Proactive Bot Detection Based on Structural Information Principles

Xianghua Zeng, Hao Peng, Angsheng Li

***Abstract*—Bot detection is crucial for combating misinformation and preserving the authenticity of online interactions on social media. However, the increasing sophistication of bots in mimicking genuine accounts and evading detection has created an ongoing arms race between detection systems and modeling techniques. In this paper, we propose a novel Structural Information principles-based Adversarial framework, namely SIAMD, designed to Model bot behaviors and achieve proactive Detection. This framework begins by organizing multi-relational interactions between user accounts and social messages into a unified heterogeneous structure, incorporating structural entropy to quantify the uncertainty inherent in historical activities. The high-dimensional entropy is then minimized to uncover a layered hierarchy within account communities, which facilitates activity determination and account selection in behavioral modeling for bot accounts. For each modeled bot and its selected account, SIAMD extracts historical messages and user descriptions to construct prompts and integrates large language models to generate the associated message content. By embedding synthetic message vertices and establishing multi-relational interactions within the original heterogeneous network, SIAMD achieves network evolution in both structure and content, thereby enhancing graph-based proactive detection in an adversarial manner. Extensive comparative experiments on well-established real-world datasets demonstrate that SIAMD significantly and consistently outperforms state-of-the-art detection baselines for social bots in terms of effectiveness, generalizability, robustness, and interpretability.**

***Index Terms*—Social bot detection, structural information principles, behavioral modeling, network evolution, large language models.**

I. Introduction

SOCIAL bots, automated accounts fully or partially controlled by software to post content and interact with others in ways that mimic human behavior, have become a prominent phenomenon in online networks [1]–[3]. By disseminating unreliable or malicious information, social bots exacerbate confusion and undermine public discourse on critical societal issues such as online disinformation [4]–[6], election interference [7]–[9], and extremist ideologies [10], [11]. Therefore, developing effective detection models is crucial to mitigate their negative influence and preserve information credibility on social networks. Recent advancements in generative technology have enabled increasingly sophisticated content generation and more human-like interactions, further intensifying the competition between detection systems and modeling approaches [12]–[14].

Traditional detectors primarily rely on feature-based models that extract account metadata [15], user timelines [16], and follower relationships [17]. However, these detectors are vulnerable to adversarial manipulations, as bots can modify the manually designed features to evade detection by exploiting predictable feature selection patterns [18], [19]. Alternatively, content-based approaches have emerged, using pre-trained linguistic models to analyze textual elements and identify malicious purposes [20]–[22]. Yet, the strategy of bots appropriating content from human accounts and mixing it with their own fabricated messages leads to detection inaccuracies, such as misclassifications and false negatives [23], [24]. The advent of graph neural networks (GNNs) [25], [26] has enabled graph-based methods to better capture information flows between accounts, facilitating the identification of complex relationships and improving bot detection accuracy [23], [27], [28]. Despite these advances, a growing body of work has revealed that GNNs are highly vulnerable to adversarial perturbations on node features and graph structure [29]–[31], and often struggle in low-homophily or heterogeneous settings that are characteristic of multi-bot ecosystems [32]. Addtionally, most graph-based methods for bot detection still exhibit a significant weakness: they overlook the evolving nature of social bots, passively reacting to evasion tactics [19], [33], [34]. In particular, their detection performance is severely compromised by the absence of training data on unforeseen bot behaviors, resulting in substantial performance degradation under adaptive attacks.

To move beyond reactive mechanisms, researchers are exploring proactive detection techniques that explicitly model adversarial evolution. On the one hand, reinforcement learning (RL) has been used to simulate adversarial bot behaviors and control strategies. For example, hierarchical multi-agent RL has been proposed to coordinate the activities of socialbots so as to maximize their influence while remaining undetected [35], and related RL-based influence maximization and rumor mitigation models on social graphs demonstrate the ability of RL to capture long-term, sequential effects of actions on network dynamics [36], [37]. On the other hand, adversarial influence models formulate attacks on GNNs directly as optimization problems over diffusion processes on graphs, e.g., by casting adversarial attacks as influence maximization [38], further underscoring the vulnerability of graph-based detectors to adaptive structural perturbations. Generative adversarial networks (GANs) [39] have also been applied to bot detection,

Xianghua Zeng and Angsheng Li are with State Key Laboratory of Software Development Environment, the School of Computer Science and Engineering, Beihang University, Beijing 1000191, China. E-mail: {zengxianghua, angsheng}@buaa.edu.cn;

Hao Peng is with the School of Cyber Science and Technology, Beihang University, Beijing 1000191, China. E-mail: penghao@buaa.edu.cn.

Manuscript received January 2025. (Corresponding author: Hao Peng.)

producing synthetic samples that anticipate bot evolution and facilitate proactive detection [33], [40], [41]. Subsequently, conditional adversarial learning frameworks, such as CALEB [42], were introduced to improve multi-class bot detection by generating synthetic instances for different bot types and using the discriminator as an enhanced detector. While these GAN-based and adversarial augmentation methods emphasize the generation of realistic or diversified bot representations, they typically operate as black-box models, providing limited interpretability and only coarse control over how structural and content-level behaviors co-evolve over time. Thus, developing accurate and interpretable models of multi-bot behaviors that jointly capture both network structure and message content remains a pressing challenge for effective proactive detection.

Different from traditional information theory [43], which quantifies transmission efficiency in communication systems, structural information principles [44] employ structural entropy to measure dynamic uncertainty in complex networks through a hierarchical partitioning strategy. This strategy represents the network structure as an encoding tree, where each node[1] corresponds to a community that includes a subset of network vertices. In this work, we propose a novel framework, **SIAMD**, which leverages network **S**tructural **I**nformation to **A**dversarially **M**odel the behaviors of multiple bots in both structure and content, thereby enabling proactive **D**etection with enhanced effectiveness and generalizability. Figure 1 illustrates the detailed modeling and detection process within the SIAMD framework, featuring two human accounts and two bot accounts. The graph-based detector initially misclassifies one bot account due to the absence of distinct bot-like patterns. The SIAMD predicts the account's future behavior and generates corresponding message vertices to simulate network evolution, thereby enabling accurate proactive classification.

Specifically, we begin by extracting account information from historical messages and organizing the multi-relational interactions into a unified heterogeneous graph. To analyze account relationships further, we employ meta-path instances-specific sequences of vertex types and relations-to transform the heterogeneous graph into a multi-relational account graph. Next, we define multi-relational structural entropy as the minimum number of bits required to encode accessible accounts through random interactions within the account graph. By minimizing the high-dimensional entropy of the account graph, we then derive an optimal encoding tree that represents a hierarchical community structure formed by frequent account interactions. Within each community, we leverage multi-relational entropy to measure network influence and behavioral relevance, facilitating activity determination and account selection in behavioral modeling for bots. Subsequently, we integrate the historical messages and user descriptions of each bot and its selected accounts to formulate prompts, which are processed by large language models to generate associated message content. In the heterogeneous network, we embed the generated message vertices and establish interactions linking them to modeled bots and selected accounts, simulating network evolution in both structure and content. Finally, we

[1]Vertices are defined in the graph, and nodes are in the tree.

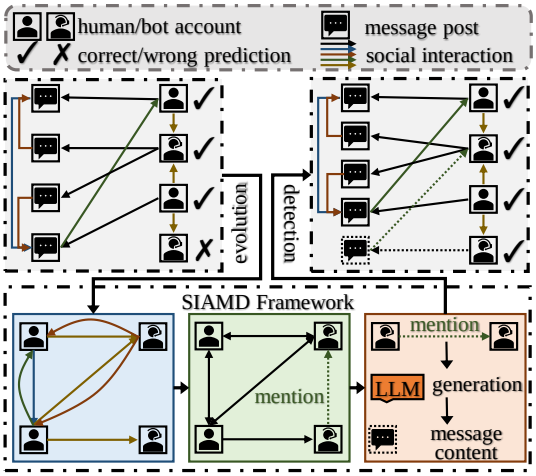

Fig. 1. The SIAMD framework for proactive multi-bot detection. SIAMD measures structural entropy on a multi-relational social interaction graph of human and bot accounts, and couples this with a large language model–driven message generator to simulate future structural and content evolution, improving the robustness and proactivity of graph-based detectors.

design an adversarial mechanism that leverages pretrained graph-based detectors for behavioral modeling. This mechanism minimizes the prediction probabilities of bot accounts and optimize the detectors on evolved networks to enhance their proactive detection capabilities.

In this work, we conduct extensive comparative analyses on well-established real-world datasets, including Cresci-2015 [45], Cresci-2017 [46], TwiBot-20 [47], and TwiBot-22 [24], to comprehensively evaluate the detection performance of SIAMD framework. The overall experimental results demonstrate the superior effectiveness of SIAMD in bot detection compared to state-of-the-art baseline methods. To evaluate generalizability, we analyze performance variations across detection methods when trained and tested on different subcommunities of the TwiBot-22 dataset. Furthermore, when tested against a large model-driven attack algorithm [48], our framework outperforms other detectors by maintaining stable detection performance, highlighting its robustness. Visualizations of the evolved network structure emphasize the interpretability of our framework, offering insights into the progression of bot behaviors over time.

Our preliminary work was accepted and presented at the AAAI Conference 2024 [49]. This journal version expands the original single-bot modeling method, transforming it into a more comprehensive, adversarial, and interpretable framework for multi-bot behavioral modeling and proactive detection. These enhancements significantly improve the methodology and structure of the framework. First, we define multi-relational structural entropy to measure the uncertainty in heterogeneous interactions directly, eliminating the previous reliance on learned account features. Second, we replace the original single-agent approach with multi-agent reinforcement learning algorithms, facilitating behavioral modeling for multiple bots. Third, apart from behavioral modeling within the network structure, we extract historical information from modeled bots and their selected accounts to construct prompts for large language models, providing a detailed perspective on the bot behaviors in both structure and content. Finally, utilizing the evolved networks, we design an adversarial

TABLE I
MULTI-RELATIONAL INTERACTION TYPES BETWEEN ACCOUNT AND MESSAGE VERTICES IN THE HETEROGENEOUS SOCIAL NETWORK.

Type	Source	Target	Description
post	account	message	account u_i posts message m_j
retweet	message	message	message m_i retweets message m_j
mention	message	account	message m_i mentions account u_j
reply	message	message	message m_i replies message m_j
follow	account	account	account u_i follows account u_j

optimization mechanism to further enhance the performance of existing graph-based detectors, improving their proactive detection capabilities.

The main contributions of this paper are as follows:

• A novel structural information principles-based adversarial framework, SIAMD, is proposed for behavioral modeling and proactive detection of social bots.

• Multi-relational structural entropy quantifies the uncertainty of social interactions in heterogeneous networks, enabling activity determination and account selection for multi-bot behavioral modeling.

• A large language model generates content from user descriptions and historical messages, aiding in content evolution for social networks.

• An adversarial mechanism optimizes graph-based detectors in evolved social networks, enhancing their proactive detection capabilities.

• Extensive evaluations on public datasets demonstrate the SIAMD framework's detection advantage in terms of effectiveness, generalizability, robustness, and interpretability.

