ABSTRACT

Fake reviews are prevalent on review websites such as Amazon and Yelp. GNN is the state-of-the-art method that can detect suspicious reviewers by exploiting the topologies of the graph connecting reviewers, reviews, and target products. However, the discrepancy in the detection accuracy across different groups of reviewers causes discriminative treatment of different reviewers of the websites, leading to less engagement and trustworthiness of such websites. The complex dependencies over the review graph introduce difficulties in teasing out subgroups of reviewers that are hidden within larger groups and are treated unfairly. There is no previous study that defines and discovers the subtle subgroups to improve equitable treatment of reviewers. In this paper, we address the challenges of defining, discovering, and utilizing subgroup memberships for fair spam detection. We first define a subgroup membership that can lead to discrepant accuracy in the subgroups. Since the subgroup membership is usually not observable while also important to guide the GNN detector to balance the treatment, we design a model that jointly infers the hidden subgroup memberships and exploits the membership for calibrating the target GNN’s detection accuracy across subgroups. Comprehensive results on two large Yelp review datasets demonstrate that the proposed model can be trained to treat the subgroups more fairly.

1 INTRODUCTION

Graphs are widely used in various applications, such as crime forecasting system [18, 38], recommendation systems [42, 43], social networks, and spam detection [15, 26]. For those non-Euclidean data, Graph Neural Networks (GNNs) are powerful architectures for graph representation and learning. These applications are involved in our day-to-day decision-making; it will harm society if there exists any strong bias in GNN’s output. Therefore, in addition to improving the accuracy, there is growing interest in enhancing fairness in GNNs [1, 13, 25, 29, 35]. Unfortunately, those specific spam detection works [14, 15, 24, 37, 42] focus exclusively on the accuracy of the fraud detectors and ignore the model fairness. Therefore, we study the fairness of GNN in the background of the spam detection task on the review graphs [28, 33] that contain reviewers, reviews, and products nodes.

Generally, fairness issues on the graph are raised in two perspectives: (a) bias from the node feature [19, 32] (b) topology bias coming from the edge connections [9, 17, 25, 35]. In Figure 1, we give an example of a review graph where the edges describe that users leave reviews for products. Ideally, given the review graph $G$, the detector infers the suspicion for reviews and assigns higher suspicion for spams than non-spams based on their review features. In practice, restricted by the anonymity of the spammers, we have no access to the traits of users, such as gender, age, and race. Therefore, it is unrealistic to divide nodes into various groups based on node features and enforce the statistical parity on each group.

Since our spam detection task is under the transductive setting, node degrees are observable for the training and test set, representing the number of reviews associated with the users in our graph. For instance, previous work [5] reveals that users or products with fewer reviews can have a higher chance of being screened by detectors (GNN). This unfair situation is caused by the long-tail distribution of the user node degree. According to the computation graphs (see Figure 1 right side), if we only compare spams $Y_2$ (posted by a high-degree user) and $Y_1$ (posted by a low-degree user), the spam information of $Y_2$ will be diluted by other non-spam reviews as message passing from bottom to up in aggregation operation for GNN. Therefore, old businesses can reduce their suspicion and evade detection by easily hiding the latest spam reviews in many of their past non-spam reviews, which is unfair to most new businesses with few reviews. Likewise, in this work, we stipulate the “protected group” users (denoted by the sensitive attribute $A = 1$) who post fewer reviews than a certain threshold and the “favored group” users ($A = 0$) for the rest of the reviewers.

Unfortunately, unfairness exists inside the “favored” groups where the detector treats users from the “favored group” vastly differently. Based on the analyses above, the crux of reducing suspicion depends on whether one user can weaken its spam signal by aggregating signals from other genuine reviews posted by itself. For example, in Figure 1, reviews posted by two high-degree users $U_1$ (posts spam and non-spams; named as “mixed” user) and $U_2$ (posts spam only; named as “pure” user) will be treated differently. Since $U_2$ posts spam reviews only, no matter which spam is calculated at root, GNN will not downgrade its suspicion. On the other hand, for $U_1$, the detector will be deceived by those non-spams. Therefore, to achieve higher accuracy on group $A = 0$, GNN will target “pure” users like $U_2$. In other words, if we force the detector to be fair with respect to sensitive attribute $A$ only, the fairness between...
mixed and pure users suffers neglect. This paradox is the so-called “Simpson’s Paradox” raised by the aggregation bias that we wrongly assumed the detector assigns all users from the “favored” group a low suspicion, which neglects the user discrepancy. Thus, it is critical to divide the group of high-degree users (\( A = 0 \)) into subgroups: “mixed users” and “pure users”, then maintain group fairness and subgroup fairness both at the same time. There are several challenges to solve this problem:

**Define the subgroup correctly on the graph.** Most of the recent works [20, 21, 36] study the combination of different sensitive attributes which are observable and can divide the dataset into various groups. Others [2, 7, 30] work on the problem when sensitive attribute is unknown or contains noisy information. In [10], they built probabilistic models to approximate the true sensitive attribute using proxy information, such as inferring race from the last name and geolocation. All these methods utilize I.D. vector data whose sensitive attribute is already defined, such as race and gender. Unlike the above works, we handle graph data where proxies are not observed. Besides, we are looking at some sensitive attributes related to the ground truth label and specific to spam detection on graphs that have not been formally defined.

