undertaken in isolation, making it difficult to draw insights into how well a system performs overall when shilling attacks are effectively being thwarted.

In this paper, we formulated a multi-task learning problem that combines ratings prediction with shilling attack detection. While GraphRFI [6] follows a similar idea, it relies on a heavy architecture of GCN models and neural random forests and, as shown empirically, it did not perform as well as `PGT4Rec` in scenarios with low homophily.

6 Discussions

Ethics. Recommender systems often struggle to strike a balance between recommending items that are similar to what a user has already liked (homophily) and suggesting new and diverse items that the user might not have encountered before (heterophily). While homophily might lead to recommendations based solely on a user's past preferences, this could create a "filter bubble" effect [61], limiting users to a narrow set of choices. By contrast, introducing a degree of heterophily into recommendations can lead to more serendipitous discoveries and help users explore a broader range of options. Many GNN and graph-based models for recommender systems assume high homophily, causing biases and ethical issues when they are used on networks that do not exactly follow this rule [8]. Moreover, shilling attacks are often a prevalent cause of low homophily [62].

Our work aims to mitigate these drawbacks. Our methodology does not alter the homophily levels in recommender systems. Instead, we go beyond the traditional GNN models by designing a new neighbourhood aggregation scheme. We hope that the techniques outlined in this paper will improve the trustworthiness and overall performance of recommendation systems. One potential issue with our approach is the potential to mislabel a genuine user as malicious (false positive) and exact a consequence as a result – for example, suspending their account. However, this could be avoided if `PGT4Rec`'s outcomes were used solely as flags for a manual follow up by a human administrator.

Data Splitting in Recommendation Evaluation. Among the various methods for evaluating recommendation systems, the most prevalent is the "offline" evaluation, which relies on historical ratings of items and/or implicit item feed-

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

back [63], [64]. This approach hinges on a dataset containing past explicit or implicit user interactions, and, since contemporary models rely on supervised learning, this dataset must be divided into distinct training, validation, and testing sets. In general, there are four main data splitting strategies [53]. (1) *Leave One Last*, which involves isolating the last user transaction for testing purposes. Typically, the second-to-last transaction per user is allocated to validation, and the remaining transactions are used for training. (2) *Temporal Split*. This method divides historical interactions into groups based on timestamps and using a certain percentage for testing purposes, such as the last 20% of interactions. (3) *Random Split* involves randomly selecting the training/test boundary on a per user basis [6], [65]. Last, (4) *User Split* which divides data by user instead of interaction – a strategy that is not particularly common. It assigns specific users and transactions to the training set while using a different users and their transactions for testing. This approach is uncommon due to its requirement for models to recommend items to new users, a feature many methods lack.

Our work focuses on neighbouring interactions, which means random splits work well enough, as is the case with many other studies [6], [63]. Other splitting strategies, such as temporal split, would require us to add temporal information, which is not always available. However, one way to extend `PGT4Rec` across the time dimension would be to perform partitioned aggregation in each time step and, of course, design an additional embedding to capture the temporal aspects of the data [66]. Notwithstanding this idea for future work, the case study we performed with different data splitting schemes did demonstrate our model to be robust. So while we acknowledge that further in-depth analyses are required [67], we feel these are out of the scope of this particular paper.

Biases. Bias in recommender systems is a multifaceted issue that significantly impacts user interaction and the accuracy of recommendations [10], [68], [69]. Research in this domain has uncovered various patterns, such as the trend of increasing or decreasing average ratings and the J-shaped distribution of ratings, which suggest that user behavior and rating tendencies are influenced by underlying biases [70]. These biases not only skew the data on which recommender systems are trained but also lead to a lack of representation and fairness in the recommendations provided. Previous studies have explored biases related to ratings and conformity [71]. For instance, the choice-supportive bias is relevant in recommender systems with reviews, and the Matthew effect focuses on group dynamics in recommendations [72]. Furthermore, models that account for missing data as not random (MNAR models) attempt to relate missing ratings (to be predicted) to various factors such as endogeneous bias and silent minority [10]. One possible solution is using diversification for recommender systems to satisfy different user preferences at the same time by creating a diverse recommendation list [73]. However, this research direction is orthogonal to our work as we focus on identify the influence of malicious users from shilling attacks, who intentionally promote or demote certain items.

