SeBot: Structural Entropy Guided Multi-View Contrastive Learning for Social Bot Detection

Yingguang Yang
University of Science and Technology of China
dao@mail.ustc.edu.cn

Qi Wu
University of Science and Technology of China
qiwu4512@mail.ustc.edu.cn

Buyun He
University of Science and Technology of China
byhe@mail.ustc.edu.cn

Hao Peng*
Beihang University
penghao@buaa.edu.cn

Renyu Yang
Beihang University
renyu.yang@buaa.edu.cn

Zhifeng Hao
Shantou University
haozhifeng@stu.edu.cn

Yong Liao*
University of Science and Technology of China
yliao@ustc.edu.cn

ABSTRACT

Recent advancements in social bot detection have been driven by the adoption of Graph Neural Networks. The social graph, constructed from social network interactions, contains benign and bot accounts that influence each other. However, previous graph-based detection methods that follow the transductive message-passing paradigm may not fully utilize hidden graph information and are vulnerable to adversarial bot behavior. The indiscriminate message passing between nodes from different categories and communities results in excessively homogeneous node representations, ultimately reducing the effectiveness of social bot detectors. In this paper, we propose SeBot, a novel multi-view graph-based contrastive learning-enabled social bot detector. In particular, we use structural entropy as an uncertainty metric to optimize the entire graph's structure and subgraph-level granularity, revealing the implicitly existing hierarchical community structure. And we design an encoder to enable message passing beyond the homophily assumption, enhancing robustness to adversarial behaviors of social bots. Finally, we employ multi-view contrastive learning to maximize mutual information between different views and enhance the detection performance through multi-task learning. Experimental results demonstrate that our approach significantly improves the performance of social bot detection compared with SOTA methods.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Security and privacy** → **Social network security and privacy**.

*Corresponding authors

KDD '24, August 25–29, 2024, Spain

ACM ISBN 978-1-4503-9416-1/23/04
https://doi.org/10.1145/xxxxxxx.xxxxxxx

KEYWORDS

social bot detection, graph neural networks, contrastive learning, structural entropy

ACM Reference Format:
Yingguang Yang, Qi Wu, Buyun He, Hao Peng, Renyu Yang, Zhifeng Hao, and Yong Liao. 2024. SeBot: Structural Entropy Guided Multi-View Contrastive Learning for Social Bot Detection. In *The 30th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/xxxxxxx.xxxxxxx

1 INTRODUCTION

Social bots are automated controlled accounts that widely exist on social platforms such as Twitter and Weibo, in most cases, for malicious purposes such as spreading misinformation [16, 42], manipulating public opinion [18, 43], and influencing political elections [10, 29]. They will undoubtedly give rise to societal disharmony in social network environments.

Conventional methods for social bots detection primarily focus on extracting discriminative features, ranging from user attributes, [48], text features [21] to structure features[5], which can be then used to train classifiers in a supervised manner. Due to the continuous evolution of social bots [9], including their ability to steal information from legitimate accounts and mimic normal account behaviors [26], these traditional methods turn out to be ineffective in identifying the latest generation bots. A recent advancement in social bot detection is introducing graph neural networks [1] that treat accounts and the interactions in-between as nodes and edges, respectively. Multi-relational heterogeneous graphs can be established [13] and a Relation Graph Transformer (RGT) is responsible for aggregating information from neighbors. Such approaches consider the features of each account as interdependent ones and leverage the semantic information of relationships between accounts to generate semantically-richer representations.

While successful, there are two challenges facing the existing graph-based methods (illustrated in Figure 1). *i) How to fully exploit the hierarchical information hidden in the graph structure?* Unlike other semi-supervised node classification tasks, some accounts in

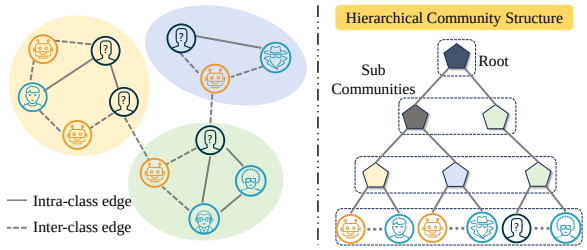


Figure 1: Illustration of community structure and inter-class interactions co-exists in social networks. The abstraction of the hierarchical community structure is presented in the form of an encoding tree.

social platforms tend to exhibit stronger correlations with others in the topological structure due to shared interests, events of interest, etc. [9], indicating an intrinsic hierarchical community structure in the graph constructed by social bot detectors. The existing graph-based social bot detection methods [13, 15, 20, 33, 46, 50, 51, 55] primarily focused on aggregating node-level information on the original graph, neglecting comprehensive utilization of high-order graph structural information. As a result, the generated representations only capture low-order graph information and lack high-order semantic information. *ii) How to handle the adversarial behaviors of bots intentionally interacting with humans to evade detection?* Existing bot detection methods rely on the assumption that bots and humans exhibit stronger connections with nodes of their category. However, for social bots, especially in the case of advanced social bot control programs, gathering human-specific information from neighbors would be more advantageous for concealing their true identity. Therefore, bots intentionally tend to establish associations with human accounts to escape detection [7, 11].

To tackle these two challenges, we propose SEBOT, a novel Structural Entropy Guided Social Bot Detection framework through graph contrastive learning enhanced classification with both intra-class and inter-class edges. Motivated by minimum entropy theory [23], indicating systems at the minimum level of uncertainties, and structural entropy [27] further offers an effective measure of the information embedded in an arbitrary graph and structural diversity. We construct encoding trees for both the entire input graph and the subgraphs of each node by minimizing their structural entropy. We then use the optimal encoding trees to describe the hierarchical structural information of the graph and obtain cluster assignment matrixes for the nodes. Subsequently, node representations are obtained through structural entropy pooling and unpooling for node and graph classification. On the other hand, we designed an encoder that can adaptively make neighbors similar or differentiate them to handle the adversarial behavior of bots. Finally, we utilize the node representations generated by the three modules to calculate both cross-entropy loss and self-supervised contrastive learning loss.

The contributions of this work can be summarized as follows:

- We are the first to introduce structural entropy to capture semantic information at both subgraph and entire-graph levels in a self-supervised manner.
- We propose a self-supervised contrastive learning framework called SEBOT, which integrates node-node and subgraph-subgraph level contrastive learning tasks in a multi-task learning manner to capture high-order semantic information.
- Experiments on two real-world bot detection benchmark datasets demonstrate that our method outperforms current state-of-the-art social bot detection methods.

2 RELATED WORK

Graph-based Social Bot Detection. Graph-based social bot detection has been of ultimate importance in modeling various interactions intrinsically existing in social networks. Previous methods [1, 13, 15, 46] have focused primarily on designing information aggregation strategies for better detection performance. [1] takes the first attempt to use graph convolutional neural networks (GCNs) [25] for detecting social bots. Typically, BotRGCN [15] utilizes relational graph convolutional networks (RGCNs) [35] to aggregate neighbor information from edges of different relations. [13] proposed RGT, which utilizes a self-attention mechanism to adaptively aggregate information from neighbors in each relational view of the graph. Although these methods have shown significant improvements compared to traditional feature engineering and text-based approaches [12], they may not fully exploit the crucial semantic information concealed in the graph structure and the graph structure obtained from sampling social networks contain a significant amount of uncertainty and randomness.

Graph Self-Supervised Learning. Self-supervised learning has achieved great success in the fields of natural language processing [49] and computer vision [24] without the need for prohibitively costly labeled data. Graph contrastive learning (GCL) is a typical paradigm of self-supervised learning on graphs, aiming at learning invariant representations between different graph views. For instance, DGI [41] utilizes a local-global mutual information maximization approach to obtain node representations. [52] proposes a series of graph augmentation methods including node dropping, edge perturbation, attribute masking, and so on. [60] propose an adaptive way for graph augmentation, which assigns adaptive probabilities to attribute and topological perturbation. However, these augmentation methods inevitably suffer from the loss of essential information or introduce class-redundant noise. Due to the significant impact of the quality of generated views on contrastive learning, [44] theoretically proves that the anchor view containing essential semantic information should have the minimum structural entropy. Inspired by this, structural entropy is employed by us to generate the anchor view with minimum uncertainty.