**Infer unknown subgroup membership.** We would like to improve subgroup fairness inside favored group \( A = 0 \) between “pure user” such as \( U_0 \) who posts all the spam reviews (denoted as \( A' = 0 \)) and “mixed user” such as \( U_1 \) who posts both spam and non-spam reviews (denoted as \( A' = 1 \)). In fact, we obtain the precise sensitive attribute \( A' \) for each user only after fully observing the label of all the reviews, which is unattainable for test users. Therefore, we have to infer subgroup membership for the test set. It is forthwith another problem: it is hard to identify the subgroup and give them special treatment without enough data about the subgroup users within the favored group. Prior researchers had done lots of works on data augmentation for graph data, such as GraphSomer [46], GraphMixup [40, 41], GraphCrop [39], which expanded the training set by generating synthetic data. However, most data augmentation strategies focus on mitigating the class-imbalance issue but are not designed to provide data to help detect subgroups. Meanwhile, implementing all these data augmentation methods requires knowing which class is in the minority and needs to be expanded. Therefore, we can only augment data for minority subgroups after inferring the subgroup membership.

**Maintain subgroup fairness with group fairness.** Improving the subgroup fairness after inferring the subgroup membership is another challenge. Most of the related works [20] formulated the optimization problem with multiple fairness constraints for each subgroup combination. In [36], it incorporates multiple classifiers and one for each subgroup. However, when the subgroup needs to be inferred, e.g., subgroup membership is probabilistic rather than deterministic, the prior optimization formulation or model designs such as introducing fairness regularizer are not applicable.

To solve the above challenges, we define the subgroup member according to the property of our review graph. This new sensitive attribute \( A' \) for user node reflects the label distribution of reviews posted by each user from group \( A = 0, \) i.e., a high-degree user posts either single-class reviews (all genuine or all fake) or mixed-class reviews (both fake and genuine). Since when one user posts reviews with the same labels, no matter which reviews is at the root of the computation graph, GNN’s detection will not be affected by aggregating other reviews’ messages. Then, we have the true sensitive attribute \( A' \) for the training users by accessing the ground truth labels for their associated reviews. Next, We infer the unknown subgroup memberships for the test users using the data augmentation method. By duplicating subgroup users with their reviews from the training set then pruning part of their associated spams, we synthesize nodes to enlarge the size of the minority subgroup. Rather than introducing more fairness constraints for subgroups data, we directly treat the new sensitive attribute as an indicator variable appending to the node feature. Finally, we construct a joint-training framework which adopts two GNNs: the first GNN \( f_W(\cdot) \) is to infer indicator variables while the second GNN \( g_\theta(\cdot) \) utilizes the output of \( f_W(\cdot) \) and classifies the node class.

## 2 PRELIMINARIES

### 2.1 Spam detection based on GNN

We study the spam detection on the review-graph defined as \( G = (V, E) \), where \( V = \{v_1, \ldots, v_N\} \) denotes the set of nodes and \( E \subseteq V \times V \) represents the set of undirected edges. Each node \( v_i \in V \) has a feature vector \( x_i \), where the subscript is the node index.

There are three types of nodes in \( G \), i.e., user, review, and product, respectively, and each node can be of only one of the three types. We denote the subsets of each type of node as \( V^U, V^R, V^P \subseteq V \). The neighbor of node \( v_i \) is represented as \( N(i) = \{v_j \in V | e_{ij} \in E\} \).

GNN [22] is the state-of-the-art method for node prediction task containing multiple layers. For the GNN classifier \( f_W(\cdot) \), let \( h^{(l)}_i \) be the learned representation of node \( v_i \) at layer \( l \), where \( l = 1, \ldots, L \). \( h^{(l)}_i \) is calculated from the message-passing as follows:

\[
    h^{(l)}_i = \text{COMBINE}\left(h^{(l-1)}_j, \text{AGGREGATE}\left(\left\{h^{(l-1)}_j \mid j \in N(i)\right\}\right)\right)
\]

where AGGREGATE and COMBINE are two functions that can take various forms. In this work, AGGREGATE computes the mean of the input vectors, and COMBINE consists of the ReLU and an affine mapping with parameters \( W \). The input vector \( x_i = h^{(0)}_i \) is treated as the representation at layer 0. Let \( \hat{y}_i \) be the prediction for node \( v_i \) given by GNN, where \( f_W(x_i) = \hat{y}_i = \text{Sigmoid}(h^{(L)}_i) \). We minimize the cross-entropy loss for the training node set \( V^T \) :

\[
    L(W; G) = -\frac{1}{|V^T|} \sum_{v_i \in V^T} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \quad (1)
\]

where \( y_i \in \{0, 1\} \) is the target label for node \( v_i \in V^U \cup V^R \). We list the main notations in Table 1.