7 Conclusion

In this paper, we proposed `PGT4Rec`, a model for recommender systems that jointly optimizes ratings predictions and shilling attack detection. Identifying malicious users discourages large shifts in the ratings prediction process, while accurate predictions concurrently help the model to detect shilling attacks. `PGT4Rec` caters to low homophily with a partitioned aggregation method that exploits class labels and produces three types of embeddings as inputs to a Transformer encoder. The results of our experiments demonstrate `PGT4Rec` to be highly effective, outshining many existing techniques.

In future work, we intend to consider different types of interactions between users and items in addition to rating, such as likes, reviews, views, and purchases [74] using relation-aware GNNs [22]. We may also expand our model to perform item classifications, which would provide the opportunity to analyse the items targeted in shilling attacks [23]. Another way to extend `PGT4Rec` would be to incorporate causality analysis [57] by performing a causal intervention on the user-item graph first before applying our method. We will also consider more classes of users such as hardcore users and minority users [10], as well as include contextual information [48]. Applying our method to other types of recommender systems based on Language Models (LLMs) and Retrieval-Augmented Generative Models (RAGs) [75] is also a trending direction.

Acknowledgments

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE200101465) and ARC DP Project (Grant No. DP240101108).

References

[1] S. Gu, X. Wang, C. Shi, and D. Xiao, "Self-supervised graph neural networks for multi-behavior recommendation," in *IJCAI*, 2022, pp. 2052–2058.

[2] G. Guo, E. Yang, L. Shen, X. Yang, and X. He, "Discrete trust-aware matrix factorization for fast recommendation." in *IJCAI*, 2019, pp. 1380–1386.

[3] Z. Wu, C. Li, J. Cao, and Y. Ge, "On scalability of association-rule-based recommendation: A unified distributed-computing framework," *TWEB*, vol. 14, no. 3, pp. 1–21, 2020.

[4] Z. Chen, J. Wu, C. Li, J. Chen, R. Xiao, and B. Zhao, "Co-training disentangled domain adaptation network for leveraging popularity bias in recommenders," in *SIGIR*, 2022, pp. 60–69.

[5] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *CSUR*, vol. 55, no. 5, pp. 1–37, 2022.

[6] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," in *SIGIR*, 2020, pp. 689–698.

[7] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: a review," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 291–319, 2020.

[8] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[9] W. Cai, W. Ma, J. Zhan, and Y. Jiang, "Entity alignment with reliable path reasoning and relation-aware heterogeneous graph transformer," in *IJCAI*, 2022, pp. 1930–1937.

[10] D. Liu, C. Lin, Z. Zhang, Y. Xiao, and H. Tong, "Spiral of silence in recommender systems," in *WSDM*, 2019, pp. 222–230.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

[11] O. Platonov, D. Kuznedelev, A. Babenko, and L. Prokhorenkova, "Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond," *NeurIPS*, vol. 36, 2024.

[12] J. Liu, M. He, G. Wang, Q. V. H. Nguyen, X. Shang, and H. Yin, "Imbalanced node classification beyond homophilic assumption," in *IJCAI*, 2023, pp. 7206–7214.

[13] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, 2019, pp. 21–29.

[14] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup, "Revisiting heterophily for graph neural networks," *NeurIPS*, vol. 35, pp. 1362–1375, 2022.

[15] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," *NeurIPS*, vol. 34, pp. 20 887–20 902, 2021.

[16] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *NeurIPS*, vol. 33, pp. 7793–7804, 2020.