This paper is organized as follows: Section II outlines the preliminaries and notation, Section III introduces the adversarial detection architecture, Section IV describes the detailed design of SIAMD framework, Sections V and VI present the experimental setup and evaluation, and Section VII discusses related work, followed by the conclusion in Section VIII.

II. PRELIMINARIES AND NOTATIONS

In this section, we offer formal definitions for key concepts, including **Heterogeneous Social Network**, **Multi-Relational Account Graph**, **Bot Detection**, **Bot Modeling**, **Encoding Tree**, and **Structural Entropy**.

Definition 2.1: A **heterogeneous social network** is modeled as a directed graph $G_h = (M, U, X_m, X_u, E_h)$, where M denotes the set of social messages, U represents the set of user accounts, and E_h captures the multi-relational interactions between these entities. The feature matrices X_m and X_u correspond to the content embeddings of messages and the description embeddings of user accounts, respectively. Each graph vertex refers to either a message $m \in M$ or an account $u \in U$ with a d-dimensional content embedding $x_m \in X_m$ or description embedding $x_u \in X_u$. The different interaction types, including their source and target vertices, as well as detailed descriptions, are outlined in Table I.

Definition 2.2: By extracting account information and social interactions, we transform the heterogeneous network G_h into a **multi-relational account graph**, represented as the tuple $G_m = (U, X_u, \{E_r^w\}_{r\in R}, W)$. Here, $e_{i,j}^r \in E_r^w$ represents a directed edge between accounts u_i and u_j, associated with the relation $r \in R$ and a weight $w_{i,j}^r \in W$. Specifically, an edge $e_{i,j}^r$ indicates the message transmission from account u_i to account u_j through the relation type $r \in R$, allowing u_i to influence u_j.

Definition 2.3: In a graph-based **bot detection** framework f_θ, for each user account $u \in U$, a graph neural network (GNN) recursively aggregates feature information from its neighboring vertices to compute a hidden representation h_u of the account. This hidden representation is then mapped to a binary label $\hat{y} \in \{0, 1\}$. The cross-entropy loss for the account set U is defined as follows:

$$\mathcal{L}_{det}\left(G_h, f_\theta\right) = \sum_{u_i \in U} \left[-\log\left(y_i \cdot \sigma\left(f_\theta\left(X_u, E_h\right)_{u_i}\right)\right)\right], \quad (1)$$

where σ denotes the sigmoid function applied for the binary classification of user accounts.

Definition 2.4: In a **bot modeling** framework, a social bot is a vertex in G_h that attempts to mimic human-like behavior in order to maximize the spread of propaganda or low-credible content, while simultaneously evading detection by the pre-trained graph-based detector f_{θ^*}, whose parameters θ^* are inaccessible to the bot, as defined by:

$$\theta^* = \arg\min \mathcal{L}_{det}\left(G_h, f_\theta\right). \quad (2)$$

Definition 2.5: In the context of structural information principles, the **encoding tree** of an irreducible, homogeneous graph $G = (V, E, W)$ is a rooted tree T as follows:

• The root node λ in T corresponds to the entire vertex set V, denoted by $V_\lambda = V$.

• Each leaf node ν in T corresponds to a singleton containing a single vertex $v \in V$, denoted by $V_\nu = \{v\}$.

• Each non-root and non-leaf node α in T corresponds to a vertex subset V_α, where $V_\alpha \subseteq V$.

• For each non-root node α in T, its parent node is denoted by α^- with $V_\alpha \subset V_{\alpha^-}$.

• For each non-leaf node α in T, its children are numbered as L_α, and the i-th child is denoted by α_i, ordered from left to right in increasing value of i.

• For each non-leaf node α in T, the vertex subsets V_{α_i} are disjointed, and $V_\alpha = \bigcup_{i=1}^{L_\alpha} V_{\alpha_i}$.

Definition 2.6: The **one-dimensional structural entropy** quantifies the minimum number of bits required to encode the accessible vertices encountered in a single-step random walk on G, assuming no any partitioning structure. This entropy is equivalent to the Shannon entropy of the stationary distribution over vertex degrees, and is defined as follows:

$$\mathcal{H}^1(G) = -\sum_{v\in V}\left[\frac{d_v}{\mathcal{V}_\lambda}\cdot\log_2\frac{d_v}{\mathcal{V}_\lambda}\right], \quad \mathcal{V}_\lambda = \sum_{v\in V} d_v, \quad (3)$$

where $\mathcal{V}_\lambda$ denotes the total degree sum of all vertices in the set V_λ. The dynamical uncertainty inherent in G is upper bounded by $\mathcal{H}^1(G)$, and an encoding tree provides a hierarchical partitioning strategy that effectively eliminates this uncertainty.

Definition 2.7: Given an encoding tree T with height at most K, the ***K*-dimensional structural entropy** quantifies the

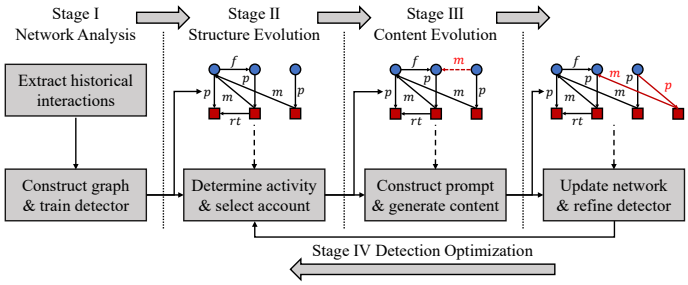

Fig. 2. Adversarial multi-stage detection architecture of SIAMD. The framework operates on a heterogeneous social network and comprises four stages: (I) social network analysis and black-box GNN pre-training, (II) network structure evolution via structural information–driven multi-bot behavior modeling, (III) network content evolution via LLM-based message generation and graph augmentation, and (IV) adversarial detector optimization on the evolved network.

remaining uncertainty in G. The structural entropy assigned to each non-root node α in T is defined as follows:

$$\begin{aligned} \mathcal{H}^T(G;\alpha) &= -\frac{g_\alpha}{\mathcal{V}_\lambda}\log_2\frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \\ g_\alpha = \sum_{v_i\notin V_\alpha}\sum_{v_j\in V_\alpha} w_{i,j}, &\quad \mathcal{V}_\alpha = \sum_{v\in V_\alpha} d_v, \end{aligned} \tag{4}$$

where g_α represents the sum of the weights of all edges connecting vertices in V_α to vertices outside V_α, and $\mathcal{V}_\alpha$ denotes the sum of degrees of vertices in V_α. The K-dimensional structural entropy of G is then defined as follows:

$$\mathcal{H}^K(G) = \min_T \left\{\mathcal{H}^T(G)\right\}, \quad \mathcal{H}^T(G) = \sum_{\alpha\neq\lambda}\mathcal{H}^T(G;\alpha). \tag{5}$$

III. Adversarial Detection Architecture

To address the challenges of proactive bot detection, our SIAMD framework employs an adversarial optimization mechanism that iteratively refines both the modeling and detection capabilities for social bots. Specifically, the SIAMD architecture consists of four primary stages: social network analysis, network structure evolution, network content evolution, and bot detection optimization. These are denoted as Stage I, Stage II, Stage III, and Stage IV in Figure 2.

During the network analysis stage, we extract historical interactions of various types between user accounts and social messages to construct a heterogeneous network. We then pre-train a graph neural network distinguish bot accounts from human accounts, which will serve as the black-box detector in subsequent evolution stages.

During the structure evolution stage, we leverage structural information inherent in heterogeneous interactions to analyze the network influence and behavioral relevance of user accounts, and model the future behaviors of multiple bot accounts. In this stage, we define two behavioral objectives for bot accounts: (1) to minimize the detection probability by evading the black-box detection system, and (2) to maximize the spread of their messages across accounts to amplify their network influence.

During the content evolution stage, we parse the bot account, interaction type, and target account for each modeled behavior to construct prompts and utilize large language models to generate message content associated with the modeled behavior. In the heterogeneous network, we add the generated message as a new vertex and connect it with the modeled bot and targeted account based on the interaction type, thereby updating both the network structure and its content.

During the detection optimization stage, we fine-tune the graph-based bot detector on the updated heterogeneous network to maximize its prediction probability of identifying bot accounts, in an adversarial manner aligned with the objectives of behavioral modeling. After optimization in each iteration, the optimized graph-based model is used as the black-box detector in the behavioral modeling of the next iteration, thereby gradually improving proactive detection performance.

IV. The Proposed SIAMD Framework

This section provides an overview of our proposed SIAMD framework, comprising four primary modules: social network analysis, network structure evolution, network content evolution, and adversarial bot detection.

In the network analysis module, we organize the multi-relational interactions between account and message elements into a unified graph structure. We then utilize meta-path instances to explore account relationships and quantify interaction uncertainties, which are critical for understanding the network dynamics. In the structure evolution module, we reduce the dynamic uncertainty in multi-relational interactions by applying hierarchical partitioning strategies to derive a community structure of user accounts. This approach simultaneously captures and models the behavioral patterns of bot accounts accurately. In the content evolution module, we leverage the historical activities of each bot and its selected account, employing large language models to simulate realistic message generation and enable the evolution of network content. In the detection optimization module, we introduce an adversarial mechanism to iteratively optimize the above network evolution and improve the accuracy of graph-based detection, as detailed in Section III. A more detailed design of the SIAMD framework is illustrated in Figure 3.

A. Social Network Analysis

To preserve the original information embedded in historical interactions between social messages and user accounts, we organize these social elements into a unified heterogeneous graph. This structure integrates various types of vertices (representing accounts and messages) and edges (indicating different types of interactions) to accurately reflect the complexity of social interactions. By defining typical sequences of relations between these vertex types as meta-path instances, we model multi-type relationships between accounts within a multi-relational graph structure. Additionally, we embed account descriptions and message content to obtain representations that capture both the essential semantic and structural features of accounts and messages.