### 2.2 Fairness regularizer

**Group.** Based on the node degrees (sensitive attribute \( A \)), we split user nodes \( V^U \) into “protected group” \( V^U_1 \) whose degree is smaller than the 95-th percentile of all the user nodes’ degree, and “favored group” \( V^U_0 \) for the remaining users. The subscript denotes the value of the sensitive attribute. Then, we divide the review nodes \( V^R \) into \( V^R_1 \) and \( V^R_0 \) following the group of their associated users, i.e., the user and its associated reviews have the same value of \( A \). The
Table 1: Notations and definitions.

| Notations | Definitions |
|-----------|-------------|
| \( G \)   | Review graph |
| \( V \)   | Nodes of graph \( G \) |
| \( E \)   | Edges of graph \( G \) |
| \( x_i, y_i \) | Feature and label of node \( v_i \) |
| \( N(i) \) | Set of direct neighbors of \( v_i \) |
| \( |V| \) | Cardinality of a set \( V \) |
| \( \mathcal{V}^T, \mathcal{V}^E \) | Training nodes, test nodes |
| \( \mathcal{V}^U, \mathcal{V}^R, \mathcal{V}^P \) | User, review, and product nodes |
| \( A, A' \) | Binary sensitive attributes |
| \( \mathcal{V}^R, \mathcal{V}^U_A \) | The set of review, user nodes with \( A \in \{0, 1\} \) |
| \( \mathcal{V}^R_A \) | The set of user nodes with attributes \( A, A' \) |

\[
\begin{align*}
\mathcal{L}(\theta; G) &= -\frac{1}{|\mathcal{V}|} \sum_{i,j:y_{ij} \leq \tilde{y}_j} \log \left(1 + \exp\left(h^{(L)}_i - h^{(L)}_j\right)\right) \\
\mathcal{L}_{\text{GNN}}(W; G) &= \mathcal{L}(W; G) + \lambda \cdot R_{\text{fair}}(W; G)
\end{align*}
\]

where \( \lambda \) is the coefficient of the regularization term.

### 3 METHODOLOGY

#### 3.1 Subgroup definition and selection

**Subgroups.** We let the sensitive attribute \( A \) indicate the “favored” \( (A = 1) \) and “protected” \( (A = 0) \) groups based on node degree. For users who post a large number of reviews \((A = 0)\), GNN can still treat individual users differently based on the label distribution of their posted reviews. In other words, the GNN model can be biased by the neighborhood label distribution. In particular, due to the aggregation operator of GNN, the suspiciousness of a spam review of the user \( v_i \) will be reduced by other non-spams posted by \( v_i \). We use an additional sensitive attribute \( A' \) to identify whether one user will be “favored” due to its non-spams reviews:

\[
A' = \begin{cases} 
1 & \text{if } \alpha |N(j)| < \sum_{j \in N(i)} y_j < \beta |N(j)| \\
0 & \text{otherwise}
\end{cases}
\]

where \( v_i \in \mathcal{V}_{0,1}^U \) and \( N(i) \) are reviews posted by \( v_i \) and \( 0 < \alpha < \beta < 1 \). When \( A' = 1 \), it represents a user who posts both fake \((y = 1)\) and genuine \((y = 0)\) reviews. \( A' = 0 \) indicates that a user posts multiple reviews belonging to either the fake or genuine class but not both.

The unfairness issue among high-degree users \( \mathcal{V}_{0,1}^U \) lead to the split of \( \mathcal{V}_{0,1}^U \) into \( \mathcal{V}_{0,0}^U \) and \( \mathcal{V}_{0,1}^U \) where the first and second term in the subscript represents the value of \( A \) and \( A' \), respectively. Almost all users from \( \mathcal{V}_{0,1}^U \) post reviews in just one class; thus, the sensitive attribute \( A' \) is not relevant.

**Infer unknown subgroup membership.** We can determine the value of the subgroup indicator \( A' \) for users whose reviews are known to be spam or not. In practice, most of the review labels are unknown, while we hypothesize that inferring \( A' \) for all user nodes within the larger group \( A = 0 \) will help resolve the subgroup unfairness, as GNN can use the inferred attribute values to strike a more equitable treatment of the two subgroups. Therefore, we introduce a second GNN model \( g_{\theta}(\cdot) \) that helps infer \( A' \) for users whose reviews are not fully labeled. In principle, we can use any predictive model to map \( v_i \) to \( A' \), but we choose GNN due to its capability of modeling neighborhood data distribution that can be helpful for the inference. Let \( \tilde{A}'_i \) be the predicted \( A' \) for user \( v_i \in \mathcal{V}^U \). The loss of \( g_{\theta} \) becomes:

\[
\mathcal{L}(\theta; G) = -\frac{1}{|\mathcal{V}^E|} \sum_{v_i \in \mathcal{V}^E} A'_i \log \Pr(\tilde{A}'_i = 1\mid G, \theta)
\]

where \( A'_i \in \{0, 1\} \) is the ground truth sensitive value of attribute \( A' \) for those user nodes whose reviews are fully labeled. However, this elementary model is not sufficient: on a review graph, very few user nodes are labeled as \( A' = 1 \) (see Table 3), while the majority of users nodes are accessible during training time (transductive learning), though their reviews are not fully labeled to provide a value of \( A' \). We propose two novel methods to address these two challenges in the next two sections.

#### 3.2 Data augmentation for minority groups

**Augmentation for minority user subgroup \( \mathcal{V}_{0,1}^U \).** The lack of users in the subgroup makes the training of \( g_{\theta} \) difficult, leading to poor performance of the inference of \( A' \). We will augment the data in \( \mathcal{V}_{0,1}^U \) to address this issue.

First, the augmentation is to mimic the original distribution of data in \( \mathcal{V}_{0,1}^U \). Oversampling [8] adds multiple copies of the minority data as an augmentation and is straightforward. We replicate the minority user nodes in the subgroup \( \mathcal{V}_{0,1}^U \) along with their reviews and their connections the same products.