Structural Entropy. After Shannon proposed information entropy to measure system uncertainty [36], the measurement of uncertainty in graph structure has been widely studied, and several methods [30, 34, 38] have been developed to quantify it. Among them, structural entropy [6, 27, 53, 54, 56] has been widely used in recent years and has shown promising results in graph structure learning [61], graph pooling[45], and other tasks. Since structural entropy can be used as a metric to measure the complexity of graph hierarchical structure, previous applications have mainly focused on minimizing the structural entropy of the constructed encoding tree. For instance, SEP [45] defines MERGE, REMOVE, and FILL operations to update the constructed encoding tree based on the principle of minimizing structural entropy. In this paper, we

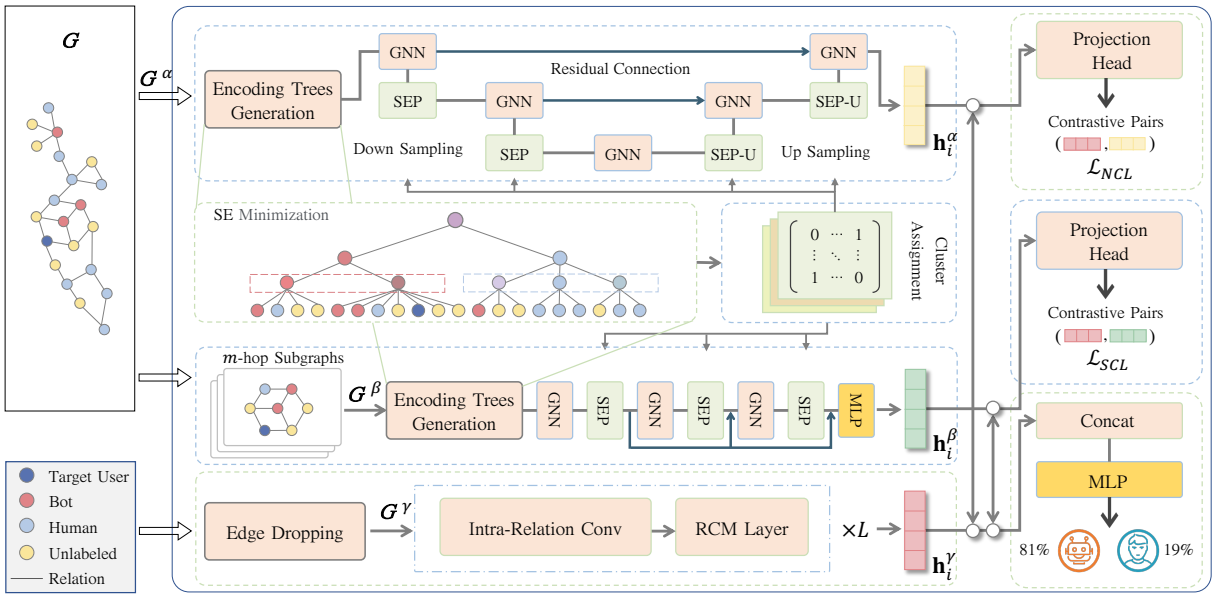


Figure 2: Overview of our proposed framework SeBot, which mainly consists of three modules: 1) Node-level encoding tree generation and bottom-up message passing; 2) Subgraph-level encoding trees generation and message-passing; 3) Relational information aggregation beyond homophily. Contrastive learning loss and classification loss are later calculated on obtained tree types of representations.

employ structural entropy in a self-supervised manner to capture information at both the node level and subgraph level.

3 PRELIMINARIES

In this section, we first illustrate the graph-based social bot detection task, followed by an introduction to the definition of graph contrastive learning.

Definition 3.1. Graph-based Social bot detection. Graph-based social bot detection can be regarded as a semi-supervised node binary classification problem on a multi-relational graph. It involves treating the accounts in social platforms as nodes and the interactions such as "following" and "follower" as edges of different relations. Constructed graph in this task can be formulated as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$, where $\mathcal{V} = \{v_i \mid i = 1, 2, \ldots, n\}$ is the set of all nodes, $\mathcal{E} = \cup_{r=1}^{R} \mathcal{E}_r$ represents the set of edges formed by R relations and $\mathcal{X}$ is the feature matrix. Row i of $\mathcal{X}$ represents the feature vector of the i-th node. Total detection process is to use the graph $\mathcal{G}$ and the labels of training nodes $\mathrm{Y}_{\text{train}}$ to predict the labels of test nodes Y_{test} :

$$f\left(\mathcal{G}, \mathrm{Y}_{\text{train}}\right) \rightarrow \bar{\mathrm{Y}}_{\text{test}} . \tag{1}$$

Definition 3.2. Graph Contrastive Learning. In the general graph contrastive learning paradigm for node classification, two augmented graphs $\mathcal{G}^{\alpha}$, and $\mathcal{G}^{\beta}$ are generated using different graph augmentation methods (such as edge dropping, feature masking, etc.) on the input graph $\mathcal{G}$. Subsequently, an encoder consisting of multi-layer graph neural networks is employed to generate node representations including topological information existing in graph structure. During the first training stage, these representations are further mapped into an embedding space by a shared projection head for contrastive learning. A typical graph contrastive loss, InfoNCE [31], treats the same node in different views v_i^{α} and v_i^{β} as positive pairs and other nodes as negative pairs. The graph contrastive learning loss $\mathcal{L}_i$ of node v_i and total loss $\mathcal{L}$ can be formulated as:

$$\begin{aligned} \mathcal{L}_i^{\alpha} &= -log\left(\frac{e^{sim(\mathbf{z}_i^{\alpha}, \mathbf{z}_i^{\beta})/\tau}}{\sum_{j=1}^{N} \mathbb{1}_{j \neq i} e^{sim(\mathbf{z}_i^{\alpha}, \mathbf{z}_j^{\alpha})/\tau} + e^{sim(\mathbf{z}_i^{\alpha}, \mathbf{z}_j^{\beta})/\tau}}\right), \\ \mathcal{L} &= InfoNCE(\mathbf{Z}^{\alpha}, \mathbf{Z}^{\beta}) = \sum_{i=1}^{N} \mathcal{L}_i^{\alpha} + \mathcal{L}_i^{\beta}, \end{aligned} \tag{2}$$

where N is the batch size, τ is the temperature coefficient, $\mathbb{1}_{j \neq i} = 1$ when $j \neq i$ and $sim(\cdot, \cdot)$ stands for cosine similarity function.

4 METHODOLOGY

4.1 Overview of SeBot

The total pipeline of SeBot is illustrated in Figure 2. To begin with, an attributed multi-relational graph $\mathcal{G}$ is constructed by representing social network interactions as edges. Subsequently, it is fed into three different modules to obtain representations at multi-grained levels under various receptive field scopes $\mathbf{h}_i = [\mathbf{h}_i^{\alpha} \parallel \mathbf{h}_i^{\beta} \parallel \mathbf{h}_i^{\gamma}]$. The two modules above are responsible for constructing encoding trees by minimizing structural entropy for the entire graph view $\mathcal{G}^{\alpha}$ and m-order subgraph view $\mathcal{G}^{\beta}$ respectively. $\mathcal{G}^{\beta}$ is formed by the target node and surrounding neighbor nodes. The view $\mathcal{G}^{\gamma}$ is generated by edge removing. Then, bottom-up information propagation is performed according to these encoding trees to obtain node embeddings $\mathbf{h}_i^{\alpha}$ and $\mathbf{h}_i^{\beta}$. In the third module, to counteract

the intentional evasion behaviors of social bots, we devise a graph convolutional layer that can make neighbors similar or discriminative adaptively while maintaining normalization. Simultaneously, a relational channel-wise mixing (RCM) layer is proposed to integrate information from different relations to obtain $\mathbf{h}_i^Y$. Finally, the fusion of the three modules involves computing cross-entropy loss for classification and utilizing graph self-supervised contrastive learning loss to capture shared information between three views, enhancing the classification learning process.

4.2 Community-aware Hierarchical Augment

In social networks, some accounts may exhibit more pronounced connections with each other due to shared interests, events, and so on, thus forming communities. However, previous detection methods have not effectively leveraged the community structure information within social networks. To reveal the hierarchical structure within the graph, we utilize structural entropy minimization to obtain fixed-height encoding trees, where the child nodes of each node belong to the same community. For the sake of clarity, we first illustrate the definition of structural entropy and its minimization algorithm.