The social network in Figure 3 is represented as a heterogeneous graph, G_h, extracted from historical interactions between social messages M and user accounts U. The edge set E_h captures common interactions that users engage in, including *post*, *retweet*, *mention*, *reply*, and *follow*, as detailed

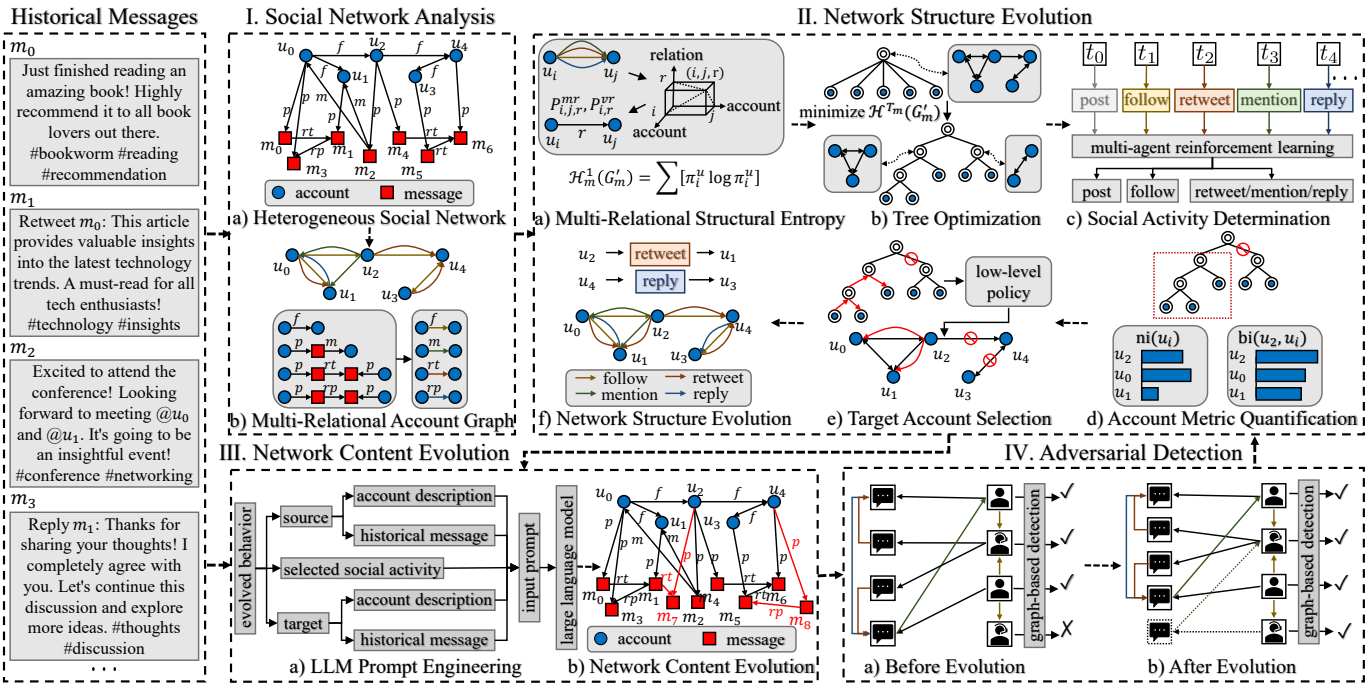

Fig. 3. Detailed design of the proposed SIAMD framework. The framework operates on a heterogeneous social network and instantiates four modules: social network analysis via multi-relational graph construction and meta-path–based interaction modeling, network structure evolution via structural information–driven community partitioning, network content evolution via LLM-based message generation, and adversarial optimization of a graph-based bot detector.

in Table I. For each message $m \in M$, we use a pre-trained text embedding model, such as RoBERTa [50], to convert the original message text into a dense vector representation $x_m \in X_m$. For each account $u \in U$, we separately extract its categorical and numerical features from the user descriptions, and concatenate them into a comprehensive feature representation $x_u \in X_u$.

To retain the heterogeneous information present among social elements, we map the network G_h to a non-negative weighted multi-relational graph $G_m = (U, X_u, \{E_r^w\}_{r\in R}, W)$ that represents account vertices. If message vertices are associated with accounts u_i and u_j through distinct interactions in G_h, corresponding multi-relational edges are established in G_m to reflect these interactions. Specifically, we define meta-paths as specific sequences of vertex types and edge types in G_h, which enable the construction of the relation set $R = \{f, m, rt, rp\}$. The distinct edge sets under each type of relation are formulated as follows:

$$\begin{aligned} E_f^w &= \{e_{i,j}^f || A_f|_{i,j} > 0\}, \\ E_m^w &= \{e_{i,j}^m || A_p \cdot A_m|_{i,j} > 0\}, \\ E_{rt}^w &= \{e_{i,j}^{rt} || A_p \cdot A_{rt} \cdot A_p|_{i,j} > 0\}, \\ E_{rp}^w &= \{e_{i,j}^{rp} || A_p \cdot A_{rp} \cdot A_p|_{i,j} > 0\}, \end{aligned} \tag{6}$$

where A_f, A_p, A_m, A_{rt}, A_{rp} denote the adjacency matrices of the heterogeneous network G_h for the relations *follow*, *post*, *mention*, *retweet*, and *reply*, respectively. For each edge $e_{i,j}^r \in E_r^w$, we assign a normalized value as its weight, calculated as:

$$w_{i,j}^r = \begin{cases} \frac{2\cdot|A_f|_{i,j}}{\mathrm{sum}(A_f)}, & r \text{ belongs to } \textit{follow}, \\ \frac{2\cdot|A_p\cdot A_m|_{i,j}}{\mathrm{sum}(A_p\cdot A_m)}, & r \text{ belongs to } \textit{mention}, \\ \frac{2\cdot|A_p\cdot A_{rt}\cdot A_p|_{i,j}}{\mathrm{sum}(A_p\cdot A_{rt}\cdot A_p)}, & r \text{ belongs to } \textit{retweet}, \\ \frac{2\cdot|A_p\cdot A_{rp}\cdot A_p|_{i,j}}{\mathrm{sum}(A_p\cdot A_{rp}\cdot A_p)}, & r \text{ belongs to } \textit{reply}, \end{cases} \tag{7}$$

where $\mathrm{sum}(\cdot)$ denotes the sum operation of all elements within the corresponding adjacency matrix.

B. Multi-Relational Structural Entropy

Applying the homogeneous structural information principle [44] separately to each type of user relationship and then summing the resulting uncertainties implicitly treats these relationships as independent. This independence assumption prevents the model from capturing the joint effects and interdependencies among different social relationship types. To overcome this limitation, we define a random walk on the multi-relational, weighted account graph, in which transition probabilities are determined jointly by all available relationship types. This yields a single Markov chain over users with a unique stationary distribution, from which we derive a one-dimensional measure of multi-relational structural entropy. We then develop an optimization algorithm that minimizes this entropy by applying operators on the encoding tree, enabling the model to exploit interactions among relations while preserving a compact structural representation.

1) Transition Probability Quantification: For the multi-relational account graph G_m, we first apply an adjustment algorithm to ensure strong connectivity within each single-relational subgraph G_r^m. Specifically, for each relation type $r \in R$, we require that for any pair of accounts there exists a directed path from one to the other under relation r. To achieve this, we extract each single-relational subgraph G_r^m (line 4 in Algorithm 1), compute its strongly connected components C_r (line 5 in Algorithm 1), and then add directed edges with small weights between these components to form a directed cycle (lines 6 - 10 in Algorithm 1). This guarantees that every G_r^m is strongly connected after adjustment.

We denote the adjacency tensor of the adjusted graph G'_m by $A_m \in \mathbb{R}^{|U|\times|U|\times|R|}$, where each entry $A_{i,j,r}^m$ represents the

Algorithm 1 Multi-relational graph adjusting.

Input: the non-negative, multi-relational graph G_m
Output: the strong-connected graph G'_m
for each relation $r \in R$ **do**
  $G_r^m \leftarrow$ extract single-relational subgraph under each relation $r \in R$
  $C_r \leftarrow$ identify strongly connected components of the graph G'_m
  if $\text{len}(C_r) > 1$ **then**
    for $i \in [0, \text{len}(C_r) - 1]$ **do**
      $j \leftarrow (i+1) \mod \text{len}(C_r)$
      if no edges from C_i^r to C_j^r **then**
        $G_r^m \leftarrow$ add an edge with a small weight ϵ from C_i^r to C_j^r
      end if
    end for
  end if
end for
return the strongly-connected graph G'_m

non-negative weight of the directed edge from account $u_i \in U$ to $u_j \in U$ under relation $r \in R$. During a single-step random walk on G'_m, the probability of moving from u_i to u_j via relation r is decomposed as follows:

$$p(u_i, r, u_j) = p(u_j|r, u_i)p(r|u_i)p(u_i), \tag{8}$$

where $p(u_j \mid r, u_i)$ denotes the probability of transitioning to u_j given current account u_i and relation r, $p(r \mid u_i)$ is the probability of selecting relation r at u_i, and $p(u_i)$ is the prior probability of being at account u_i. Thus, different edge types are combined into a single Markov chain via the relation-selection probabilities $p(r \mid u_i)$ and the relation-specific neighbor distributions $p(u_j \mid r, u_i)$, rather than being treated independently.

To instantiate the conditional probabilities in Equation 8, we construct two transition tensors, $P_{mr} \in \mathbb{R}^{|U|\times|U|\times|R|}$ and $P_{vr} \in \mathbb{R}^{|U|\times|R|}$, defined as follows:

$$P_{i,j,r}^{mr} = A_{i,j,r}^m / \sum_{u_k \in U} A_{i,k,r}^m, \tag{9}$$

$$P_{i,r}^{vr} = \sum_{u_j \in U} \left[A_{i,j,r}^m / \sum_{r' \in R} A_{i,j,r'}^m \right], \tag{10}$$

where $P_{i,j,r}^{mr}$ and $P_{i,r}^{vr}$ quantify the conditional probabilities $p(u_j \mid r, u_i)$ and $p(r \mid u_i)$, respectively. By construction, these tensors satisfy the following properties:

$$\sum_{u_j \in U} p(u_j|r, u_i) = \sum_{u_j \in U} P_{i,j,k}^{mr} = 1, \tag{11}$$

$$\sum_{r \in R} p(r|u_i) = \sum_{r \in R} P_{i,r}^{vr} = 1, \tag{12}$$

ensuring that P_{mr} and P_{vr} define valid probability distributions along the corresponding dimensions.

Leveraging the transition tensors P_{mr} and P_{vr}, we derive the transition matrix P that governs random walks among accounts:

$$P_{i,j} = \sum_r \left[P_{i,r}^{vr} P_{i,j,r}^{mr} \right], \tag{13}$$

so that the entry $P_{i,j}$ corresponds to the transition probability $p(u_j \mid u_i)$:

$$p(u_j|u_i) = \sum_r \left[p(r|u_i) p(u_j|r, u_i) \right]. \tag{14}$$

The above formulation can be viewed as a two-step decision process at each account. When the random walk is at account u_i, it first chooses which type of social relation to follow according to $p(r \mid u_i)$. After the relation type r is selected, it then chooses a specific neighbor u_j connected to u_i by relation r with probability $p(u_j \mid r, u_i)$. Thus, the overall transition probability from u_i to u_j aggregates all relation types through this "first pick a relation, then pick a neighbor" mechanism, allowing the model to capture how different relations jointly shape the flow on the multi-relational graph.

The following theorem establishes the existence and uniqueness of the stationary distribution π_u for random walks on accounts in G'_m.

Theorem 1: Given the transition matrix P for random walks among accounts in G'_m, there exists a unique stationary distribution π_u. This distribution coincides with the (normalized) eigenvector associated with the maximal eigenvalue 1.

2) Multi-Relational Entropy Definition: For an irreducible, homogeneous graph $G = (V, E, W)$, the one-dimensional structural entropy $\mathcal{H}^1(G)$ can be expressed in terms of the stationary distribution π over the vertices V as follows:

$$\mathcal{H}^1(G) = \sum_{v_i \in V} \left[-\pi_i \log \pi_i \right], \tag{15}$$

where π_i is the stationary probability of vertex $v_i \in V$.

Analogously, we define the one-dimensional multi-relational structural entropy of G'_m using the stationary distribution π_u over the accounts U:

$$\mathcal{H}_m^1(G'_m) = \sum_{u_i \in U} \left[\pi_i^u \log \pi_i^u \right], \tag{16}$$

where π_i^u denotes the stationary probability of account $u_i \in U$ in the distribution π_u.