Rather than using exactly the same duplication, we slightly perturb the copies of user and review nodes to generate more variations. Similar to augmentation methods for image data, including flipping [23], cropping [31], rotation [11], noise injection [3], and so on, we randomly prune the reviews of the replicated users to create diverse neighbor distributions of the replicated users. GNN will
The augmentation of user nodes in $\mathcal{V}_{0,1}^U$ and their associated reviews, therefore generate diverse representations of the membership subgroup. We duplicate the minority subgroup “mixed user” $\mathcal{V}_{0,1}^U$ and their associated reviews, then randomly prune edges linked to the non-spam reviews.

3.2.2 Augmentation for minority review group $\mathcal{V}_{0,1}^R$. The number of reviews posted by group $A = 0$, denoted by $\mathcal{V}_{0,1}^R$, is minor compared to the other group, and the model $g_\theta$ may have difficulty modeling the small set of reviews. Mixup [44], one of the data augmentation methods, generates synthetic image data using convex combinations between any two original labeled data points to interpolate the otherwise sparse training distribution. Our method for augmenting the minority group $\mathcal{V}_{0,1}^R$ is based on the specific mixup framework [40] for graph data which considers both the node features and topology structures. Unlike their work, we do not implement the mixup over all the training nodes. Precisely, our augmentation method carefully delimits nodes for mixup so that the synthetic data will not violate fairness across groups. Here are the mixup for GNN’s input, node embeddings at each layer, and label for the synthetic data:

\[
\begin{align*}
\bar{x}_{ij} &= \alpha \cdot x_i + (1-\alpha) \cdot x_j, \quad (6) \\
\bar{h}_{ij} &= \alpha \cdot \bar{h}_{ij}^{(l)} + (1-\alpha) \cdot \bar{h}_{ij}^{(l)}, \quad (7) \\
\bar{y}_{ij} &= \alpha \cdot y_i + (1-\alpha) \cdot y_j. \quad (8)
\end{align*}
\]

where $\alpha \in [0, 1]$ and $\bar{x}_{ij}$ represents the mixture of node attributes $x_i$ and $x_j$ at the input layer. $\bar{h}_{ij}^{(l)}$ denotes the mixture at the $l$-th layer generated from the two aggregated node hidden representations $\bar{h}_{ij}^{(l)}$ and $\bar{h}_{ij}^{(l)}$. $\bar{y}_{ij}$ denotes the label for the synthetic data $\bar{x}_{ij}$.

First, since we would like the synthetic reviews to be similar to the existing reviews from the minority group $\mathcal{V}_{0,1}^R$, we ensure that at least one of the two nodes to be mixed up is sampled from users in $\mathcal{V}_{0,1}^R \cap \mathcal{V}_{0,1}^T$. Second, we also want to mitigate the class-imbalance issue within the minority group $A = 0$. We require one of the mixed node to be a spam review ($y_i = 1$). Formally, the first node to be mixed is sampled from

\[
S_1 = \{ v_i \mid v_i \in \mathcal{V}_{0,1}^R \cap \mathcal{V}_{0,1}^T, y_i = 1 \}.
\]

We propose three sources from which the second node can be sampled. The first case is treated as our method, and the second and third cases can be seen as two baselines.

First, we need to augment spam reviews and sample the second node from the spam review set. Compared to $\mathcal{V}_{0,1}^R$, there are more spam reviews inside the “protected” reviews group $\mathcal{V}_{1}^R$ (see Table 3), so that we sample the second node from

\[
S_{2,1} = \{ v_j \mid v_j \in \mathcal{V}_{1}^R \cap \mathcal{V}_{0,1}^T, y_i = 1 \}.
\]

Second, during transductive training, we can access the node degree of the test nodes (denoted as $\mathcal{V}_{0,1}^T$), which can be divided into the “protected” and “favored” groups according to node degrees. At this time, we only ensure that the second node belongs to the same group ($A = 1$) to the first node. The second node can be sampled from

\[
S_{2,2} = \{ v_j \mid v_j \in \mathcal{V}_{0,1}^R \cap \mathcal{V}_{0,1}^T \}.
\]

The third option is to utilize the “protected” reviews from the test set. In this case, the second node comes from a different group to the first node.

\[
S_{2,3} = \{ v_j \mid v_j \in \mathcal{V}_{1}^R \cap \mathcal{V}_{0,1}^T \}.
\]

Since we do not know the label for the second node in the second and third cases, the synthetic node will share the same label to the first node, i.e., $\bar{y} = 1$. To ensure the synthetic reviews are similar to ones from the minority group $\mathcal{V}_{0,1}^R$, we let the mixup weight $\alpha = 0.8$.

3.3 Joint model

In this section we present how to improve model fairness with the help of inferring subgroup membership $A'$. **Fairness for the subgroup.** Suppose we had known the subgroup membership, the most straightforward way to improve subgroup fairness is to introduce an additional fairness regularizer, such as disparate impact, over the subgroups to Eq. (3) when training $f_W$. However, we do not observe the value of $A'$ but can only infer its value probabilistically using the model $g_\theta$. Such probability output cannot be used by existing fairness regularizers that require deterministic sensitive attributes. We design the following joint model that can find $A'$ and optimize for fairness simultaneously.