[17] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2020.

[18] E. Zangerle and C. Bauer, "Evaluating recommender systems: survey and framework," *CSUR*, vol. 55, no. 8, pp. 1–38, 2022.

[19] Y. Ge, S. Liu, Z. Fu, J. Tan, Z. Li, S. Xu, Y. Li, Y. Xian, and Y. Zhang, "A survey on trustworthy recommender systems," *ACM Transactions on Recommender Systems*, 2022.

[20] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 42, pp. 767–799, 2014.

[21] T. T. Nguyen, N. Quoc Viet Hung, T. T. Nguyen, T. T. Huynh, T. T. Nguyen, M. Weidlich, and H. Yin, "Manipulating recommender systems: A survey of poisoning attacks and countermeasures," *CSUR*, 2024.

[22] Y. Wang, J. Zhang, Z. Huang, W. Li, S. Feng, Z. Ma, Y. Sun, D. Yu, F. Dong, J. Jin *et al.*, "Label information enhanced fraud detection against low homophily in graphs," in *TheWebConf*, 2023, pp. 406–416.

[23] Z. Yang, Q. Sun, Y. Zhang, and W. Wang, "Identification of malicious injection attacks in dense rating and co-visitation behaviors," *TIFS*, vol. 16, pp. 537–552, 2020.

[24] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," *arXiv preprint arXiv:2009.03509*, 2020.

[25] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *NeurIPS*, vol. 30, 2017.

[27] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *CIKM*, 2020, pp. 315–324.

[28] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and choose: a gnn-based imbalanced learning approach for fraud detection," in *TheWebConf*, 2021, pp. 3168–3177.

[29] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *SIGIR*, 2020, pp. 1569–1572.

[30] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?" in *ICWSM*, vol. 7, no. 1, 2013, pp. 409–418.

[31] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *KDD*, 2015, pp. 985–994.

[32] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *CIKM*, 2019, pp. 2703–2711.

[33] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *TheWebConf*, 2013, pp. 897–908.

[34] A. Fayazi, K. Lee, J. Caverlee, and A. Squicciarini, "Uncovering crowdsourced manipulation of online reviews," in *SIGIR*, 2015, pp. 233–242.

[35] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *WSDM*, 2018, pp. 333–341.

[36] F. Shi, Y. Cao, Y. Shang, Y. Zhou, C. Zhou, and J. Wu, "H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections," in *TheWebConf*, 2022, pp. 1486–1494.

[37] A. Barrat, M. Barthélemy, and A. Vespignani, "Modeling the evolution of weighted networks," *Physical review E*, vol. 70, no. 6, p. 066149, 2004.

[38] B. Mehta, T. Hofmann, and W. Nejdl, "Robust collaborative filtering," in *RecSys*, 2007, pp. 49–56.

[39] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[40] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *TheWebConf*, 2019, pp. 417–426.

[41] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *UMUAI*, vol. 19, no. 1, pp. 65–97, 2009.

[42] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "Svm-tia a shilling attack detection method based on svm and target item analysis in recommender systems," *Neurocomputing*, vol. 210, pp. 197–205, 2016.

[43] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *TOIS*, vol. 40, no. 4, pp. 1–46, 2021.

[44] G. Zhang, J. Wu, J. Yang, A. Beheshti, S. Xue, C. Zhou, and Q. Z. Sheng, "Fraudre: fraud detection dual-resistant to graph inconsistency and imbalance," in *ICDM*, 2021, pp. 867–876.

[45] T. Yang, Y. Wang, Z. Yue, Y. Yang, Y. Tong, and J. Bai, "Graph pointer neural networks," in *AAAI*, vol. 36, no. 8, 2022, pp. 8832–8839.

[46] T. Wang, D. Jin, R. Wang, D. He, and Y. Huang, "Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily," in *AAAI*, vol. 36, no. 4, 2022, pp. 4210–4218.