Structural Entropy. Structural entropy is initially proposed by [27] to measure the uncertainty of graph structure information. The structural entropy of a given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$ on its encoding Tree T is defined as:

$$\mathcal{H}^T(\mathcal{G}) = -\sum_{v_\tau \in T} \frac{g_{v_\tau}}{vol(\mathcal{V})} \log \frac{vol(v_\tau)}{vol(v_\tau^+)}, \tag{3}$$

where v_τ is a node in T except for root node and also stands for a subset $\mathcal{V}_\tau \in \mathcal{V}$, g_{v_τ} is the number of edges connecting nodes in and outside $\mathcal{V}_\tau$, v_τ^+ is the immediate predecessor of of v_τ and $vol(v_\tau)$, $vol(v_\tau^+)$ and $vol(\mathcal{V})$ are the sum of degrees of nodes in v_τ, v_τ^+ and $\mathcal{V}$, respectively. The structural entropy of graph $\mathcal{G}$ is the entropy of the encoding tree with the minimum structural entropy: $\mathcal{H}(\mathcal{G}) = \min_{\forall T}\{\mathcal{H}^T(\mathcal{G})\}$. According to this definition, structural entropy can be used to decode the hierarchical structure of a given graph into an encoding tree as a measurement of community division. In addition, the generated encoding tree T can be seen as the natural multi-grained hierarchical community division result.

Minimization Algorithm. In addition to the optimal encoding tree with the minimum structural entropy, a fixed level of community partitioning is preferred for the specific scenario. Considering this, the k-dimension structural entropy of $\mathcal{G}$ is defined as the optimal encoding tree with a fixed height k:

$$\mathcal{H}^{(k)}(\mathcal{G}) = \min_{\forall T:\text{Height}(T)=k} \{\mathcal{H}^T(\mathcal{G})\}. \tag{4}$$

The total process of generation of a encoding tree with a fixed height k can be divided into two steps: 1) construction of the full-height binary encoding tree and 2) compression of the binary encoding tree to height k. Given root node v_r of the encoding tree T, all original nodes in graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are treated as leaf nodes. We first define two iterative operations on T.

Definition 4.1. Assuming v_c^1 and v_c^2 as two children of root node v_r, the function $\textbf{Merge}(v_c^1, v_c^2)$ is defined as adding a new node v_i as the child of v_r and the parent of v_c^1 and v_c^2:

$$\begin{aligned} v_i.children &= \{v_c^1, v_c^2\}, \\ v_r.children &= \{v_i\} \cup v_r.children. \end{aligned} \tag{5}$$

Definition 4.2. Given node v_τ and its parent node v_τ^+ in T, the function $\textbf{Drop}(v_\tau)$ is defined as adding the children of v_τ and itself to the child set of v_τ^+:

$$v_\tau^+.children = v_\tau^+.children \cup v_\tau.children. \tag{6}$$

The generation of the encoding tree with a fixed height k primarily involves iterations through two operations to obtain the minimum structural entropy, which is shown in Algorithm 1. To start with, we initialize an encoding tree T by treating all nodes in $\mathcal{V}$ as children of root node v_r. During step 1, an iterative $\textbf{Merge}(v_c^1, v_c^2)$ is conducted with the aim of minimizing structural entropy to obtain a binary coding tree without height limitation. In this way, selected leaf nodes are combined to form new community divisions with minimal structural entropy. Then, to compress height to a specific hyperparameter k, $\textbf{Drop}(v_\tau)$ is leveraged to merge small divisions into larger ones, and thus the height of the coding tree is reduced, which is still following the structural entropy minimization strategy. Eventually, the encoding tree with fixed height k and minimal structural entropy is obtained.

Obtained encoding tree in this manner can also be seen as the anchor view that includes minimal but sufficient important information in an unsupervised manner [44]. The quality of views generated by graph augmentation technologies plays a crucial role in learning informative representations. According to graph information bottleneck theory (GIB) [47], retaining important information in a graph view should involve maximizing mutual information between the output and labels while reducing mutual information between input and output. This can be formally expressed as follows:

$$\text{GIB}: \max I(f(G); Y) - \beta I(G; f(G)), \tag{7}$$

Algorithm 1: Structural entropy minimization algorithm

Input : input undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a specific height $k > 1$

Output : encoding tree T with a fixed height k

initialize encoding tree T with root node v_r and nodes in $\mathcal{V}$ as its children ;

// Step 1: full-height binary coding tree construction ;

while $|v_r.children| > 2$ **do**

  select child node pair $(v_c^1, v_c^2) \leftarrow argmax_{(v_c^1, v_c^2)}\{\mathcal{H}^T(\mathcal{G}) - \mathcal{H}^{T_c}(\mathcal{G}) | v_c^1, v_c^2 \in v_r.children\}$;

  Merge(v_c^1, v_c^2);

end

// Step 2: binary coding tree squeeze to height k ;

while $Height(T) > k$ **do**

  select node $v_\tau \leftarrow argmin_{(v_\tau)}\{\mathcal{H}^T(\mathcal{G}) - \mathcal{H}^{T_m}(\mathcal{G}) | v_\tau \in T \& v_\tau \neq v_r \& v_\tau \notin \mathcal{V}\}$;

  Drop(v_τ) ;

end

return encoding tree T ;

where $I(\cdot;\cdot)$ represents the mutual information between inputs. A lot of graph augmentation methods have been proposed, including edge removal, feature masking, graph diffusion, and more. Yet, these methods cannot ensure the preservation of essential information regarding downstream tasks and may introduce class-redundant noise.

4.3 Message Passing on Encoding Tree

To obtain node representations and subgraph representations, the message passing on the encoding tree is carried out bottom-up, where the generated parent nodes aggregate information from their child nodes. This process begins with the leaf nodes (i.e., the nodes in $\mathcal{V}$) at the first layer transmitting information to their second-layer parent nodes. Specifically, given the cluster assignment matrix $\mathbf{S}_t \in \mathbb{R}^{n_t \times n_{t+1}}$ where n_t and n_{t+1} are the number of nodes and assigned clusters (i.e. nodes in the next layer) in the t-th layer, each element in $\mathbf{S}_t$ equal to 1 indicates that the node belongs to a corresponding cluster. The adjacency matrix $\mathbf{A}_{t+1} \in \mathbb{R}^{n_{t+1} \times n_{t+1}}$ and the hidden layer representations $\mathbf{P}_{t+1} \in \mathbb{R}^{n_{t+1} \times d}$ for the $(t+1)$-th layer can be obtained by matrix multiplication:

$$\textbf{SEP}: \mathbf{A}_{t+1} = \mathbf{S}_t^\top \mathbf{A}_t \mathbf{S}_t; \quad \mathbf{P}_{t+1} = \mathbf{S}_t^\top \mathbf{H}_t, \tag{8}$$

where $\mathbf{A}_t$ is the adjacency matrix and $\mathbf{H}_t$ denotes the hidden features martix in the t-th layer. As pooling continues, the number of nodes decreases. To obtain representations of nodes in $\mathcal{V}$, we further employ unpooling to ensure that the number of nodes matches the number of nodes in $\mathcal{V}$:

$$\textbf{SEP-U}: \mathbf{A}_{t+1} = \mathbf{S}_t \mathbf{A}_t \mathbf{S}_t^\top; \quad \mathbf{P}_{t+1} = \mathbf{S}_t \mathbf{H}_t, \tag{9}$$

where $\mathbf{S}_t$ is the same matrix used in previous pooling layers. The node-level representation $\mathbf{h}_i^\alpha$ is obtained through multiple layers of SEP to obtain high-order community representations and multi-layer SEP-U reconstructions.

On the other hand, the representation of each subgraph extracted for every target user can be obtained by concatenating the results of SEP pooling layers:

$$\begin{aligned} \textbf{SEP-G}: \mathbf{h}_i^{\beta,t} &= Readout(SEP_t(GCN_t(\mathbf{H}_t, \mathbf{A}_t), \mathbf{S}_t)) \\ \mathbf{h}_i^{\beta} &= Concat(\mathbf{h}_i^{\beta,1}, \cdots, \mathbf{h}_i^{\beta,l}), \end{aligned} \tag{10}$$

where $SEP_t(\cdot)$ denotes the t-th SEP layer and $Readout(\cdot)$ can be chosen between average pooling and sum pooling. These high-order representations obtained in this manner enable social bot detection to be implemented through subgraph classification.

4.4 Relational Information Aggregation

To alleviate graph adversarial attacks by social bots (i.e., actively establishing relationships with humans), we propose a relational information aggregation mechanism beyond homophily and a relation channel-wise mixing layer in this subsection.