For the inferred subgroup membership $A'$ of test user nodes, we treat it as additional information to inform the GNN model $f_W$ about how to treat the two subgroups differently. At this time, the concatenated feature of test user node fed into the classifier $f_W$ becomes to $x' = [x, A'] \in \mathbb{R}^{d+1}$ where $A'$ is the probability that one user has sensitive attribute $A' = 1$ inferred by $g_\theta$. For the training subgroup user node, we append the ground truth of $Pr(A' = 1)$ to the feature vector so that $x' = [x, A'] \in \mathbb{R}^{d+1}$ where $A'$ can be obtained from Eq. 4.

**Optimization for the joint models.** There are two GNNs, $g_\theta$ and $f_W$, to be trained. We propose to optimize $g_\theta$ and $f_W$ jointly so that
We first demonstrate the subgroup fairness issue commonly found where
which involves
Algorithm 1 Joint training for subgroup fairness
\begin{itemize}
\item \textbf{Input:} graph $G$: node features $X$; sensitive attribute $A$; number of training epochs $T$; hyper-parameter $\lambda$, $\beta_1$, $\beta_2$ and $k$.
\item \textbf{Output:} optimal model parameters $\theta$ and $W$.
\end{itemize}
Initialize parameters $\theta$ and $W$ of the two GNN models.
Replicate user nodes $k$ times as in section 3.2.1. $\triangleright$ Augment data for minority subgroup
for $t = 1, \ldots, T$ do
\begin{itemize}
\item Prune replicated edges as in section 3.2.1. $\triangleright$ Add data variations
\item Infer $Pr(A' = 1)$ for test users using $g_{\theta}$.
\item Concatenate $Pr(A' = 1)$ (ground truth $A'$, resp.) to test (training, resp.) user feature vectors.
\item Mixup using users sampled from $S_i$ with users sampled from one of $\{S_2, 1, S_3, S_3\}$. $\triangleright$ Data simultaneously for minority group
\item Evaluate $L(\theta; G)$ in Eq. (5) and $L_{\text{GNN}}(W; G)$ in Eq. (3).
\item Update $W$ and $\theta$ following Eq. (13) and (14).
\end{itemize}
end for

the two models can co-adapt to each other during training. The attribute $A'$ is not considered an observed constant but a function of the parameter $\theta$ as the model $g_{\theta}$, as we train the model $f_W$. The concatenated user feature becomes $x'(\theta) = [x, Pr(A' = 1|\theta)]$, which involves $\theta$ as parameters. Therefore, the loss for optimizing classifier $f_W$ in Eq. (3) becomes $L_{\text{GNN}}(W; \theta; G)$. During the training, $W$ and $\theta$ will be updated simultaneously:
\begin{equation}
W \leftarrow W - \beta_1 \nabla_W L_{\text{GNN}}(W; \theta; G)
\end{equation}
\begin{equation}
\theta \leftarrow \theta - \beta_2 \nabla_\theta L_{\text{GNN}}(W; \theta; G) - \beta_3 \nabla_\theta L_{\text{GNN}}(W; \theta; G)
\end{equation}
where $\beta_1$ and $\beta_2$ are two learning rates for updating parameters $W$ and $\theta$ respectively. See Algorithm 1 for a full description.

A baseline is to optimize the two models separately. On the training set, we can observe $A'$ of the users since we have access to the label of their posted reviews. Therefore, we can optimize the classifier $g_{\theta}$ by minimizing the loss function $L(\theta; G)$ in Eq. (5) on the training users. When predicting classes for the test users, we concatenate $Pr(A' = 1)$ to the test user feature vectors and applied the classification GNN $f_W$ that was trained separately using the ground truth $A'$ on the training set.

4 EXPERIMENTS

4.1 Results

We first demonstrate the subgroup fairness issue commonly found in a predictive GNN model, even with a fairness regularizer as in Eq. 2. Then, we demonstrate that subgroup fairness can be improved by introducing the subgroup membership $A'$. Lastly, we show the advantages of inferring $A'$ and node class simultaneously and study the sensitivity of the impacts of three subgroup augmentation strategies. We seek to answer the following research questions:

\begin{itemize}
\item \textbf{Q1:} Do fairness issues exist between the subgroups of “mixed” and “pure” users and between the groups of “favored” and “protected” users when using GNN for spam detection?
\item \textbf{Q2:} How to infer the subgroup membership with very limited data from one subgroup?
\item \textbf{Q3:} Will inferring subgroup membership help improve group and subgroup fairness?
\end{itemize}

4.2 Datasets

We use two commonly used Yelp review datasets (see Table 3) in previous spam detection [5, 14, 15]. For sensitive attribute $A$, we set 5% as the cutoff degree of user nodes to distinguish “favored” (top 5% high-degree user nodes, denoted as $A = 0$) from “protected” groups (the remaining user nodes, denoted as $A = 1$). All the review nodes have the same value of sensitive attribute $A$ corresponding to their associated users. In the last column of Table 3, we give the ratios of spam between groups $A = 0$ and $A = 1$. The imbalance distribution of spam reflects the fact that one review posted by a “favored user” is less likely to be spam. To study the subgroup fairness inside the favored group ($A = 0$), we further split the “favored user” into “pure” ($A' = 0$) and “mixed” ($A' = 1$) subgroups based on the label of their posted reviews following Eq. (4). We split all user nodes into training (30%), validation (10%), and test (60%) sets and divide the review nodes according to users who post the reviews. Note that $A'$ is known only on the training set.