Based on the transition tensors P_{mr} and P_{vr}, which capture the relational dynamics within G'_m, we adapt the terms g_α and $\mathcal{V}_\alpha$ to reflect the multi-relational random walk. This yields the following definition of the assigned entropy $\mathcal{H}_m^{T_m}(G'_m; \alpha)$:

$$\mathcal{H}_m^{T_m}(G'_m; \alpha) = -\frac{g_\alpha}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{17}$$

$$\mathcal{V}_\alpha = \sum_{u_i \in U} \sum_{u_j \in U_\alpha} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right], \tag{18}$$

$$g_\alpha = \sum_{u_i \notin U_\alpha} \sum_{u_j \in U_\alpha} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right]. \tag{19}$$

In the multi-relational setting, $\mathcal{V}_\alpha$ and g_α can be interpreted in terms of the random-walk flow induced by all relation types. The quantity $\mathcal{V}_\alpha$ measures the total stationary flow that enters

Algorithm 2 High-dimensioanl multi-relational structural entropy optimization.

1: **Input:** the initial encoding tree T_m, the maximal tree height $K \in \mathbb{Z}^+$
2: **Output:** the K-layer optimal encoding tree T_m^*
3: **while** height of T_m is less than K **do**
4: $(\alpha^*, \beta^*) \leftarrow \arg\max\{\Delta_{mg}(T_m, \alpha, \beta)\}$
5: **if** $\Delta_{mg}(T_m, \alpha^*, \beta^*) > 0$ **then**
6: $T_m \leftarrow \eta_{mg}(T_m, \alpha^*, \beta^*)$
7: continue
8: **end if**
9: $(\alpha^*, \beta^*) \leftarrow \arg\max\{\Delta_{cb}(T_m, \alpha, \beta)\}$
10: **if** $\Delta_{cb}(T_m, \alpha^*, \beta^*) > 0$ **then**
11: $T_m \leftarrow \eta_{cb}(T_m, \alpha^*, \beta^*)$
12: continue
13: **end if**
14: break
15: **end while**
16: $T_m^* \leftarrow T_m$
17: **return** T_m^*

the vertex set U_α of the encoding tree, aggregated over all incoming accounts u_i, all vertices $u_j \in U_\alpha$, and all relations $r \in R$. In contrast, g_α counts only the portion of this flow that crosses the boundary of U_α, i.e., transitions from accounts outside U_α to accounts inside U_α. Thus, $\mathcal{V}_\alpha$ reflects how much of the random walk "lives" inside the group U_α, while g_α quantifies how strongly this group is connected to the rest of the graph through multi-relational edges.

Consequently, we redefine the K-dimensional multi-relation structural entropy of G'_m as follows:

$$\mathcal{H}^K(G'_m) = \min_{T_m}\{\mathcal{H}_m^{T_m}(G'_m)\} = \min_{T_m}\{\sum_{\alpha\neq\lambda}\mathcal{H}_m^{T_m}(G'_m;\alpha)\}, \tag{20}$$

where T_m ranges over all encoding trees of G'_m with maximal height K, and λ denotes the root of the encoding tree. Importantly, we compute a joint entropy for the single multi-relational random walk in G'_m, rather than summing separate entropies over individual relations.

3) Multi-Relational Entropy Optimization: To minimize the high-dimensional multi-relational structural entropy $\mathcal{H}_m^{T_m}(G'_m)$, we adopt the *merge* ($\eta_{mg}$) and *combine* ($\eta_{cb}$) operators from the deDoc algorithm [51], and iteratively optimize the encoding tree T_m for the account graph G'_m.

We first introduce the term $g_{\alpha,\beta}$, which facilitates computing the variation in multi-relational entropy $\mathcal{H}_m^{T_m}(G'_m)$ during optimization:

$$g_{\alpha,\beta} = \sum_{u_i\in U_\alpha}\sum_{u_j\in U_\beta}\sum_{r_k\in R}\left[P_{i,j,r}^{mr}P_{i,r}^{vr}\pi_i^u\right], \tag{21}$$

where $U_\alpha \subseteq U$ and $U_\beta \subseteq U$ denote the account subsets associated with tree nodes α and β, respectively. Intuitively, $g_{\alpha,\beta}$ measures the stationary random-walk flow from accounts in U_α to accounts in U_β, aggregated over all relations $r \in R$.

When the *merge* operation is applied to a pair of sibling nodes α and β, a new tree node δ_{mg} is created to replace both α and β. In this case, the parent of δ_{mg} is set to be the original parent of α and β, and all children of α and β become children of δ_{mg}. The corresponding entropy variation $\Delta_{mg}(T_m, \alpha, \beta)$ is given by:

$$\begin{aligned}\Delta_{mg}(T_m,\alpha,\beta) = &\frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathcal{V}_\lambda}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{mg}}} - \\ &\frac{\sum_{i\neq j}^{L_\alpha} g_{\alpha_i,\alpha_j}}{\mathcal{V}_\lambda}\cdot\log_2\frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\alpha} - \frac{\sum_{i\neq j}^{L_\beta} g_{\beta_i,\beta_j}}{\mathcal{V}_\lambda}\cdot\log_2\frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\beta},\end{aligned} \tag{22}$$

where $\{\alpha_i\}_{i=1}^{L_\alpha}$ and $\{\beta_i\}_{i=1}^{L_\beta}$ are the children of α and β, respectively, and α^- is the parent of α (which is also the parent of β before merging).

In contrast, when the *combine* operation is applied to sibling nodes α and β, a new tree node δ_{cb} is inserted as their parent, while α and β themselves are preserved. More precisely, the original parent of α and β becomes the parent of δ_{cb}, and δ_{cb} becomes the new parent of α and β. The associated entropy variation $\Delta_{cb}(T_m, \alpha, \beta)$ is:

$$\Delta_{cb}(T_m,\alpha,\beta) = \frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathcal{V}_\lambda}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{cb}}}. \tag{23}$$

The *merge* and *combine* operators modify the encoding tree structure in complementary ways. The *merge* operator fuses two sibling communities into a single, larger community when the flow between them (captured by $g_{\alpha,\beta}$ and $g_{\beta,\alpha}$) is sufficiently strong, thereby simplifying the tree. In contrast, the *combine* operator inserts a new parent above two siblings, creating a higher-level community that groups them together without collapsing their internal structure. In both cases, the entropy variations Δ_{mg} and Δ_{cb} quantify how these structural changes affect the description length of the multi-relational random walk.

Building on the *merge* and *combine* operators, we design a greedy, iterative optimization algorithm summarized in Algorithm 2. In each iteration, the algorithm traverses all pairs of sibling nodes α and β (lines 4 and 9 in Algorithm 2) and evaluates the corresponding entropy variations for applying either the *merge* or *combine* operator. It then selects the pair (α^*, β^*) and operator that yield the largest reduction in multi-relational entropy (lines 6 and 11 in Algorithm 2). The iteration terminates when no further entropy reduction is possible (line 14 in Algorithm 2) orthe maximal tree height K is reached (line 3 in Algorithm 2). The final encoding tree is returned as the optimal tree T_m^* (line 16 in Algorithm 2).

The optimal tree T_m^* naturally induces a hierarchical organization of account communities: each node $\alpha \in T_m^*$ corresponds to an account subset $U_\alpha \subset U$, which we interpret as an account community characterized by frequent multi-relational interactions among the accounts in U_α.

C. Network Structure Evolution

In this subsection, we break down the evolution of network structures into two key tasks: determining social activities and selecting target accounts. We then employ multi-agent reinforcement learning to simulate future interactions within each account community, thereby modeling bot behaviors with enhanced accuracy and efficiency. By default, within each

community U_{λ_i}, which is represented by the child node λ_i of the root in the tree T_m^*, we randomly sample k bot accounts to serve as the modeled bot accounts, denoted by $b_1, b_2, \ldots, b_k$.

1) Social Activity Determination: The activity determination problem is formulated as a Markov Decision Process (MDP) characterized by the tuple $\mathcal{M}_h = (\mathcal{S}_h, \mathcal{A}_h, \mathcal{R}_h, \mathcal{P}, \gamma)$, where $\mathcal{S}_h$ is the state space, $\mathcal{A}_h$ is the action space, $\mathcal{R}_h$ is the reward function, $\mathcal{P}$ is the transition function and γ is the discount factor.

Within the account community U_{λ_i}, the high-level policy $\pi_h : \mathcal{S}_h \times \mathcal{A}_h \to [0, 1]$ is responsible for determining each modeled bot b_j should participate in which type of interaction $r \in R$. At each timestep t, the high-state $s_t^h \in \mathcal{S}_h$ represents a snapshot of the historical interactions performed by all target bots. To mitigate the increasing computational and space overhead over time, we encode the sequence of historical activities using the conditional distribution as defined in Equation 10, which represents the probability of selecting a specific interaction type $r \in R$ for each bot b_j. This distribution is then used to construct the high-level state $s_t^h \in \mathbb{R}^{k|R|}$. Given the high-level state s_t^h, the policy π_h selects an action $a_t^h \in \mathcal{A}_h$, denoted as a binary tensor $\{0, 1\}^{k \times |R|}$. This tensor determines the interaction type r_j^t of each target bot b_j at timestep t.

The associated reward $\mathcal{R}_h(s_t^h, a_t^h)$ is designed to reflect the variation in the prediction probabilities of bot detection outcomes for all modeled bots, as assessed by a black-box bot detector f_θ. The reward is computed as follows:

$$\mathcal{R}_h(s_t^h, a_t^h) = \sum_{j=1}^{k} \left[f_\theta(X'_u, E'_h)_{b_j} - f_\theta(X_u, E_h)_{b_j} \right], \quad (24)$$

where X'_u and E'_h are the updated feature representations of heterogeneous network G'_h, reflecting the latest changes induced by network evolution in structure and content.

To model the behavioral objective of evading bot detection, we optimize the high-level policy π_h to maximize the long-term expected discounted reward. This involves selecting appropriate interaction types for modeled bots to reduce the prediction probabilities by the bot detector f_θ. The optimization objective is formalized as follows:

$$J(\pi_h) = \mathbb{E}\left[\sum_{t=0} \gamma^t \mathcal{R}_h(s_t^h, a_t^h) \right]. \quad (25)$$

2) Target Account Selection: The target account selection problem is modeled as a multi-agent MDP described as the tuple $\mathcal{M}_l = (\mathcal{I}, \mathcal{S}_l, \mathcal{A}_l, \mathcal{R}_l, \mathcal{P}, \gamma)$, where $\mathcal{I} = \{b_1, \ldots, b_k\}$ is the set of modeled bots (agents), $\mathcal{S}_l$ is the state space, $\mathcal{A}_l$ is the action space, and $\mathcal{R}_l$ is the reward function. The transition function $\mathcal{P}$ and discount factor γ are the same as the high-level MDP $\mathcal{M}_h$.

Within the account community U_{λ_i}, the low-level policy $\pi_j^l : \mathcal{S}_l \times \mathcal{A}_l \to [0, 1]$ determines which human or bot account is connected to the modeled bot b_j through the type of interaction r selected by the high-level policy π_h. When the policy π_j^l selects target accounts, two primary factors are considered for each user account:

• **Network Influence:** A user account with greater network influence has a higher probability of being selected.