[47] D. He, C. Liang, H. Liu, M. Wen, P. Jiao, and Z. Feng, "Block modeling-guided graph convolutionafl neural networks," in *AAAI*, vol. 36, no. 4, 2022, pp. 4022–4029.

[48] S. Raza and C. Ding, "Progress in context-aware recommender systems—an overview," *CSR*, vol. 31, pp. 84–97, 2019.

[49] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *AAAI*, vol. 33, no. 01, 2019, pp. 7370–7377.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[51] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *NeurIPS*, vol. 32, 2019.

[52] C. C. Aggarwal, "Attack-resistant recommender systems," in *Recommender Systems*, 2016, pp. 385–410.

[53] Z. Meng, R. McCreadie, C. Macdonald, and I. Ounis, "Exploring data splitting strategies for the evaluation of recommendation models," in *RecSys*, 2020, pp. 681–686.

[54] P. Baltescu, H. Chen, N. Pancha, A. Zhai, J. Leskovec, and C. Rosenberg, "Itemsage: Learning product embeddings for shopping recommendations at pinterest," in *KDD*, 2022, pp. 2703–2711.

[55] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, and S. Yuan, "Fight fire with fire: Towards robust recommender systems via adversarial poisoning training," in *SIGIR*, 2021, pp. 1074–1083.

[56] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua, "Adversarial training towards robust multimedia recommender system," *TKDE*, vol. 32, no. 5, pp. 855–867, 2019.

[57] T. Xiao and S. Wang, "Towards unbiased and robust causal ranking for recommender systems," in *WSDM*, 2022, pp. 1158–1167.

[58] J. Song, Z. Li, Z. Hu, Y. Wu, Z. Li, J. Li, and J. Gao, "Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems," in *ICDE*, 2020, pp. 157–168.

[59] H. Zhang, Y. Li, B. Ding, and J. Gao, "Practical data poisoning attack against next-item recommendation," in *TheWebConf*, 2020, pp. 2458–2464.

[60] T. T. Nguyen, K. N. D. Quach, T. T. Nguyen, T. T. Huynh, V. H. Vu, P. Le Nguyen, J. Jo, and Q. V. H. Nguyen, "Poisoning gnn-based recommender systems with generative surrogate-based attacks," *TOIS*, 2022.

[61] Y. Xu, E. Wang, Y. Yang, and H. Xiong, "Gs^2-rs: A generative approach for alleviating cold start and filter bubbles in recommender systems," *TKDE*, 2023.

[62] H. Cai and F. Zhang, "Bs-sc: An unsupervised approach for detecting shilling profiles in collaborative recommender systems," *TKDE*, vol. 33, no. 4, pp. 1375–1388, 2019.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2024.3485880

[63] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," *TKDE*, vol. 35, no. 5, pp. 4425–4445, 2022.
[64] P. Li and A. Tuzhilin, "Dual metric learning for effective and efficient cross-domain recommendations," *TKDE*, vol. 35, no. 1, pp. 321–334, 2021.
[65] T. Zhu, L. Sun, and G. Chen, "Embedding disentanglement in graph convolutional networks for recommendation," *TKDE*, vol. 35, no. 1, pp. 431–442, 2021.
[66] J.-H. Chiang, C.-Y. Ma, C.-S. Wang, and P.-Y. Hao, "An adaptive, context-aware, and stacked attention network-based recommendation system to capture users' temporal preference," *TKDE*, vol. 35, no. 4, pp. 3404–3418, 2022.
[67] A. Sun, "Take a fresh look at recommender systems from an evaluation standpoint," in *SIGIR*, 2023, pp. 2629–2638.
[68] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *TKDE*, 2023.
[69] D. Liu, P. Cheng, H. Zhu, Z. Dong, X. He, W. Pan, and Z. Ming, "Debiased representation learning in recommendation via information bottleneck," *TORS*, vol. 1, no. 1, pp. 1–27, 2023.
[70] Y. Liu, X. Cao, and Y. Yu, "Are you influenced by others when rating? improve rating prediction by conformity modeling," in *RecSys*, 2016, pp. 269–272.
[71] H. Yang, G. Ling, Y. Su, M. R. Lyu, and I. King, "Boosting response aware model-based collaborative filtering," *TKDE*, vol. 27, no. 8, pp. 2064–2077, 2015.
[72] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, "Modeling user exposure in recommendation," in *TheWebConf*, 2016, pp. 951–961.
[73] M. Kunaver and T. Požrl, "Diversity in recommender systems–a survey," *KBS*, vol. 123, pp. 154–162, 2017.
[74] Y. Lyu, H. Yin, J. Liu, M. Liu, H. Liu, and S. Deng, "Reliable recommendation with review-level explanations," in *ICDE*, 2021, pp. 1548–1558.
[75] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *AAAI*, 2024, pp. 1–9.