4.3 Experimental Settings

4.3.1 Evaluation Metrics. Because of the imbalanced distributions of spams and non-spams, we utilize NDCG to evaluate the detector’s capability of ranking spams on the top for human inspection. NDCG can be evaluated on the entire test set or on individual (sub)groups. As we are also interested in the subgroup fairness within group $A = 0$, we propose another metric called “Average False Ranking Ratio” to evaluate the average of relative ranking between spams from “mixed” and “pure” users,

\[
AFRR = \frac{1}{Z} \sum_{j,y \in T} \sum_{i=1}^{\vert \mathcal{V}_r \vert} \Vert \hat{y}_i > \hat{y}_j, y_i = 0 \Vert \sum_{i=1}^{\vert \mathcal{V}_r \vert} \Vert \hat{y}_i > y_j, y_i = 0 \Vert
\]

where $Z$ denotes the total number of spams from a subgroup. The inner ratio of AFRR calculates the proportion of non-spams ranked higher than spam over all the non-spams from group $A = 0$. Then, we take the average of these ratios for all the spam reviews. NDCG for each subgroup only tells us whether spams are ranked higher than non-spams from the same subgroup, while AFRR considers all the non-spams across different subgroups. The lower the AFRR, the higher the ranks of spams over all non-spams. It is reasonable since we would like the detector to give a higher probability of being suspicious to spams than non-spams from all users rather than just a (sub)group of users.

4.3.2 Baselines. To demonstrate the importance of including subgroup information during the training and answering the above questions, we introduce two sets of baselines. The proposed method is denoted as “Joint+GNN-S2+1”. We use “a + b” to denote various experimental settings where “a” represents a variant for model $g_{\theta}$ and “b” represents a variant for model $f_W$.

Variants of $g_{\theta}$:

- W/O: vanilla GNN $f_W$ without the subgroup membership $A'$.
- Random: randomly concatenates 1/0 as subgroup memberships to user feature vectors in the group $A = 0$ on the test set.
Table 2: NDCG for GNN’s prediction on two Yelp datasets. Shown is the average results over ten different splits.

| GNN (fw) | NDCG(%) | YelpNYC | GNN(g₀) | Joint (Ours) | YelpZip | GNN(g₀) | Joint (Ours) |
|----------|---------|---------|---------|-------------|---------|---------|-------------|
|          |         | W/O     | Pre-trained |                         | W/O     | Pre-trained |                         |
| GNN      |         |         |          |              |         |          |              |
| Vₐ₊ᵛ      | 85.2 ± 0.8 | 85.1 ± 0.8 | 85.2 ± 0.5 | 88.4 ± 1.4 | 87.6 ± 1.5 | 88.6 ± 1.0 |
| Vₐ₋ᵛ      |           |           |          |              |         |          |              |
| Vₐ₋ᵛ'     | 21.9 ± 0.7 | 21.8 ± 0.8 | 21.3 ± 0.9 | 36.3 ± 10.4 | 34.3 ± 10.8 | 34.8 ± 11.1 |
| GNN-S₂₁ (Ours) |         |         |          |              |         |          |              |
| Vₐ₊ᵛ      | 85.8 ± 0.1 | 85.9 ± 0.0 | 85.9 ± 0.0 | 89.7 ± 0.2 | 89.7 ± 0.1 | 89.6 ± 0.1 |
| Vₐ₋ᵛ      | 85.9 ± 0.0 | 86.0 ± 0.0 | 86.0 ± 0.0 | 89.7 ± 0.2 | 89.7 ± 0.1 | 89.6 ± 0.1 |
| Vₐ₋ᵛ'     | 19.1 ± 5.5 | 19.0 ± 5.2 | 17.9 ± 6.1 | 38.7 ± 7.2 | 36.0 ± 9.0 | 34.3 ± 11.3 |
| GNN-S₂₂ (Ours) |         |         |          |              |         |          |              |
| Vₐ₊ᵛ      | 85.3 ± 0.6 | 85.4 ± 0.5 | 85.4 ± 0.4 | 89.4 ± 0.6 | 89.6 ± 0.1 | 89.0 ± 0.06 |
| Vₐ₋ᵛ      | 85.3 ± 0.6 | 85.5 ± 0.4 | 85.5 ± 0.4 | 89.4 ± 0.06 | 89.0 ± 0.01 | 89.1 ± 0.06 |
| Vₐ₋ᵛ'     | 21.9 ± 6.9 | 21.9 ± 6.3 | 20.9 ± 9.4 | 38.9 ± 7.6 | 36.9 ± 9.5 | 34.9 ± 11.0 |
| GNN-S₂₃ (Ours) |         |         |          |              |         |          |              |
| Vₐ₊ᵛ      | 85.7 ± 0.1 | 85.8 ± 0.1 | 85.8 ± 0.2 | 89.6 ± 0.3 | 89.6 ± 0.1 | 89.5 ± 0.3 |
| Vₐ₋ᵛ      | 85.8 ± 0.1 | 85.8 ± 0.1 | 85.8 ± 0.1 | 89.6 ± 0.3 | 89.6 ± 0.1 | 89.5 ± 0.3 |
| Vₐ₋ᵛ'     | 21.0 ± 5.4 | 19.8 ± 5.4 | 19.3 ± 6.8 | 38.7 ± 7.2 | 36.2 ± 9.5 | 34.6 ± 11.4 |