• **Behavioral Relevance:** A user account with greater behavioral relevance to bot b_j has a higher likelihood of being selected.

To quantify these factors, we define two key metrics for each account $u \in U_{\lambda_i}$: $\mathrm{ni}(u)$ and $\mathrm{br}(b_j, u)$. These metrics quantify the network influence of u and its behavioral relevance to b_j, based on multi-relational entropy as defined in Equation 17. They are computed as follows:

$$\mathrm{ni}(u) = \sum_{u \in U_\alpha} \mathcal{H}_m^{T_m^*}(G'_m; \alpha), \quad (26)$$

$$\mathrm{br}(b_j, u) = \sum_{u \in U_\alpha \cap b_j \in U_\alpha} \mathcal{H}_m^{T_m^*}(G'_m; \alpha), \quad (27)$$

where $\mathrm{ni}(u)$ represents the number of bits required to determine the engagement level of account u in a random interaction, reflecting its network influence. On the other hand, $\mathrm{br}(b_j, u)$ represents the number of shared bits between the engagement patterns of b_j and u in a random interaction, indicating the behavioral relevance of account u to bot b_j.

By incorporating these metrics into the structural embedding of the subgraph induced by the community U_{λ_i}, we construct the low-level state $s_t^l \in \mathcal{S}_l$ at timestep t. For each account $u \in U_{\lambda_i}$, the state $s_t^l \in \mathbb{R}^{|U_{\lambda_i}|(d+2)}$ consists of the influence metric $\mathrm{ni}(u)$, the relevance metric $\mathrm{br}(b_j, u)$, each of size 1, and the vertex embedding of size d, obtained using the unsupervised encoding method $\mathrm{node2vec}$ [52]. Based on this low-level state s_t^l, the policy π_j^l of modeled bot b_j selects a target account $u \in U_{\lambda_i}$ to establish a new interaction of type r, represented as a one-hot action a_t^l of size $|U_{\lambda_i}|$.

The low-level reward $\mathcal{R}_l(s_t^l, a_t^l)$ is defined as the sum of the influence metrics of all modeled bots, as described in Equation 26, and is given by:

$$\mathcal{R}_l(s_t^l, a_t^l) = \sum_{j=1}^{k} \mathrm{ni}(b_j), \quad (28)$$

where $\mathrm{ni}(b_j)$ denotes the network influence of modeled b_j.

To model the behavioral objective of maximizing the total influence of all modeled bots, we optimize the multi-agent low-level policies $\boldsymbol{\pi_l}$, and the optimization objective is formalized as follows:

$$J(\boldsymbol{\pi_l}) = \mathbb{E}\left[\sum_{t=0} \gamma^t \mathcal{R}_l(s_t^l, a_t^l) \right]. \quad (29)$$

D. Network Content Evolution

Leveraging the evolved network structure, we identify modeled bots, target accounts, and interaction types to construct relevant and contextually appropriate prompts. These prompts are then used by large language models (LLMs) to generate message content. In the heterogeneous social network, we introduce new message vertices with the generated content and establish interactions between modeled bots and their target accounts, thereby simulating the network's dynamic evolution in both structure and content.

Previous studies [34], [53] have emphasized the importance of prompt engineering in improving language model

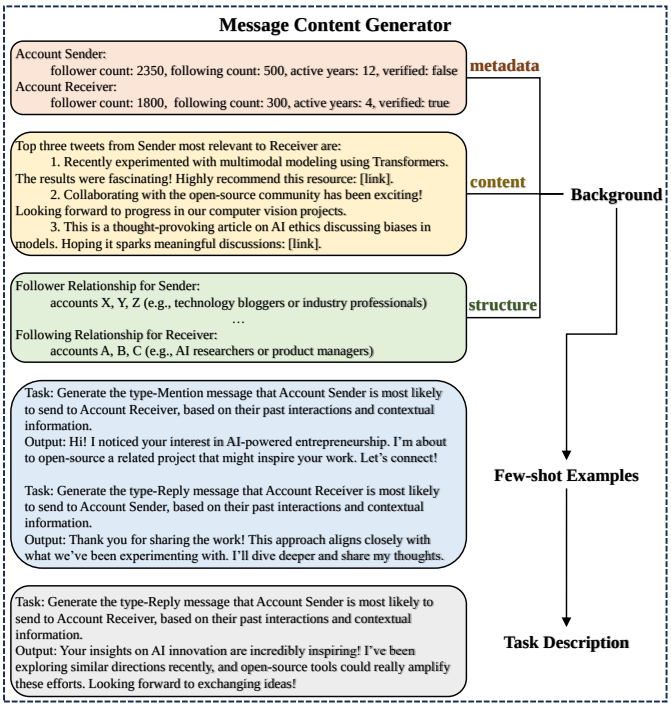

Fig. 4. Message content generation pipeline for each modeled behavior. Given the evolved heterogeneous social network, SIAMD identifies modeled bot accounts, target accounts, and interaction types to construct structured prompts, which are fed into a LLM to synthesize behavior-consistent messages that are reinserted into the network as new message vertices.

outputs through explicit task design and context incorporation. Therefore, we integrate user metadata, historical content, and social structure into a prompt-based approach, allowing LLMs to generate contextually accurate and relevant predictions. As shown in Figure 7, the process involves three stages: background representation, task formulation, and content generation through in-context learning [54].

To provide LLMs with a rich understanding of account profiles and interactions, we represent background knowledge in three dimensions, inspired by prior work on leveraging structured data in natural language inputs [55], [56]:

• **Metadata**: Accounts' categorical information, including follower count, following count, and account descriptions, is reformulated into natural language sequences.

• **Content**: Using user embedding and similarity-based text retrieval techniques, we identify the top three tweets from each bot that exhibit the highest cosine similarity to its target account's embedding vector.

• **Structure**: Follower and following relationships for each account are encoded into natural language lists, as studies have highlighted the utility of graph-based representations in social network tasks [57].

The predictive task is articulated succinctly, minimizing unnecessary repetition and effectively leveraging background knowledge to create clear and focused instructions for the generation model. For instance, the task might be framed as: ''Task: Generate the type-[interaction type] message that Account [Sender] is most likely to send to Account [Receiver], based on their past interactions and contextual information."

To guide the model, we provide few-shot examples comprising historical interactions between modeled bots and their target accounts (input) along with the corresponding message content (output). The background knowledge and examples are combined to construct a comprehensive prompt for content generation. We further employ temperature tuning [58] to adjust the randomness of predictions and optimize the diversity of the generated outputs.

Algorithm 3 Adversarial detection based on structural information principles.

1: **Input:** the heterogeneous social network G_h, the pre-trained graph-based detection model f_θ
2: **Output:** the predicted binary label $\hat{y}$ for each account u
3: **for** each iteration **do**
4: $G_m \leftarrow$ construct the multi-relational account graph using meta-path instances in G_h
5: $G'_m \leftarrow$ adjust the multi-relational graph G_m
6: $T^*_m \leftarrow$ derive the optimal encoding tree by minimizing multi-relational entropy $\mathcal{H}^{T_m}_m(G'_m)$
7: **for** each community U_{λ_i} **do**
8: $b_1, b_2, \ldots, b_k \leftarrow$ sample modeled bot accounts
9: **for** each timestep t **do**
10: $a^h_t \leftarrow$ select the high-level action a^h_t according to policy π_h
11: $a^l_t \leftarrow$ select the low-level action a^l_t according to policy π_l
12: **for** each modeled bot b_j **do**
13: $m^t_j \leftarrow$ generate the message content m^t_j for interaction type r^t_j between bot b_j and user u^t_j
14: $G'_h \leftarrow$ update the structure and content of network G_h
15: **end for**
16: **end for**
17: **end for**
18: $f_\theta \leftarrow$ optimize the graph-based detector f_θ on the updated network G'_h
19: **end for**
20: **return** $\hat{y} \leftarrow$ predict binary label for each account $u \in U$

E. Time Complexity Analysis

We summarize the proactive detection process of our SIAMD framework in Algorithm 3. At each adversarial iteration, we calculate and optimize the multi-relational structural entropy of the account graph G_m to derive the tree-structured account communities T^*_m with a time complexity of $O(|E_h|+|U|\log^2|U|)$ (lines 4–6 in Algorithm 3). Within each community U_{λ_i}, we identify the interaction types and target accounts associated with all modeled bots (lines 10 and 11 in Algorithm 3) and leverage large language models to generate the associated messages and update the network structure (lines 13 and 14 in Algorithm 3), with a time complexity of $O(k|R| + |U_{\lambda_i}|)$. In summary, the total time complexity of SIAMD per iteration is $O(|E_h|+|U|\log^2|U|+(k|R|+1)|U|)$.

V. Experimental Setup

A. Datasets

To evaluate the detection performance of the SIAMD framework, we conduct comparative experiments using four well-

established bot datasets from the Bot Repository—Cresci-15 [45], Cresci-17 [18], [46], TwiBot-20 [47], and TwiBot-22 [24]—all of which are publicly available for research. For each dataset, we perform a stratified random split with a $7:2:1$ ratio into training, validation, and testing sets, ensuring that the proportion of human and bot accounts remains consistent across splits, thereby mitigating class imbalance and ensuring a fair comparison.

B. Compared Baselines

We compare the detection performance of SIAMD with state-of-the-art baselines from three categories: feature-based methods, including BotHunter [59] and SGBot [15], content-based methods, including BGSRD [60] and RoBERTa [50], and graph-based methods, such as GraphHist [61], SATAR [22], Botometer [62], SimpleHGN [63], BotRGCN [23], and RGT [27].

C. Implementation Details

We implement the SIAMD framework using Python 3.10 and PyTorch on Linux servers (Ubuntu 20.04 LTS), with a 64-core Intel Xeon Platinum 8336C CPU running at 2.30 GHz and an NVIDIA A800-SXM4-80GB GPU. For all baselines, we utilize their open-source implementations with the default hyperparameters provided by the original authors. For the proposed SIAMD, we set the hidden dimension to 64, the learning rate to 0.001, and use the Adam optimizer. All models are initialized randomly using a truncated normal distribution with a mean of 0 and a standard deviation of 0.01. In the structure evolution stage, we set the maximum height of the encoding tree to 5, the proportion of modeled bot accounts within each community to 0.5, and the factor γ to 0.01. In the content evolution stage, we select LLaMA2-70B [64] as the default large language model, using at most 3 few-shot in-context examples with the temperature set to 0.1. In the detection optimization stage, we choose GCN [25] as the underlying graph-based detection model and set the number of optimization iterations to 10, ensuring that the total number of training epochs matches those of the baselines. For statistical robustness, we repeat each experiment five times with different random seeds (1, 10, 50, 100, 500) and report the average performance along with the standard deviation. During training, we adopt an early stopping criterion based on the F1 score on the validation set, with the patience parameter set to 10; the checkpoint with the highest validation F1 is used for performance evaluation on the test set.