4.4.1 Relational graph convolution beyond resemblance limitation. Previous work [15] has adopted RGCN to detect bot accounts and has shown promising success in modeling different relations. However, the information aggregation of RGCN is based on the homophily assumption (i.e., nodes belonging to the same class tend to be connected), and advanced bots may consciously interact more with humans. Considering this issue, we incorporate high-frequency information (i.e., the differences between nodes) into the information aggregation strategy of RGCN through the generation of negative attention coefficients. However, positive and negative weights generated directly through the tanh activation function can not be normalized. To ensure the consistency of information aggregation, we further introduce the Gumbel-Max reparametrization trick [22] to make the weights close to 1 or -1. Specifically, the attention weight $\omega_{ij}^{r,\{l\}}$ of the edge e_{ij} in relation r and the layer l is generated by:

$$\omega_{ij}^{r,\{l\}} = \tanh\left((\mathbf{g}_r^{\{l\},\top} \left[\mathbf{h}_i^{\{l-1\}} \parallel \mathbf{h}_j^{\{l-1\}} \right] + log\epsilon - log(1-\epsilon))/\tau_G \right), \tag{11}$$

where $\mathbf{g}_r^{\{l\}} \in \mathbb{R}^{2D}$ denotes the trainabele parameter, $\epsilon \sim Uniform(0,1)$ is the sampled Gumbel random variate and τ_G is a small temperature used to amplify $\omega_{ij}^{r,\{l\}}$. In this manner, when the weights are close to 1, it retains similar information with neighbors, whereas when close to -1, it preserves dissimilar information. In addition, normalization can be directly performed based on the number of neighbors:

$$\mathbf{h}_i^{r,\{l\}} = \frac{1}{|N_r(v_i)|} \sum_{v_j \in N_r(v_i)} \omega_{ij}^{r,\{l\}} \mathbf{h}_j^{\{l-1\}} W_r^{\{l\}}, \tag{12}$$

where $N_r(v_i)$ is the neighbors of v_i and $W_r^{\{l\}}$ is the weight matrix. Due to the adopted parametrization tick, normalization is approximately satisfied by $|N_r(v_i)| \approx \sum_{v_j \in N_r(v_i)} |\omega_{ij}^{r,\{l\}}|$.

4.4.2 Relational Channel-wise Mixing. After aggregating information independently across different relations, it is necessary to merge this relational information to acquire representations with richer semantics. Previous adaptive method [13] is limited to generating normalized weight coefficients for representations in each relation, inevitably introducing some noise information present in specific relations. Inspired by [28], we propose a relational channel-wise adaptive fusion layer. Specifically, given the embeddings $\mathbf{h}_i^{r,\{l\}}$ generated in each relation and at layer l, we first calculate the weight vector of the relation:

$$\hat{\mathbf{u}}_i^{r,\{l\}} = [\mathbf{h}_i^{1,\{l\}} \parallel \cdots \parallel \mathbf{h}_i^{R,\{l\}}] \hat{W}_r^{\{l\}}, \tag{13}$$

where $\hat{W}_r^{\{l\}}$ is the weight matrix and $[\cdot||\cdot]$ is the concat operation.

Then, we apply a channel-wise softmax function to normalize weight coefficients at each feature channel:

$$[\mathbf{u}_i^{1,\{l\}}, \ldots, \mathbf{u}_i^{R,\{l\}}] = Softmax([\hat{\mathbf{u}}_i^{1,\{l\}}, \ldots, \hat{\mathbf{u}}_i^{R,\{l\}}]). \tag{14}$$

Last, $\mathbf{u}_i^{r,\{l\}}$ can be used to mix representations from different relations and add transformed original features:

$$\mathbf{h}_i^{\{l\}} = \mathbf{h}_i^{\{l-1\}} W_{root}^{\{l\}} + \sum_{r=1}^{R} \mathbf{h}_i^{r,\{l\}} \odot \mathbf{u}_i^{r,\{l\}}, \tag{15}$$

where $\odot$ denotes the Hadamard product operation and $W_{root}^{\{l\}}$ is the weight matrix to transform original features.

In fact, when each edge weight $\omega_{ij}^{r,\{l\}} = 1$ and $\mathbf{u}_i^{r,\{l\}} = \mathbf{1}$, the proposed encoder is equal to RGCN and thus can be seen as an extension of it. Besides, L layers of the two networks described above are used to generate $\mathbf{h}_i^{\gamma}$ on $\mathcal{G}^{\gamma}$.

4.5 Multi-Task Optimization and Learning

After obtaining node representations from the three modules, it is necessary to design the loss function as the learning objective. The primary objective of the model remains classification, to identify bot accounts. We employ a two-layer MLP as the classification layer for the concatenation $\mathbf{h}_i = [\mathbf{h}_i^{\alpha} \parallel \mathbf{h}_i^{\beta} \parallel \mathbf{h}_i^{\gamma}]$ and calculate the classification loss using binary cross-entropy:

$$\begin{aligned} \mathcal{L}_{CE} &= -\sum_{v_i \in \mathcal{V}_t} [y_i log(p_i) + (1 - y_i) log(1 - p_i)] \\ p_i &= softmax(\sigma(\mathbf{h}_i W_1 + b_1) W_2 + b_2). \end{aligned} \tag{16}$$

Additionally, as shown in Figure 1, in real-world scenarios, hierarchical community structure and heterophily coexist in social networks. Our goal is to learn unified node representations that can incorporate both types of information simultaneously. However, to our knowledge, there are no methods that capture both types of information simultaneously. Therefore, we use three modules to obtain three representations, $\mathbf{h}_i^{\alpha}$, $\mathbf{h}_i^{\beta}$, and $\mathbf{h}_i^{\gamma}$, each containing different semantic information. Specifically, $\mathbf{h}_i^{\alpha}$ includes global hierarchical structure information, $\mathbf{h}_i^{\beta}$ includes local hierarchical structure information, and $\mathbf{h}_i^{\gamma}$ includes specific category information of the neighborhood.

However, representations obtained through independent modules are often *one-sided*. Therefore, we further use self-supervised contrastive learning to capture the consistency between different representations. Specifically, by maximizing the mutual information between different representations. The proposed two level contrastive learning losses $\mathcal{L}_{NCL}$ and $\mathcal{L}_{SCL}$ are computed based on the representations of all nodes (i.e., $\mathbf{H}^{\alpha}$, $\mathbf{H}^{\beta}$ and $\mathbf{H}^{\gamma}$) across different views:

$$\begin{aligned} \mathcal{L}_{NCL} &= InfoNCE(\psi_N(\mathbf{H}^{\alpha}), \psi_N(\mathbf{H}^{\gamma})), \\ \mathcal{L}_{SCL} &= InfoNCE(\psi_S(\mathbf{H}^{\beta}), \psi_S(\mathbf{H}^{\gamma})), \end{aligned} \tag{17}$$

where $\psi_N(\cdot)$ and $\psi_S(\cdot)$ are defined as projection heads for node-level and subgraph-level contrastive learning, respectively. In this way, two seemingly contradictory challenges are unified and addressed in a self-supervised manner, rather than being simply solved independently.

Ultimately, the overall loss of the proposed method is calculated by summing the aforementioned three learning losses:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{NCL} + \lambda_2 \mathcal{L}_{SCL}. \tag{18}$$

where λ_1 and λ_2 are two hyperparameters ranging from 0 to 1, used to adjust the magnitudes and weights of different losses.

4.6 Complexity Analysis

Given a mutli-relational graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$, the runtime complexity of Algorithm 1 is $O(h_{max}(mlog\ n + n))$, in which h_{max} is the height of coding tree T after the first step. In general, the coding tree T tends to be balanced in the process of structural entropy minimization, thus, h_{max} will be around $log\ n$. Besides, a large-scale social network generally has more edges than nodes, i.e., $m \gg n$, thus the runtime of Algorithm 1 almost scales linearly in the number of edges.

The overall time complexity of the three proposed modules is $O(n + m + h_{max}(m \log n + n))$. Specifically, in Section 4.3, the time complexities of SEP, SEP-U, and SEP-G are all $O(n)$. In Section 4.4, the proposed relational information aggregation has a time complexity of $O(L(R \cdot n + m))$, where L represents the number of layers and R represents the number of relations.