Nguyen Thanh Tam is a lecturer at Griffith University (Australia). He earned his PhD from EPFL (Switzerland). His research interests include Databases and AI for data networks and data streams. He publishes at world-leading conferences and top-tier journals such as SIGMOD, VLDB, WSDM, ICDE, IJCAI, SIGIR, TKDE, JVLDB, and TIST. He is a guest editor of IEEE Journal of Biomedical and Health Informatics, an area chair of ACL and EMNLP.

Nguyen Thanh Toan received the PhD degree at Griffith University, Australia. He received the Master degree and Bachelor degree from Ho Chi Minh City University of Technology. His main research interests include the areas of database systems, machine learning, decision support systems, and software design. He has publications in prestigious journals and conferences such as TIST, Information Sciences, ESWA, KBS, TOIS, and IJCNN.

Matthias Weidlich is professor and Chair of Databases and Information Systems at the Department of Computer Science at Humboldt-Universität zu Berlin. His research focuses on process-oriented and event-based information systems, and his results appear regularly in premier conferences (SIGMOD, VLDB, ICDE) and journals (TKDE, Inf. Sys., VLDBJ) in the field. He serves as EiC for the Information Systems journal and on the ACM DEBS Steering Committee.

Jun Jo is an A/Prof at Griffith University. He received his PhD from the University of Sydney. His research includes computer vision, machine learning, robotics, autonomous cars, drones, IoTs, satellite image analysis and medical image analysis. He has published over 150 refereed publications. Dr Jo is currently taking the position of the President of Australian Robotics Association (ARA). He is also chairing the World Innovative Technology (WIT) organisation.

Nguyen Quoc Viet Hung is an A/Prof at Griffith University, Australia. He earned his PhD degree from EPFL (Switzerland). His research focuses on data integration, data quality, information retrieval, trust management, recommender systems, machine learning and big data visualisation. He published several papers in top-tier venues such as SIGMOD, VLDB, WSDM, WWW, SIGIR, KDD, AAAI, ICDE, IJCAI, JVLDB, TKDE, TOIS, and TIST.

Hongzhi Yin received the PhD degree in computer science from Peking University, in 2014. He is a full professor and director of the Responsible Big Data Intelligence Lab (RBDI) at the University of Queensland. He is an ARC Future Fellow and an ARC DECRA Fellow. His research interests include recommendation system, user profiling, topic models, deeplearning, social media mining, and location-based services.

Alan Wee-Chung Liew is a full professor at Griffith University, Australia. Prior to joining Griffith University in 2007, he was an Assistant Professor at the Department of Computer Science and Engineering, Chinese University of Hong Kong. He is currently serving as an associate editor of IEEE Trans. Fuzzy Systems, Springer Nature Computer Science, and International Journal of Computational Intelligence Systems, Machine Intelligence Research. He is a senior member of IEEE since 2005.

Authorized licensed use limited to: Griffith University. Downloaded on November 11,2024 at 23:56:21 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.