D. Evaluation Metrics

We evaluate overall detection performance by calculating the number of classified accounts, including True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), and then computing the Accuracy which is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{30}$$

We also use the following Precision and Recall metrics to quantify the proportion of social bots identified as bots and the proportion of correctly identified bot samples:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}. \tag{31}$$

To provide a more comprehensive measure of detection effectiveness, we compute the balanced F1 score, as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}. \tag{32}$$

VI. Evaluation

In this section, we conduct comprehensive experiments to evaluate the performance of the SIAMD framework, addressing the following key questions:

- **Section VI-A**: What is the effectiveness of the SIAMD framework in modeling and detecting social bots?
- **Section VI-B**: How does SIAMD perform across various datasets, particularly in terms of generalizability, robustness, and interpretability?
- **Section VI-C**: How does SIAMD's performance vary depending on the underlying graph-based detection algorithm and content generation model?
- **Section VI-D**: What is the contribution of the structure and content evolution stages to the performance of our proposed SIAMD framework?
- **Section VI-E**: How efficient is the SIAMD framework in terms of behavioral modeling and detection across networks of varying scales?

A. Overall Performance

1) Detection Effectiveness: We summarize the detection results of both feature-based, content-based, and graph-based baseline models, as well as our SIAMD, in Table II. Our model consistently outperforms the baseline methods in both Accuracy and F1 scores across all four datasets, highlighting SIAMD's superior detection capabilities. Specifically, SIAMD outperforms all detection baselines in terms of both Accuracy and F1 score across the Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22 datasets, with improvements of at least 1.4, 2.5, 4.0, 5.2 for accuracy and 1.1, 1.8, 3.6, 5.6 for F1 score, respectively. These improvements become more pronounced as both dataset size and detection difficulty increase, further demonstrating our framework's advantage in challenging scenarios. Although SIAMD does not achieve the best Precision and Recall results on the Cresci-15 and Cresci-17 datasets, it performs close to the best method and maintains superior performance in the more comprehensive F1 score, still illustrating the effectiveness of our framework. Even on the most challenging dataset, TwiBot-22, SIAMD significantly improves both Accuracy and F1 score, increasing from 79.7 and 57.5 (for the best-performing baseline, BotRGCN) to 84.9 and 63.1, respectively, resulting in improvements of 6.5% and 9.7%. We observe that, compared to baseline methods, SIAMD significantly improved Recall by 6.6 (13.9%) on TwiBot-22. This improvement in Recall can be attributed to SIAMD's

TABLE II
DETECTION PERFORMANCE OF SIAMD AND BASELINE METHODS WITHIN FOUR BOT DETECTION DATASETS: "AVERAGE VALUE ± STANDARD DEVIATION". **BOLD** INDICATES THE BEST PERFORMANCE ON EACH DATASET, AND UNDERLINE DENOTES THE SECOND-BEST PERFORMANCE.

Method	Cresci-15				Cresci-17			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
BotHunter	96.5 ± 1.2	$\underline{98.6} \pm 0.6$	91.5 ± 4.2	95.2 ± 1.0	88.1 ± 0.2	$\mathbf{98.7} \pm 0.1$	85.4 ± 0.2	91.6 ± 3.1
SGBot	77.1 ± 0.2	97.5 ± 0.2	64.9 ± 0.1	77.9 ± 0.1	92.1 ± 0.3	98.3 ± 0.2	91.2 ± 0.4	94.6 ± 0.2
BGSRD	87.8 ± 0.6	86.5 ± 0.6	95.6 ± 2.0	90.8 ± 0.6	75.7 ± 0.2	75.6 ± 0.1	96.8 ± 1.7	86.3 ± 0.1
RoBERTa	97.0 ± 0.1	97.6 ± 0.3	94.1 ± 0.6	95.9 ± 0.2	94.1 ± 0.8	92.4 ± 1.0	96.3 ± 1.1	94.3 ± 0.2
GraphHist	77.4 ± 0.2	73.1 ± 0.1	99.0 ± 0.7	84.5 ± 8.2	73.8 ± 0.4	69.7 ± 1.0	86.6 ± 0.8	77.2 ± 0.5
SATAR	93.4 ± 0.5	90.7 ± 0.1	$\mathbf{99.9} \pm 0.1$	95.1 ± 0.3	89.2 ± 0.3	87.1 ± 0.4	93.7 ± 0.2	90.3 ± 0.7
Botometer	57.9 ± 0.3	50.5 ± 0.2	99.0 ± 0.7	66.7 ± 0.4	$\underline{94.2} \pm 0.4$	93.4 ± 0.3	97.3 ± 1.5	96.1 ± 3.1
SimpleHGN	96.7 ± 0.5	95.7 ± 0.9	99.3 ± 0.4	97.3 ± 0.4	91.5 ± 0.2	90.2 ± 0.4	94.3 ± 0.4	92.5 ± 0.8
BotRGCN	96.5 ± 0.7	95.5 ± 1.0	99.2 ± 0.3	97.3 ± 0.5	92.7 ± 0.3	91.8 ± 0.5	96.8 ± 0.4	94.4 ± 0.3
RGT	$\underline{97.2} \pm 0.3$	96.4 ± 0.6	99.2 ± 0.2	$\underline{97.8} \pm 0.2$	94.0 ± 0.6	93.8 ± 1.2	$\mathbf{98.3} \pm 0.6$	$\underline{96.3} \pm 1.1$
SIAMD (Ours)	$\mathbf{98.6} \pm 0.3$	$\mathbf{98.9} \pm 0.4$	$\underline{99.4} \pm 0.4$	$\mathbf{98.9} \pm 0.5$	$\mathbf{96.7} \pm 0.6$	$\underline{98.5} \pm 0.2$	$\underline{98.1} \pm 0.3$	$\mathbf{98.1} \pm 0.4$
Abs.(%) Avg.↑	1.4(1.4)	0.3(0.3)	0.5(0.5) ↓	1.1(1.1)	2.5(2.7)	0.2(0.2) ↓	0.2(0.2) ↓	1.8(1.9)
Method	**TwiBot-20**				**TwiBot-22**			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
BotHunter	75.2 ± 0.4	72.8 ± 0.3	86.6 ± 0.3	79.1 ± 0.4	72.8 ± 0.1	68.1 ± 0.4	14.2 ± 0.8	23.5 ± 0.1
SGBot	81.6 ± 0.5	76.4 ± 0.4	$\underline{95.6} \pm 0.1$	84.9 ± 0.4	75.1 ± 0.1	$\underline{73.1} \pm 0.1$	24.3 ± 0.4	36.6 ± 0.2
BGSRD	66.4 ± 1.0	67.6 ± 2.3	73.2 ± 7.5	70.1 ± 2.6	71.9 ± 1.8	22.6 ± 3.1	19.9 ± 2.7	21.1 ± 3.0
RoBERTa	75.5 ± 0.1	73.9 ± 1.1	72.4 ± 2.1	73.1 ± 0.2	72.1 ± 0.1	63.3 ± 0.9	12.3 ± 1.2	20.5 ± 1.7
GraphHist	51.3 ± 0.3	51.3 ± 0.2	94.7 ± 0.7	65.6 ± 0.3	37.2 ± 1.3	41.3 ± 2.1	26.1 ± 0.8	31.9 ± 2.7
SATAR	84.0 ± 0.8	81.5 ± 1.5	91.3 ± 0.5	86.1 ± 0.7	43.6 ± 1.0	37.5 ± 1.3	34.4 ± 1.5	35.8 ± 2.1
Botometer	53.1 ± 0.3	55.7 ± 0.4	50.8 ± 0.3	53.0 ± 0.2	48.3 ± 1.2	32.4 ± 1.3	$\underline{47.5} \pm 2.9$	38.9 ± 1.4
SimpleHGN	$\underline{86.7} \pm 0.2$	84.8 ± 0.5	92.1 ± 0.5	$\underline{88.3} \pm 0.2$	74.7 ± 0.3	72.6 ± 0.8	32.9 ± 1.6	45.4 ± 1.7
BotRGCN	85.8 ± 0.7	84.5 ± 0.5	90.2 ± 1.7	87.3 ± 0.7	$\underline{79.7} \pm 0.1$	74.8 ± 2.2	46.8 ± 2.8	$\underline{57.5} \pm 1.4$
RGT	86.6 ± 0.4	$\underline{85.2} \pm 0.3$	91.1 ± 0.8	88.0 ± 0.4	76.5 ± 0.4	75.0 ± 0.9	30.1 ± 0.2	42.9 ± 1.9
SIAMD (Ours)	$\mathbf{90.7} \pm 0.3$	$\mathbf{87.4} \pm 0.2$	$\mathbf{95.8} \pm 1.0$	$\mathbf{91.9} \pm 0.2$	$\mathbf{84.9} \pm 0.4$	$\mathbf{79.4} \pm 1.3$	$\mathbf{54.1} \pm 1.5$	$\mathbf{63.1} \pm 1.6$
Abs.(%) Avg.↑	4.0(4.6)	2.2(2.6)	0.2(0.2)	3.6(4.1)	5.2(6.5)	6.3(8.6)	6.6(13.9)	5.6(9.7)

TABLE III
MODELING PERFORMANCE OF SIAMD AND BASELINE METHODS IN EPISODE REWARD AND LENGTH: "AVERAGE VALUE ± STANDARD DEVIATION". **BOLD** INDICATES THE BEST PERFORMANCE ON EACH DATASET, AND UNDERLINE DENOTES THE SECOND-BEST PERFORMANCE.

Method	Cresci-15		Cresci-17		TwiBot-20		TwiBot-22	
	Reward	Length	Reward	Length	Reward	Length	Reward	Length
DEGREE	720.3 ± 24.1	48.3 ± 17.4	702.8 ± 27.4	46.1 ± 20.6	584.0 ± 62.7	41.6 ± 19.3	412.9 ± 43.2	33.5 ± 17.2
ACRON	754.8 ± 25.3	61.4 ± 20.2	739.2 ± 41.3	65.2 ± 27.9	591.1 ± 48.2	52.3 ± 26.4	447.3 ± 31.6	42.9 ± 19.5
SIASM	$\underline{772.6} \pm 31.4$	$\underline{70.6} \pm 14.7$	$\underline{762.8} \pm 29.1$	$\underline{68.5} \pm 25.3$	$\underline{648.5} \pm 45.2$	$\underline{64.9} \pm 25.8$	$\underline{602.7} \pm 34.9$	$\underline{60.5} \pm 23.8$
SIAMD (Ours)	$\mathbf{807.4} \pm 26.9$	$\mathbf{79.1} \pm 18.3$	$\mathbf{800.3} \pm 28.5$	$\mathbf{72.6} \pm 21.8$	$\mathbf{697.2} \pm 39.4$	$\mathbf{70.1} \pm 21.6$	$\mathbf{670.5} \pm 28.7$	$\mathbf{66.4} \pm 20.4$
Abs.(%) Avg.↑	34.8(4.5)	8.5(12.0)	37.5(4.9)	4.1(6.0)	48.7(7.5)	5.2(8.0)	67.8(11.2)	5.9(9.8)

network evolution in both structure and content, which enables bot accounts to exhibit more distinctive behavioral patterns, thereby facilitating more accurate and effective identification of true positive samples.