Algorithm 2: The training process of SeBot

Input : input undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ and a specific height $k > 1$, hyperparameter m, temperature τ and loss weights λ_1 and λ_2

Output : node representations $\mathbf{H}$

```
generate undirected graph view G^α and its m-hop subgraph view G^β ;
generate view G^γ through edge dropping ;
minimizing structural entropy to construct encoding trees for G^α and G^β according to Algorithm 1 ;
for epoch ← 1, 2, ··· do
    for v_i ∈ V do
        obtain node embedding h_i^α of v_i from G^α ← Equation (8-9) ;
        obtain node embedding h_i^β of v_i from G^β ← Equation (10) ;
        obtain node embedding h_i^γ of v_i from G^γ ← Equation (11-15) ;
    end
    classification loss L_CE ← Equation (16) ;
    contrastive loss L_NCL and L_SCL ← Equation (17) ;
    total loss L ← Equation (18) ;
    loss backward ;
end
return the predicted label set for the test nodes Ŷ_test .
```

5 EXPERIMENTS

In this paper, we propose the following research questions for a deep evaluation of the proposed SeBot framework:

- **RQ1:** How does SeBot perform compared with other baselines?
- **RQ2:** How does SeBot benefit from its different modules?
- **RQ3:** How does SeBot perform concerning different hyperparameters?
- **RQ4:** Can SeBot generate more class-discriminative node representations than other baselines?
- **RQ5:** Can SeBot effectively address the two challenges mentioned in the introduction?

5.1 Experimental Setup

5.1.1 Datasets. **TwiBot-20** [14] and **MGTAB** [37] are adopted to evaluate the performance of SeBot. **TwiBot-20** contains 229,580 accounts extracted from Twitter and their following and follower

interactions. **MGTAB** consists of 10,199 accounts and 7 types of relations including follower, friend, mentioned, reply, quoted, reference, and hashtag. Details of two datasets are shown in Table 1, in which **homo** represents the proportion of edges between nodes of the same class. In addition, we follow the same splits of datasets as [14] and [37] for training, validating, and testing.

5.1.2 Baselines. We compare SeBot with previous graph-based social bot detection methods as well as typical GNNs beyond homophily and self-supervised graph contrastive learning methods. All selected baselines are listed below:

- **GCN** [25] is a typical graph convolution network that can also be seen as a low-pass filter.
- **GAT** [40] lies on an attention-based information aggregation mechanism.
- **GraphSage** [19] is an inductive graph neural network capable of predicting node types that were not seen during the training process.
- **FAGCN** [4] can adaptively utilize neighbor representations to be similar or diverse by breaking the limitation of low-frequency information aggregation.
- **H2GCN** [58] introduces three simple designs: separating self and neighbor embeddings, high-order neighbor information, and concatenating node representations.
- **GPRGNN** [8] employs a learnable weight for each order of neighbor information.
- **Alhosseini et al.** [1] is the first to employ graph convolutional neural networks for detecting social bots.
- **FriendBot** [2] utilizes network, content, temporal, and user features obtained from the communication network, and employs machine learning classifiers for classification purposes.
- **RGT** [13] constructs heterogeneous views through various relations and proposes the Relation Graph Transformer to obtain node representations.
- **BotRGCN** [15] applies relational graph neural networks to perform social bot detection. **RGT** and **BotRGCN** are the current SOTA methods for social bot detection using GNNs.
- **DGI** [41] captures consistency between nodes and the entire graph by maximizing local-global mutual information.
- **GRACE** [59] employs feature masking and edge removal as graph augmentation techniques to generate views. It also uses a discriminator to encourage a uniform distribution.
- **GBT** [3] calculates the empirical cross-correlation matrix using node representations and computes the loss using the Barlow-Twins loss function.

5.1.3 Hyperparameter Setting. Hyperparameter settings of our experiments on TwiBot-20 and MGTAB are listed in Table 2. We used the AdamW optimizer to update the parameters. The learning rate is set to 0.01, which is larger compared to baseline models like RGT, resulting in faster convergence. The dropout mechanism is employed to prevent overfitting and maintain high generalization capacity. The weight of two contrastive loss λ_1 and λ_2 are set according to sensitive study results. The tree depth is set to 6, which is a compromise between training time and accuracy. Gumbel-Max reparametrization trick temperature τ_G is set to 0.01 to amplify edge attention.

Table 1: Statistics of TwiBot-20 and MGTAB.

Dataset	#nodes	#edges	class	#class	#relation	homo
TwiBot-20	229,580	227,979	human bot	5,237 6,589	2	0.53
MGTAB	10,199	1,700,108	human bot	7,451 2,748	7	0.84

Table 2: Hyerparameter setting on TwiBot-20 and MGTAB.

Parameter	T-20	MGTAB	Parameter	T-20	MGTAB
optimizer	AdamW	AdamW	hidden dimension	32	32
learning rate	0.01	0.01	subgraph order m	2	1
dropout	0.5	0.5	L2 regularization	3e-3	3e-3
loss weight λ_1	0.09	0.05	loss weight λ_2	0.03	0.05
tree depth k	6	6	maximum epochs	70	200
temperture τ	0.1	0.1	trick temperature τ_G	0.01	0.01

5.1.4 Implementation. Pytorch [32] and Pytorch Geometric [17]are leveraged to implement SeBot and other baselines. All experiments are conducted on a cluster with 8 GeForce RTX 3090 GPUs with 24 GB memory, 16 CPU cores, and 264 GB CPU memory. See our **code**[1] for more details.

5.2 RQ1: Performance Analysis

To answer **RQ1**, we evaluate the performance of SeBot and 11 other baselines on two social bot detection benchmarks. The experimental results are presented in Table 3, which illustrates that:

- SeBot demonstrates superior performance compared to all other baselines on both datasets in terms of Accuracy and F1-score on both datasets. Furthermore, it achieves relatively high results in terms of Recall and Precision on TwiBot-20 and MGTAB, indicating that SeBot is better at uncovering social bots and possesses stronger robustness. On the other hand, SeBot demonstrates the better generalization performance across both datasets, a feat that other methods cannot achieve.
- Compared to traditional GNNs (i.e., GCN, GAT, and GraphSage), SeBot not only considers the community structure but also is conscious of the adversarial structure intentionally constructed by social bots, thus exhibiting stronger detection performance. This also highlights the need for extra fine-grained designs when applying graph neural networks to social bot detection.
- Compared to GNNs beyond homophily (i.e., FAGCN, H2GCN, GPRGNN), SeBot extends further into the social bot detection scenario, specifically on multi-relation directed graphs. In addition, better performance also implies the significant importance of considering adversarial heterophily in social bot detection.
- Compared with state-of-the-art graph-based social bot detection methods (i.e., BotRGCN and RGT), SeBot achieves the best accuracy and F1-score as it further takes into account the structural semantics present in social networks, which proves to be potent for uncovering deeply concealed bots. Meanwhile, previous graph-based detection methods relied on traditional information aggregation patterns and could not fully capture the structural information within the graph.

[1] https://github.com/846468230/SEBot

Table 3: Performance comparison on TwiBot-20 and MGTAB in terms of accuracy, F1-score, recall and precision. The best and second-best results are highlighted with bold and underline.'-' indicates that the method is not applicable to MGTAB due to lack of unprocessed raw dataset.

METHODs	TwiBot-20				MGTAB				TYPE
	Accuracy	F1-score	Recall	Precision	Accuracy	F1-score	Recall	Precision	
GCN	77.53±1.73	80.86±0.86	87.62±3.31	75.23±3.08	80.07±0.77	51.71±4.05	75.07±5.93	40.08±5.90	CLASSIC
GAT	83.27±0.56	85.25±0.38	89.53±0.87	81.39±1.18	85.07±1.19	69.32±4.02	77.33±2.19	63.34±7.31	
GraphSage	85.44±0.43	86.68±0.61	87.66±2.08	**85.78±0.86**	88.58±1.11	77.55±2.00	82.43±2.39	73.32±3.12	
FAGCN	85.43±0.40	87.36±0.32	93.00±0.73	82.39±0.70	88.11±1.43	77.43±3.20	76.36±7.90	79.19±3.41	HETEROPHILY
H2GCN	85.84±0.34	87.57±0.15	92.19±1.56	83.44±1.32	89.09±1.16	79.99±1.53	81.00±5.94	79.70±5.01	
GPRGNN	86.05±0.34	87.50±0.30	90.25±0.29	84.92±0.41	89.07±1.20	80.48±1.62	**83.54±1.24**	77.67±2.37	
Alhosseini et al.	59.88±0.59	72.07±0.48	**95.69±1.93**	57.81±0.43	-	-	-	-	SOTAs
BotRGCN	85.75±0.69	87.25±0.74	90.19±1.72	84.52±0.54	89.09±0.61	79.66±0.82	80.22±3.07	79.39±4.00	
FriendBot	75.89±0.47	79.97±0.34	88.94±0.59	72.64±0.52	-	-	-	-	
RGT	86.57±0.42	88.01±0.42	91.06±0.80	85.15±0.28	89.00±1.35	79.26±2.87	78.51±6.25	80.48±2.93	
DGI	84.93±0.31	87.09±0.36	93.94±1.13	81.17±0.26	87.08±0.98	75.67 ±1.62	74.61±2.58	76.81±1.56	CONTRASTIVE
GBT	84.74±0.92	86.87±0.79	93.28±1.14	81.29±0.92	84.68±0.53	70.12±1.33	66.87±2.30	73.80±1.90	
GRACE	84.74±0.88	86.90±0.84	93.56±1.57	81.13±0.55	83.16±1.60	66.99±3.90	63.83±6.00	70.77±2.02	
SeBot	**87.24±0.10**	**88.74±0.13**	92.97±1.16	84.90±0.79	**90.46±1.44**	**82.12±2.42**	81.73±2.77	**82.52±2.19**	OURs