Table 3: Statistics of dataset. We list the number product, review and user nodes with the proportion of high-degree user (Vₐ₊ᵛ) and “mixed” user (Vₐ₋ᵛ) in each dataset. The last column gives the ratio of spans in group A = 0 and A = 1.

| Dataset | [Vₐ₊ᵛ] | | [Vₐ₋ᵛ] | | [Vₐ₋ᵛ'] | | P(Y=1) | | P(Y=1)[A=0] |
|---------|---------|---------|---------|---------|---------|----------|-----------|----------------|
| YelpNYC | 923 | 389911 | 160220 | (0.76%, 0.009%) | 0.0479 |
| YelpZip | 5044 | 608598 | 260277 | (0.27%, 0.002%) | 0.0426 |

- **GT:** concatenates ground truth of A’ to user feature vectors in group A = 0 on the test set. This is the ideal and yet unrealistic case, as A’ is unknown and g₀ has to infer A’.
- **Pre-trained:** is a variant of Joint. We pre-trained g₀ then infer A’ for the test users and fixed this inferred membership when training fw’.

**Variants of the GNN fw’:**
- S₂₂: sample the second node for mixup from set S₂₂ in Eq. (11).
- S₂₃: sample the second node for mixup from set S₂₃ in Eq. (12).

We set the number of training epochs T = 300, hyper-parameter λ = 5, learning rate β₁ = β₂ = 0.001, weight decay equals to 0.0001 for both fw and g₀, mixup weight ratio α = 0.8. Besides, we have 10 training-validation-test splits of the YelpNYC, and 9 training-validation-test splits of YelpZip. The following results are all based on the average over all the splits.

**Variant Group fairness.** To answer question Q1, we measure the difference in the NDCGs of reviews from groups A = 0 and A = 1 as the group fairness metric and the difference in AFRRs of spans between subgroups of A’ = 0 and A’ = 1 as the subgroup fairness metric. In Table 2, the columns represent three variants of g₀ on the two datasets, and the rows represent three variants of the GNN fw. We showed that the NDCG over all test reviews Vₐ₊ᵛ, the NDCG over test reviews from the “protected” group Vₐ₋ᵛ’, and the NDCG gap between groups A = 1 and A = 0 (denoted as ΔNDCG). Based on the ΔNDCG for the method W/O+GNN, there is an evident gap in detection efficiency between the favored and protected groups.

A lower NDCG score indicates that the unfair detector tends to assign a lower suspiciousness to spams from the “favored” group.

In Figure 3, we demonstrate the training and the test AFRR of the subgroups of “pure” and “mixed” with four methods, averaged over 9 training-validation-test splits of the YelpZip dataset. Based on the first subfigure representing W/O+GNN, it is clear that the detector has already generated unfairness predictions between the “pure” and “mixed” users during the training. In other words, inside group A = 0, the GNN fW tends to rank spans from “pure” users higher than spans from “mixed” users. Besides, by checking the median and the interquartile range of AFRR for the test “mixed” users, our method (subfigure b)) can improve the subgroup fairness by raising the rank of spans from “mixed” users.

Carefully checking the results in Table 2, if we fix the method for fw’, our “Joint” training method has the smallest fairness gap in most cases (only one exception for Joint-GNN on YelpZip). Meanwhile, fixing the method for g₀, our data augmentation method “GNN-S₂₁” has the best results among all the baselines on YelpNYC, and one case in YelpZip. Since YelpZip is a sparse and large graph with fewer minority subgroup nodes (cf. %Vₐ₋ᵛ in Table 3), it is difficult to augment data without modifying the original subgroup distribution. Besides, detectors are prone to overfit if we start from an extremely small training set.

**Impact of g₀.** We attempt to answer questions Q2 and Q3 by studying the impact of subgroup membership A’ and its inference accuracy. In Figure 4, we demonstrate the average impact of A’ to the group fairness by generating A’ in different methods over nine splits on the YelpZip test set. There are four groups of stacked bars which represent four different methods for g₀. Within each group of stacked bars, there are five variants for generating A’, with less and less noise in A’ going from the left-most to the right-most bars. It is clear that the fairness gap will be reduced as fw receives more accurate inference of A’: the closer the output of g₀ to the ground truth A’, the smaller the group fairness gap. In particular, the setting of using the unknown ground truth A’ provides a theoretical lower bound of fairness gap for other four methods.

Since accurate inference of A’ is vital to the group fairness, we further analyze the advantage of the joint training strategy from the perspective of prediction AUC of A’ using the model g₀. In
Figure 3: Box plot for AFRR on spams from training and test “mixed” and “pure” users over nine splits of YelpZip. (a) only second GNN $f_W$ without any mixup method. (b)-(d) Jointly training $g_\theta$ and $f_W$ with three types of mixup method. Subgroup fairness improved by introducing $A'$ and joint training method.

Figure 4: Average performance of test $\Delta_{NDCG}$ for the second GNN $f_W$ with different settings of $A'$ for the test user nodes. ("W/O": without concatenating $A'$; "Random": randomly assign $A' = 1/0$; "Pre-trained": output of pre-trained GNN $g_\theta$; "Joint": output of jointly trained GNN $g_\theta$; "GT": ground truth of $A'$.)