2) Modeling Effectiveness: To further evaluate the performance of SIAMD in behavioral modeling, we compare it with several established methods: the classical heuristic method DEGREE [65], the reinforcement learning-based method ACRON [35], and our previous version, SIASM [49]. During the structure evolution stage, we compute the average reward and average length of each testing episode, which serve as metrics for network influence and sustainable stealthiness, respectively. The mean values and standard deviations of episodic reward and length are presented in Table III. Across all datasets, SIAMD significantly outperforms SIASM, the next best method, by at least 4.5% in average reward and 6.0% in average length. This highlights the effectiveness of analyzing bot behavioral patterns through structural information principles and further demonstrates the advantage achieved by incorporating multi-relation structural entropy and multi-agent coordination during the structure evolution stage.

B. Ability Evaluation

1) Generalization Ability: To illustrate the generalization ability of SIAMD on unseen data, we identify 10 sub-communities within the TwiBot-22 network, each containing $5,000$ human accounts and $5,000$ bot accounts. Following prior studies [24], we select 5 sub-communities centered around popular topics, including *@BarackObama*, *@elonmusk*, *@CNN*, *@NeurIPSConf*, and *@ladygaga*. Additionally, we categorize social media posts related to specific hashtags, such as *#Manchester*, *#cybersecurity*, *#Colombia*, *#Industry*, and *#USMNT*, into 5 additional sub-communities.

We treat these sub-communities as folds to examine the detection performance of SIAMD and the three top-performing baselines (SGBot, BotRGCN, and RGT), trained on fold i and evaluated on fold j. The heatmaps displaying the accuracy of each model, along with the corresponding average values and standard deviations, are presented in Figure 5. Notably, compared to the three baselines, SIAMD consistently performs best in detection performance across all fold pairs, achieving the highest average accuracy of 84.05 and the lowest standard deviation of 5.91. On the testing fold 1, where the three baseline algorithms show significantly lower

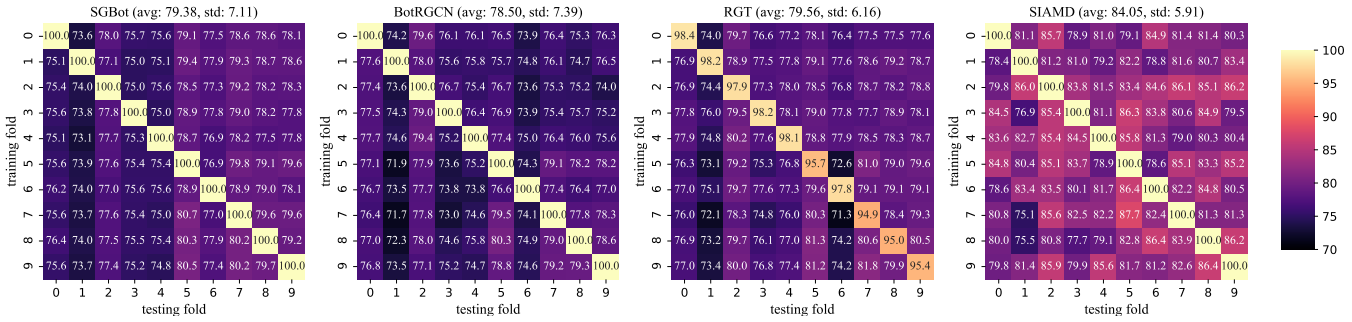

Fig. 5. Cross-fold detection accuracy (mean ± standard deviation) of SIAMD and three top-performing baselines (SGBot, BotRGCN, and RGT) on the TwiBot-22 dataset. Each heatmap cell corresponds to training on sub-community fold i and testing on fold j, with values reported as avg ± std across repeated runs.

TABLE IV
VARIATION IN DETECTION ACCURACY AND F1 SCORE ON THE PERTURBED TWIBOT-20 DATASET (MEAN ± STANDARD DEVIATION) UNDER LLAMA2-70B- AND CHATGPT-DRIVEN ADVERSARIAL ATTACKS. **BOLD** INDICATES THE BEST PERFORMANCE IN EACH SETTING, AND UNDERLINE DENOTES THE SECOND-BEST PERFORMANCE.

Method	Before Attack		LLaMA2-70B Attack		ChatGPT Attack		Average Variation	
	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
BotHunter	75.2 ± 0.4	79.1 ± 0.4	59.7 ± 0.6	62.4 ± 0.3	61.3 ± 0.5	64.9 ± 0.3	14.7 ± 0.6	15.5 ± 0.3
SGBot	81.6 ± 0.5	84.9 ± 0.4	62.5 ± 0.4	65.3 ± 0.6	65.1 ± 0.5	68.4 ± 0.3	17.8 ± 0.5	18.1 ± 0.5
BGSRD	66.4 ± 1.0	70.1 ± 2.6	47.6 ± 0.3	50.3 ± 0.7	51.8 ± 0.6	53.9 ± 0.5	16.7 ± 0.8	18.0 ± 1.1
RoBERTa	75.5 ± 0.1	73.1 ± 0.2	58.1 ± 0.7	62.8 ± 0.3	58.6 ± 0.5	63.7 ± 0.4	17.2 ± 0.4	$\underline{9.9} \pm 0.4$
GraphHist	51.3 ± 0.3	65.6 ± 0.3	37.4 ± 0.8	45.3 ± 0.5	38.2 ± 0.7	49.3 ± 0.7	13.5 ± 0.6	18.3 ± 0.5
SATAR	84.0 ± 0.8	86.1 ± 0.7	67.3 ± 0.6	72.5 ± 0.4	69.1 ± 0.7	75.8 ± 0.4	15.8 ± 0.8	12.0 ± 0.6
Botometer	53.1 ± 0.3	53.0 ± 0.2	37.2 ± 0.3	35.7 ± 0.4	38.4 ± 0.3	37.9 ± 0.4	15.3 ± 0.3	16.2 ± 0.4
SimpleHGN	$\underline{86.7} \pm 0.2$	$\underline{88.3} \pm 0.2$	74.1 ± 0.2	$\underline{77.5} \pm 0.3$	72.8 ± 0.4	75.4 ± 0.3	13.3 ± 0.4	11.9 ± 0.3
BotRGCN	85.8 ± 0.7	87.3 ± 0.7	70.2 ± 0.5	69.1 ± 0.9	$\underline{76.4} \pm 0.3$	73.2 ± 0.5	12.5 ± 0.6	16.2 ± 0.7
RGT	86.6 ± 0.4	88.0 ± 0.4	$\underline{72.7} \pm 0.5$	74.0 ± 0.6	76.2 ± 0.8	$\underline{77.9} \pm 0.6$	$\underline{12.2} \pm 0.7$	12.1 ± 0.6
SIAMD (Ours)	$\mathbf{90.7} \pm 0.3$	$\mathbf{91.9} \pm 0.2$	$\mathbf{79.8} \pm 0.4$	$\mathbf{80.4} \pm 0.5$	$\mathbf{83.7} \pm 0.2$	$\mathbf{85.3} \pm 0.5$	$\mathbf{9.0} \pm 0.6$	$\mathbf{9.1} \pm 0.4$

performance, SIAMD achieves at least 75.1 accuracy. This illustrates that the network evolution process we introduced effectively compensates for the variations between training and testing folds, thereby ensuring that SIAMD demonstrates strong generalization performance.

2) Robustness Ability: To validate the robustness of SIAMD, we introduce an attack algorithm [48] driven by large language models (LLMs), LLaMA2-70b [64] and ChatGPT, to manipulate both the content and structural information of bot accounts in TwiBot-20. For the content information, we retrieve the top-3 most similar messages from human accounts and prompt the LLM to generate a rewritten message that imitates these examples. For the structure information, we provide the LLM with metadata and historical interactions of the target bot to modify its network structure by adding or removing its social neighbors.

We evaluate the detection performance of the SIAMD framework and all baseline methods in Accuracy and F1 score on the LLM-manipulated bot accounts in TwiBot-20, with the results presented in Table IV. Consistent with observations in previous work [48], adversarial manipulation using the LLaMA2-70B model leads to greater performance degradation for both feature-based, content-based, and graph-based detectors. Compared to other baseline models, SIAMD consistently achieves the highest detection accuracy and F1 score across both attack scenarios, while exhibiting minimal performance degradation. The adversarial detection mechanism in the SIAMD framework proactively simulates potential interference scenarios in network structure and content, thereby enhancing its detection robustness.

3) Interpretability Ability: To evaluate the interpretability of SIAMD, we extract a sub-community from the TwiBot-22 network, model the behaviors of three bot accounts (represented by distinct colors), and visualize their structural relationships within the sub-community across various timesteps, as shown in Figure 6. For clarity, we focus on visualizing only the directed interactions between accounts within the sub-community, omitting specific interaction types. The behaviors of the three bot accounts are highlighted by employing distinct colors and thicker edges in their respective subfigures.

The red bot primarily engages in following or retweeting other accounts within the sub-community, with minimal expression of its own opinions. Consequently, it is classified as a human account by the detection algorithm. It frequently replies to and mentions other accounts, particularly those with lower influence in the network. This strategy facilitates rapid spread of opinions and helps build influence, ultimately aiming to foster more concentrated interactions within the sub-community. To avoid detection as a malicious entity, the red bot selectively retweets, mimicking the behaviors typical of human accounts.

In contrast, the yellow bot is more likely to express its own opinions within the sub-community in an effort to influence public opinion, which leads to its classification as a bot account. The primary objective of its early activities is to minimize the likelihood of detection. It forms retweet connections with previously mentioned accounts to simulate bidirectional interactions typical of genuine users. As the likelihood of detection decreases, the yellow bot resumes interactions with