- Compared with typical self-supervised graph contrastive learning methods (i.e., DGI, GBT, and GRACE), SeBot retains essential information within the graph by minimizing structural entropy, while other graph augmentation methods may unavoidably introduce noise or lead to the loss of crucial information relevant to downstream tasks. It's important to mention that class imbalance in MGTAB significantly impacts the performance of contrastive learning techniques like DGI, leading to suboptimal results across various metrics.

5.3 RQ2: Ablation Study

To address **RQ2**, we conducted ablation experiments as follows. We separately removed the encoding tree of the entire graph, encoding trees of subgraphs and the proposed RCM layer, and evaluated the performance of the residual modules. We also substitute RGCN for the encoder, and adopt different policies of graph augmentation. The results are presented in Table 4. Removing different modules from SeBot resulted in performance degradation on TwiBot-20 and MGTAB datasets, indicating the pivotal role in overall model effectiveness. This emphasizes the importance of co-considering community structure and heterophilic relations in social bot detection. By comparison, graph augmentation, feature masking, or dropout may disrupt the original feature distribution structure, a critical step in node classification, meanwhile, adding edges may introduce some additional structure noise, and thus lead to model performance degradation.

5.4 RQ3: Sensitive Analysis

To answer **RQ3**, we conducted experiments on TwiBot-20 (MGTAB see in Appendix A.1) to analyze the impact of fixed encoding tree depth and contrastive losses. We set k to 6, 7, and 8, and varied hyperparameters λ_1 and λ_2 from 0.01 to 0.1 in steps of 0.01. The results of the experiments are presented in the form of heatmaps in Figure 3. From this, we can intuitively observe that increasing the depth of the tree improves the overall performance of the proposed model. A greater depth enables a finer-grained community partition, benefiting social bot detection, but it requires longer training time. Further, as the tree depth increases, the impact of hyperparameters λ_1 and λ_2 on model performance exhibits a fluctuating pattern, initially decreasing and then increasing. Notably, when $k = 7$, the model's accuracy shows minimal variation across different hyperparameters. However, increasing or decreasing k results in greater sensitivity of the accuracy to changes in the hyperparameters. This corresponds to our assertion in Section 4.2 that, in practical scenarios, a specific tree depth is generally preferred.

Table 4: Ablation study of SeBot on TwiBot-20 and MGTAB.

Settings	TwiBot-20		MGTAB	
	Accuracy	F1-score	Accuracy	F1-score
Full model	**87.24±0.10**	**88.74±0.13**	**90.46±1.44**	**82.12±2.42**
w/o entire graph tree	86.39±0.30	87.84±0.26	89.71±0.98	81.09±1.41
w/o subgraph trees	86.45±0.07	88.02±0.09	90.07±1.04	81.79±1.73
w/o RCM layer	86.24±0.49	87.69±0.55	89.58±1.52	80.51±2.63
RGCN as encoder	86.35±0.24	87.96±0.26	90.32±1.48	81.99±2.44
Feature Mask	84.93±0.67	87.00±0.70	90.00±1.28	81.59±1.85
Feature Dropping	83.39±0.61	85.53±0.92	89.36±1.48	80.58±1.27
Edge Adding	86.52±0.42	87.96±0.53	89.29±1.39	80.60±2.95

5.5 RQ4: Visualization

To address **RQ4**, we visually represent the 128-dimensional node embeddings generated by GCN, FAGCN, BotRGCN, RGT, GRACE, and SeBot on TwiBot-20 by projecting them onto a 2-dimensional space using T-SNE [39], as depicted in Figure 4. The embeddings from GCN and BotRGCN exhibit more scattering, whereas FAGCN, RGT, and our SeBot produce denser embeddings. GRACE embeddings are quite uniform but suffer from the class collapse issue, where nodes from different classes are located closely to each other [57] due to the absence of label information. It is worth noting that, compared to RGT and FAGCN, our method exhibits local

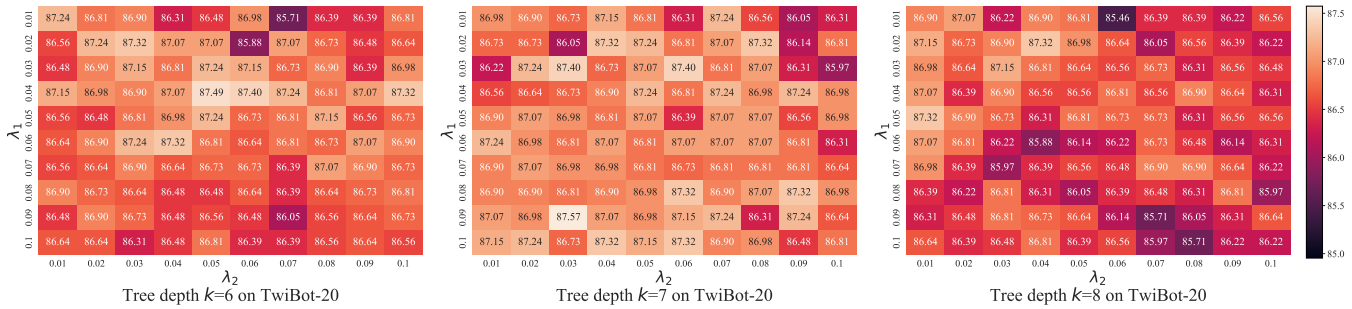
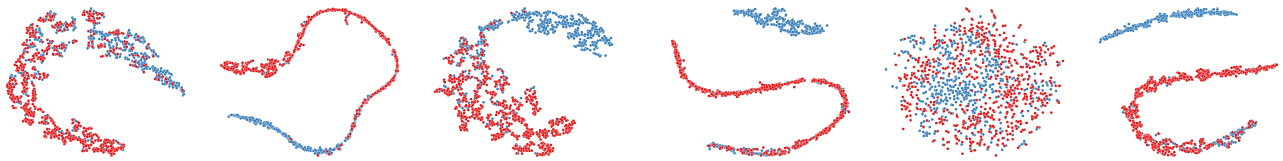


Figure 3: Sensitive analysis of hyperparameter λ_1 and λ_2 on TwiBot-20.



(a) GCN **(b) FAGCN** **(c) BotRGCN** **(d) RGT** **(e) GRACE** **(f) SeBot**

Figure 4: Account representations visualization on TwiBot-20. Red represents bots, while blue represents humans.

clustering rather than smooth curves visible in 2-dimensional space. This implies that nodes of similar features or belonging to the same community tend to aggregate together in clusters or beads. This also indicates the ability of our method to capture the inherent communities present in the graph structure. Overall, the embeddings generated by SeBot demonstrate relatively better class discrimination compared to other methods, and the inclusion of reparameterization techniques ensures that the representations avoid excessive clustering.

5.6 RQ5: Case Study

To answer **RQ5**, we visualize a local community consisting of 3 subcommunities and 7 social accounts, one of which is a bot account. Due to the employment of the reparameterization technique, edge weights generated adaptively are equal to or near 1 or -1, and edges of two different colors are to represent them, respectively. As shown in Figure 5, social networks contain both the edges between nodes of the same class and the edges between nodes of different classes. The edge weight adaptive mechanism proposed by us can simultaneously model both types of edges by enabling positive and negative edge weights. Furthermore, the fine-grained hierarchical community structure is obtained by minimizing structural entropy with a height constraint. Message passing on the encoding tree provides higher-order feature information favorable for classification.