Table 4: Performance of first GNN $g_\theta$. AUCs for the prediction on sensitive attribute $A'$ for the test Group $A = 0$ users. The top rows are under the joint training strategy with different mixup methods for the second GNN $f_W$. The bottom gray row is AUC of $g_\theta$ under the pre-trained strategy where $g_\theta$ is fixed when training $f_W$. AUC over the test users from the group $A = 0$ is improved using most “Joint” training methods. The only two exceptions are “Joint+GNN” on both two datasets. Different from “Pre-trained”, the “Joint” training method updates the parameters $\theta$ using the gradient from $f_W$ (see Eq. 14). When there is no data augmentation (i.e., Joint+GNN), this gradient will bias $g_\theta$ to overfit the training group $A = 0$ data resulting in a low AUC on the test set. We find that the proposed mixup strategies over review nodes for $f_W$ can mitigate the overfitting problem of $g_\theta$.

Ablation and sensitivity study. We study the average impact of our mixup method (GNN-S(2,1)) and other mixup baselines (GNN-S(2,2), GNN-S(2,3), and no mixup) on group fairness. In Figure 5, we show the group fairness gap between “protected” and “favored” groups with different mixup strategies. Each line represents one variant of the model $f_W$, and the dash lines depict the methods without mixup. The detector has the smallest NDCG gap for each solid line by using our mixup strategy GNN-S(2, 1). It can demonstrate
that delimiting appropriate nodes for mixup is important for the fairness problem. Comparing the solid and dash lines, we can see that the solid lines are below their corresponding dash lines in most cases on YelpNYC, and $f_W$ has the smallest fairness gap using the proposed mixup method. However, on YelpZip, the mixup method improves group fairness only using our “Joint” training strategy (green line). The sparsity of the YelpZip graph causes this effect. Although mixup can mitigate overfitting problems for $g_0$ regardless of graph characteristics such as sparsity and size, we still need to consider those characteristics when using mixup to improve the group fairness.

Besides, we study the sensitivity of duplication times $k$ and the impact of pruning non-spam edges during the augmentation for the minority “mixed” subgroup. In Figure 6, we show the test AUCs of the model $g_{\theta}$ on two datasets with different numbers of replications $k = \{50, 100\}$ and whether to prune replicated edges. It is clear that pruning replicated edges has better AUCs than only replication only in most cases. We fix the $k = 100$ for YelpNYC and $k = 50$ for YelpZip in other experiments.

5 RELATED WORK

Fairness on graphs. People studied fairness on graphs from many perspectives. In [6, 32] researchers attempt to obtain fairness graph embedding and representation, where the sensitive attribute is known and able to divide the graph into several groups. In [1, 4, 12], people introduce the adversarial framework to eliminate the unfairness bias concerning the sensitive attribute. They add fairness regularization terms to constrain their model insensitive to specific protected (sensitive) attributes. Besides, other works solve unfairness node embedding problems by modifying graph structure or re-weighting the edges. FairAdj [25] adjusts the graph connections and learns a fair adjacency matrix by adding graph structural constraints. FairEdit [27] proposes model-agnostic algorithms which perform edge addition and deletion by leveraging the gradient of their fairness loss. FairDrop [35] excludes the biased edges to counter-act homophily which causes the unfairness issue in graph representation learning. However, these methods depend on the known or observable sensitive attributes so that these methods only aim to de-bias the node representation toward each group defined by sensitive attributes.

Subgroup fairness. Some works study the subgroup fairness problem on the LLD. data. If we force the group fairness with respect to the pre-defined sensitive attribute, then there will be some fairness violations on the subgroups of these pre-defined groups. In [20, 21], they learn classifiers subject to fairness constraints when the number of protected groups is large. The classifier satisfies the fairness constraints for the combinatorially large collection of structured subgroups definable over protected attributes. In [29], they study the subgroup generalization and fairness on a graph and demonstrate that distance between a test subgroup and the training set leads to the performance of GNN.

Augmentation on the graph. Data augmentation on the graph has increasingly received attention recently. In [16, 40], they study the graph augmentation on the node-level that synthetic data by mixup nodes or removing nodes from the original graph. Besides, some researchers operate graph augmentation on edge-level where they modify (adding or removing edges) in deterministic [45] or stochastic way [34].

6 CONCLUSION

In this paper, we studied the subgroup fairness problem on the graph-based spam detection task. We first present that subgroup unfairness exists inside the “favored” group divided by the pre-defined sensitive attribute (node-degree, denoted as $A$). To address this subgroup unfairness problem, we propose a new sensitive attribute ($A'$) for users based on the label distribution of users’ posted reviews. For users associated with unlabeled reviews, we introduce another GNN model to infer $A'$ with the graph augmentation method (duplicating the minority subgroup and pruning the non-spam edges). Furthermore, we treat the inference of $A'$ as an indicator variable concatenating to the original user node feature and train another GNN with implementing the mixup method on the minority group for the spam detection task. According to the experimental results, our joint training strategy for two GNN models with two augmentation methods (one for minority subgroup, one for minority group) effectively reduces the NDCG gap between groups and the AFRR between subgroups.

REFERENCES

[1] Chirag Agarwal, Himabindu Kakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In Uncertainty in Artificial Intelligence. PMLR, 2114–2124.

[2] Pranjal Awasthi, Matthias Klenkens, and Jamie Morgenstern. 2020. Equalized odds postprocessing under imperfect group information. In International Conference on Artificial Intelligence and Statistics. PMLR, 1779–1780.