6 CONCLUSION

In this paper, we propose SeBot, a novel graph-based social bot detection framework that takes into consideration both community structure and adversarial behaviors of social bots. SeBot addresses the aforementioned issues using three separate modules

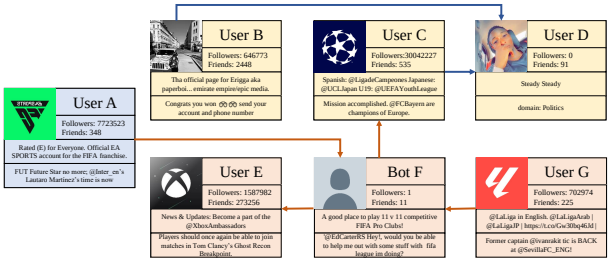


Figure 5: A case study of local community structure and generated edge attention. The same background color represents belonging to the same sub-community, with the same parent node on the constructed encoding tree.

that leverage structural entropy minimization and a heterophily-aware encoder. SeBot employs self-supervised contrastive learning to unify and learn the intrinsic characteristics of nodes more effectively. Comprehensive experiments show that SeBot exhibits superior generalizability and robustness on two real-world datasets compared with all other baselines.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China through the grants 2022YFB3104700, 2022YFB3105405, 2021YFC3300502, NSFC through grants 62322202 and 61932002, Beijing Natural Science Foundation through grant 4222030, Guangdong Basic and Applied Basic Research Foundation through grant 2023B1515120020, Shijiazhuang Science and Technology Plan Project through grant 231130459A.

REFERENCES

[1] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. 2019. Detect me if you can: Spam bot detection using inductive representation learning. In *WWW*. 148–153.

[2] David M Beskow and Kathleen M Carley. 2020. You are known by your friends: Leveraging network metrics for bot detection in twitter. *SMA* (2020).

[3] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. 2022. Graph Barlow Twins: A self-supervised representation learning framework for graphs. *KBS* 256 (2022), 109631.

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *AAAI*. 3950–3957.

[5] Nan Cao, Conglei Shi, Sabrina Lin, Jie Lu, Yu-Ru Lin, and Ching-Yung Lin. 2015. Targetvue: Visual analysis of anomalous user behaviors in online communication systems. *TVCG* 22, 1 (2015), 280–289.

[6] Yuwei Cao, Hao Peng, Angsheng Li, Chenyu You, Zhifeng Hao, and Philip S Yu. 2024. Multi-Relational Structural Entropy. In *UAI*.

[7] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. 2017. Temporal patterns in bot activities. In *WWW*. 1601–1606.

[8] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*. OpenReview.net.

[9] Stefano Cresci. 2020. A decade of social bot detection. *Commun ACM* (2020).

[10] Ashok Deb, Luca Luceri, Adam Badaway, and Emilio Ferrara. 2019. Perils and challenges of social media and election manipulation analysis: The 2018 us midterms. In *WWW*. 237–247.

[11] Nicolas Guenon des Mesnards, David Scott Hunter, Zakaria el Hjouji, and Tauhid Zaman. 2022. Detecting bots and assessing their impact in social networks. *Operations Research* 70, 1 (2022), 1–22.

[12] Shangbin Feng, Zhaoxuan Tan, and etc. Herun Wan. 2022. TwiBot-22: Towards Graph-Based Twitter Bot Detection. In *NeurIPS*.

[13] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo. 2022. Heterogeneity-aware twitter bot detection with relational graph transformers. In *AAAI*.

[14] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. Twibot-20: A comprehensive twitter bot detection benchmark. In *CIKM*.

[15] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. 2021. BotRGCN: Twitter bot detection with relational graph convolutional networks. In *SNAM*.

[16] Emilio Ferrara. 2017. Disinformation and social bot operations in the run up to the 2017 French presidential election. *First Monday* 22, 8 (2017).

[17] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv* (2019).

[18] Sami Abdullah Hamdi. 2022. Mining ideological discourse on Twitter: The case of extremism in Arabic. *Discourse & Communication* 16, 1 (2022), 76–92.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NIPS* 30 (2017).

[20] Buyun He, Yingguang Yang, Qi Wu, Hao Liu, Renyu Yang, Hao Peng, Xiang Wang, Yong Liao, and Pengyuan Zhou. 2024. Dynamicity-aware Social Bot Detection with Dynamic Graph Transformers. In *IJCAI*.

[21] Maryam Heidari, James H Jones, and Ozlem Uzuner. 2020. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *ICDMW*. IEEE, 480–487.

[22] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR (Poster)*. OpenReview.net.

[23] Edwin T Jaynes. 1980. The minimum entropy production principle. *ARPC* 31, 1 (1980), 579–601.

[24] Minguk Kang and Jaesik Park. 2020. Contragan: Contrastive learning for conditional image generation. *NIPS* 33 (2020), 21357–21369.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[26] Thai Le, Long Tran-Thanh, and Dongwon Lee. 2022. Socialbots on Fire: Modeling Adversarial Behaviors of Socialbots via Multi-Agent Hierarchical Reinforcement Learning. In *WWW*. 545–554.

[27] Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *TOIT* 62, 6 (2016), 3290–3339.

[28] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv* (2021).

[29] Luca Luceri, Ashok Deb, Adam Badawy, and Emilio Ferrara. 2019. Red bots do it better: Comparative analysis of social bot partisan behavior. In *WWW*. 1007–1012.

[30] Abbe Mowshowitz and Matthias Dehmer. 2012. Entropy and the complexity of graphs revisited. *Entropy* 14, 3 (2012), 559–570.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv* (2018).

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

[33] Hao Peng, Jingyun Zhang, Xiang Huang, Zhifeng Hao, Angsheng Li, Zhengtao Yu, and Philip S Yu. 2024. Unsupervised Social Bot Detection via Structural Information Theory. *TOIS* (2024).

[34] Nicolas Rashevsky. 1955. Life, information theory, and topology. *Bull. Math. Biol.* 17 (1955), 229–235.

[35] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.

[36] Claude Elwood Shannon. 1948. A mathematical theory of communication. *BSTJ* 27, 3 (1948), 379–423.

[37] Shuhao Shi, Kai Qiao, Jian Chen, Shuai Yang, Jie Yang, Baojie Song, Linyuan Wang, and Bin Yan. 2023. Mgtab: A multi-relational graph-based twitter account detection benchmark. *arXiv* (2023).

[38] Ernesto Trucco. 1956. A note on the information content of graphs. *Bull. Math. Biol.* 18 (1956), 129–135.

[39] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[40] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR (Poster)*. OpenReview.net.

[41] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR* 2, 3 (2019), 4.

[42] Patrick Wang, Rafael Angarita, and Ilaria Renna. 2018. Is this the era of misinformation yet: combining social bots and fake news to deceive the masses. In *WWW*. 1557–1561.

[43] Zixuan Weng and Aijun Lin. 2022. Public opinion manipulation on social media: Social network analysis of twitter bots during the covid-19 pandemic. *JERPH* 19, 24 (2022), 16376.

[44] Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. 2023. SEGA: Structural Entropy Guided Anchor View for Graph Contrastive Learning. In *ICML (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 37293–37312.

[45] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022. Structural entropy guided graph hierarchical pooling. In *ICML*. PMLR, 24017–24030.

[46] Qi Wu, Yingguan Yang, Buyun He, Hao Liu, Xiang Wang, Yong Liao, Renyu Yang, and Pengyuan Zhou. 2023. Heterophily-aware Social Bot Detection with Supervised Contrastive Learning. *arXiv* (2023).

[47] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph information bottleneck. *NIPS* 33 (2020), 20437–20448.

[48] Yuhao Wu, Yuzhou Fang, Shuaikang Shang, Jing Jin, Lai Wei, and Haizhou Wang. 2021. A novel framework for detecting social bots with deep neural networks and active learning. *KBS* 211 (2021), 106525.

[49] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. In *ACL/IJCNLP (1)*.

[50] Yingguang Yang, Renyu Yang, Yangyang Li, Kai Cui, Zhiqin Yang, Yue Wang, Jie Xu, and Haiyong Xie. 2023. Rosgas: Adaptive social bot detection with reinforced self-supervised gnn architecture search. *TWEB* (2023).

[51] Yingguang Yang, Renyu Yang, Hao Peng, Yangyang Li, Tong Li, Yong Liao, and Pengyuan Zhou. 2023. FedACK: Federated Adversarial Contrastive Knowledge Distillation for Cross-Lingual and Cross-Model Social Bot Detection. In *WWW*. 1314–1323.

[52] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NIPS* 33 (2020), 5812–5823.

[53] Guangjie Zeng, Hao Peng, Angsheng Li, Zhiwei Liu, Chunyang Liu, S Yu Philip, and Lifang He. 2023. Unsupervised Skin Lesion Segmentation via Structural Entropy Minimization on Multi-Scale Superpixel Graphs. In *ICDM*.

[54] Xianghua Zeng, Hao Peng, and Angsheng Li. 2023. Effective and stable role-based multi-agent collaboration by structural information principles. In *AAAI*.

[55] Xianghua Zeng, Hao Peng, and Angsheng Li. 2024. Adversarial socialbots modeling based on structural information principles. In *AAAI*.

[56] Xianghua Zeng, Hao Peng, Angsheng Li, Chunyang Liu, Lifang He, and Philip S Yu. 2023. Hierarchical state abstraction based on structural information principles. In *IJCAI*.

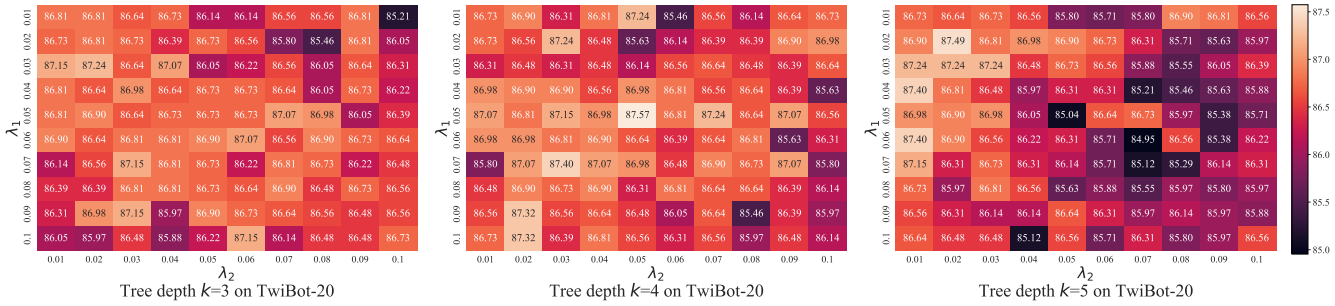
[57] Mingkai Zheng, Fei Wang, Shan You, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. 2021. Weakly supervised contrastive learning. In *ICCV*. 10042–10051.

[58] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *NIPS* 33 (2020), 7793–7804.

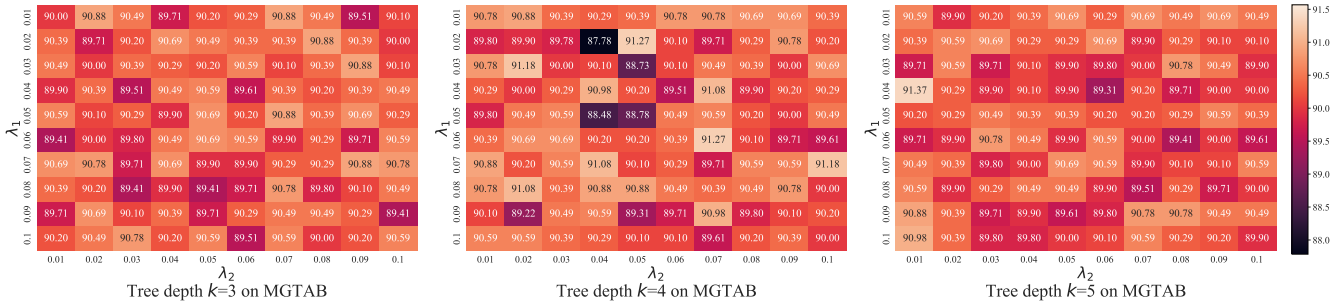
[59] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv* (2020).

[60] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.

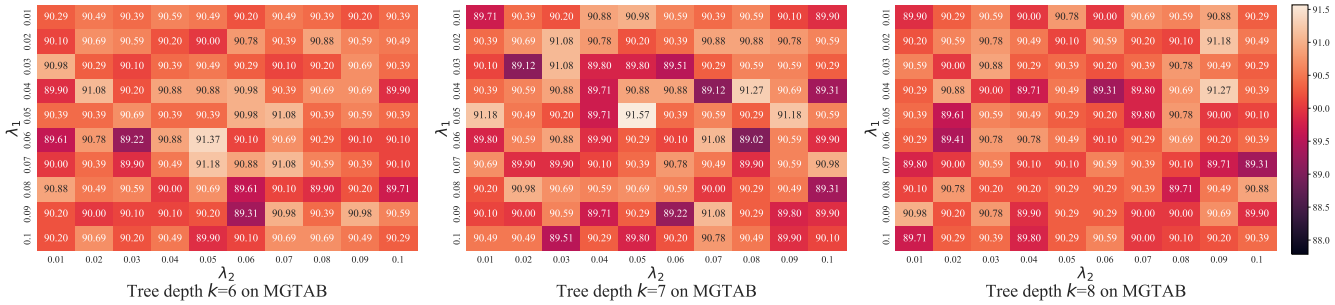
[61] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *WWW*. 499−−510.



(a) Sensitive analysis of hyperparameter λ_1 and λ_2 on TwiBot-20. (Tree depth 3-5)



(b) Sensitive analysis of hyperparameter λ_1 and λ_2 on MGTAB. (Tree depth 3-5)



(c) Sensitive analysis of hyperparameter λ_1 and λ_2 on MGTAB. (Tree depth 6-8)

Figure 6: Sensitive analysis of hyperparameter λ_1 and λ_2 on TwiBot-20 and MGTAB.

A APPENDIX

A.1 Sensitive Analysis Supplement

In this subsection, we provide additional experimental results on the impact of hyperparameters λ_1, λ_2 and tree depth k on accuracy, as shown in Figure 6. From Figure 6a, when k is set to 5, the trained model is more sensitive to changes in hyperparameters λ_1 and λ_2, and the impact is larger. When k=3, the overall performance is less affected by hyperparameters, and the differences in accuracy are smaller. We also conduct experiments on MGTAB in the same way, which is shown in Figure 6b and 6c. It is visually evident that the influence of hyperparameters is relatively small on MGTAB compared to TwiBot-20. Moreover, with the increase in tree depth, there isn't a significant overall improvement in performance. Even when k=3, satisfactory results can be achieved, indicating that the performance of SeBot on MGTAB is minimally affected by the granularity of community partitioning.

A.2 Data Efficiency Study

Current approaches to social bot detection predominantly follow a supervised paradigm, heavily reliant on an adequately rich set of annotated training data. However, acquiring such a dataset is a costly endeavor, and issues such as inaccurate annotations, noise, and insufficient richness are widespread. Consequently, it becomes imperative to evaluate the performance of SeBot under conditions of limited training data, relationships, and account features. To address this, we specifically design experimental conditions by training solely on a subset of the data, randomly removing edges, and masking partial features. The results are illustrated in Figure 7. Notably, even under the constraint of utilizing only 50% of the

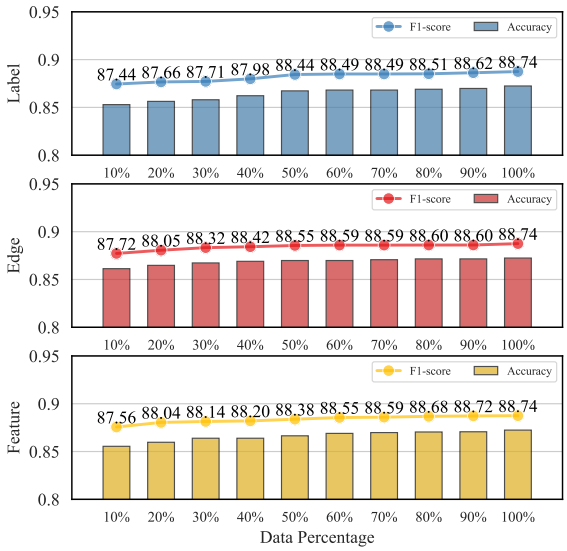


Figure 7: The experimental results of the data efficiency study regarding training data, edge quantity, and feature quantity.

training data, SeBot demonstrates superior performance compared to RGT [13], affirming its capability to mitigate dependence on data. Furthermore, our observations indicate that increased edges and features contribute to enhanced performance, suggesting that both factors play an advantageous role in detection."