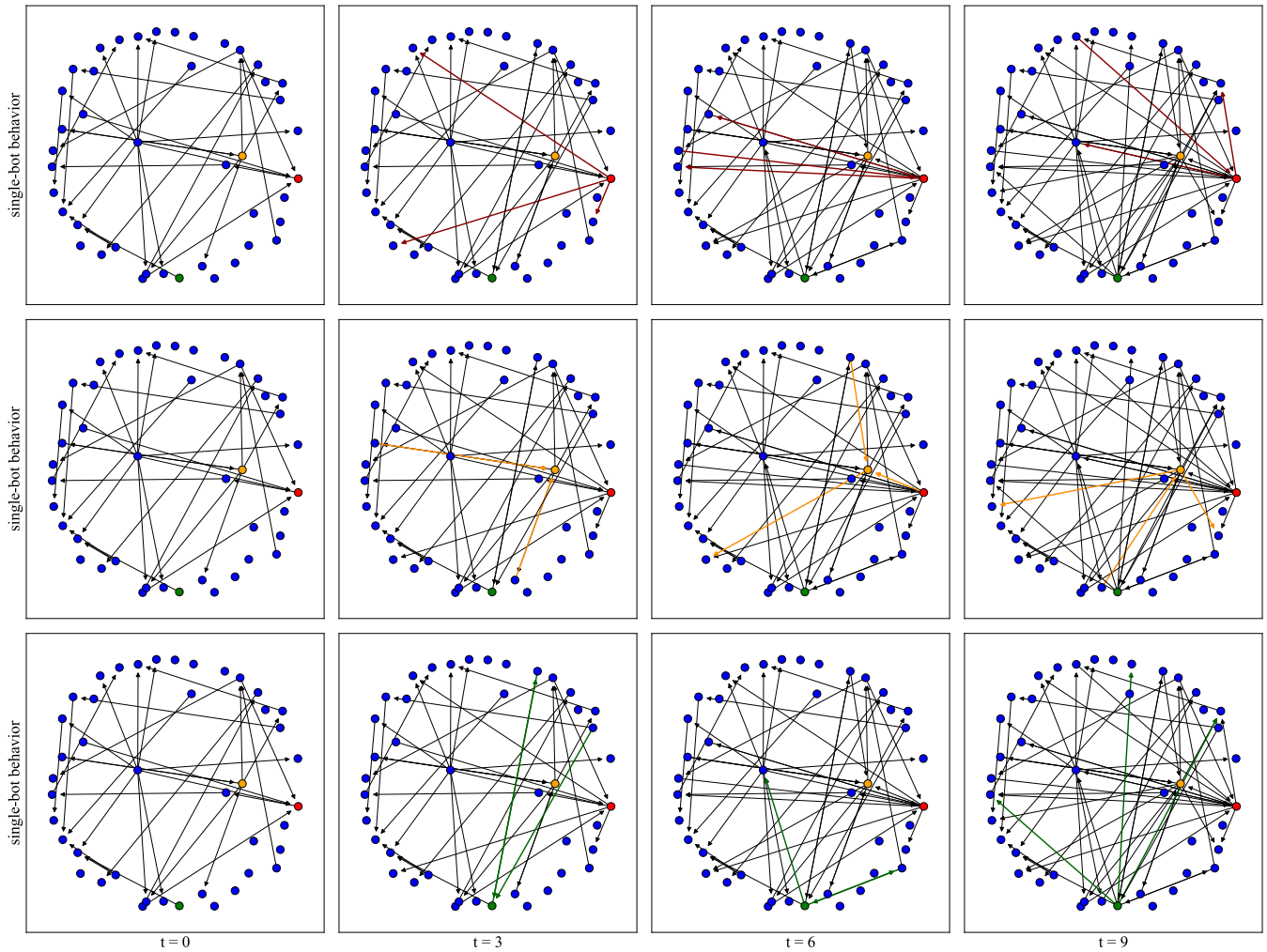

Fig. 6. Visualization of network structure evolution induced by SIAMD on the TwiBot-22 dataset. Horizontal subfigures correspond to different timesteps, while vertical subfigures correspond to different sub-communities. Three modeled bot accounts are highlighted with distinct colors and thicker directed edges to reveal how their interaction patterns evolve over time.

others by mentioning and replying, continuing to express opinions to maintain its influence within the network.

The green bot, in contrast to the other two, engages less frequently in social interactions and exhibits fewer behavioral patterns, resulting in its classification as a human account. This account primarily mimics human behavior by engaging in bidirectional interactions to spread public opinion. Additionally, it is observed that all three bot accounts overlap in their selection of target accounts for interaction. This overlap arises from the collaborative relationship modeled between these accounts, where selecting the same interaction targets amplifies the effectiveness of their public opinion manipulation.

C. Sensitive Analysis

To ensure the performance advantage of our method is independent of specific downstream detection and generation algorithms, we systematically integrate the SIAMD framework with several graph-based detection algorithms and large language models, and compare their respective performance advantages over underlying methods.

1) Detection Algorithms: In the TwiBot-22 dataset, we integrate the GCN [25], GAT [66], and HGT [67] graph-based detection algorithms into the SIAMD framework and evaluate their performance across multiple optimization iterations. The results, summarized in Table V, include key metrics such as Accuracy and F1 score. Regardless of the underlying model, the SIAMD framework consistently enhances detection performance, improving both accuracy and robustness across all iterations. For example, HGT, which initially performed poorly, achieves an Accuracy of 85.1 and an F1 score of 45.1 after SIAMD adversarial optimization. As the number of detection iterations increases, the performance improvement of the SIAMD framework becomes more pronounced relative to the original model. This enhancement is not constrained by the performance bottlenecks of the original model.

2) Generative Models: Within the SIAMD framework, we employ two large language models, Mistral-7B [68] and LLaMA2-70B [64], to generate message content during content evolution. We compare their detection performance across key evaluation metrics, including Accuracy, Precision, Recall, and F1 Score, against the top-performing baseline model. As shown in Figure 7, regardless of the model used, the SIAMD framework consistently demonstrates superior detection performance across all datasets, particularly in the Accuracy

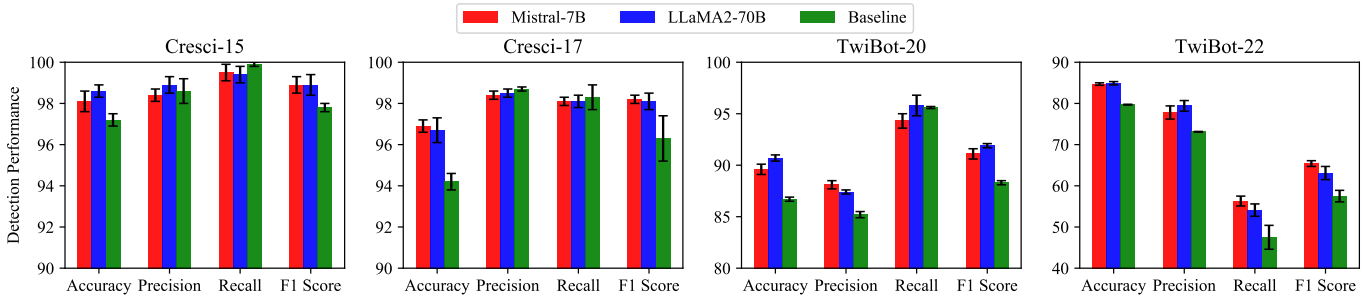

Fig. 7. Performance comparison of different generative large language models (Mistral-7B and LLaMA2-70B) instantiated within the SIAMD framework. Detection results are reported in terms of Accuracy, Precision, Recall, and F1 score and compared against the top-performing baseline detector.

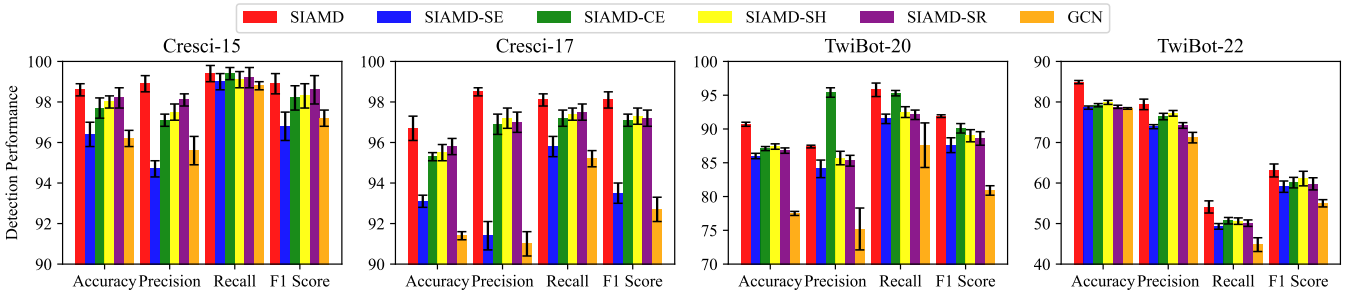

Fig. 8. Ablation study of network structure evolution, content evolution, reinforcement learning (RL) mechanism, and multi-relational graph construction in the SIAMD framework, evaluated on Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22.

TABLE V
AVERAGE DETECTION PERFORMANCE OF SIAMD INSTANTIATED WITH DIFFERENT GRAPH-BASED BACKBONES (GCN, GAT, AND HGT) ON THE TWIBOT-22 DATASET, MEASURED ACROSS OPTIMIZATION ITERATIONS (MEAN ACCURACY AND F1 SCORE).

Iteration	Accuracy					
	GCN	SIAMD	GAT	SIAMD	HGT	SIAMD
1	71.4	72.2	72.5	73.6	69.3	69.9
2	72.9	74.1	74.4	76.2	70.8	72.3
3	74.1	76.1	76.0	78.6	72.0	74.8
4	75.5	77.4	77.1	80.6	72.8	77.4
5	76.5	78.3	77.8	81.2	73.5	80.7
6	77.1	80.6	78.4	82.5	74.0	82.1
7	77.5	82.7	78.9	83.7	74.4	83.2
8	78.1	83.5	79.2	84.9	74.7	84.1
9	78.3	84.2	79.3	85.4	74.9	84.7
10	78.4	84.9	79.5	85.7	74.9	85.1
Iteration	**F1 Score**					
	GCN	SIAMD	GAT	SIAMD	HGT	SIAMD
1	45.7	46.5	40.7	40.8	35.2	35.3
2	48.3	49.7	43.3	43.8	37.4	37.8
3	49.2	52.1	45.0	46.7	38.9	39.6
4	50.8	54.1	47.7	50.2	39.2	40.1
5	52.1	55.3	49.6	53.3	39.4	40.8
6	53.3	57.0	51.9	56.2	39.5	41.7
7	54.1	58.3	53.7	59.4	39.6	43.3
8	54.7	60.2	54.6	61.9	39.5	44.1
9	54.9	61.9	55.3	64.7	39.6	44.7
10	55.0	63.1	55.9	66.4	39.6	45.1

and F1 score. Although the performance of the two SIAMD variants varies across datasets, their relative advantages remain comparable to those of the baseline algorithm. This indicates that the effectiveness of our framework is not dependent on any particular large language model.

D. Ablation Studies

To investigate the contribution of key components in our proposed framework, including the structure-evolution and content-evolution modules, the RL mechanism, and the multi-relational graph construction, we introduce four variants of SIAMD: SIAMD-SE, SIAMD-CE, SIAMD-SH, and SIAMD-SR. In the SIAMD-SE variant, we use the LLaMA model to directly modify the user's account description and historical tweets, without modeling the evolution of the network structure. In the SIAMD-CE variant, we predict structural changes in the social network and select the most contextually relevant historical tweets to drive interactions between each pair of users. In the SIAMD-SH variant, we train an LSTM-based network to select high-level social activities and employ a single agent, instead of multiple agents, for key account selection. In the SIAMD-SR variant, we eliminate the construction of a multi-relational graph and instead treat the social network as a single-relation graph; we then use the original homogeneous structural entropy to partition the network into hierarchical account communities. As shown in Figure 8, across all datasets, the complete SIAMD model consistently outperforms all its variants, which in turn outperform the underlying baseline detection algorithm, GCN. Furthermore, compared with the other variants, the SIAMD-CE variant demonstrates significantly better performance, emphasizing the critical role of behavioral modeling grounded in structural information principles within the framework.

E. Computational Efficiency

To analyze the efficiency of SIAMD in behavior modeling and bot detection, we select high-performance baseline models for these two tasks and compare their per-iteration training time with that of SIAMD. It is important to note that because the hierarchical community partitioning based on multi-relational structural entropy in the SIAMD method introduces significant additional time overhead, we preassign hierarchical communities of social users during the dataset preprocessing stage (i.e., during the user-message feature

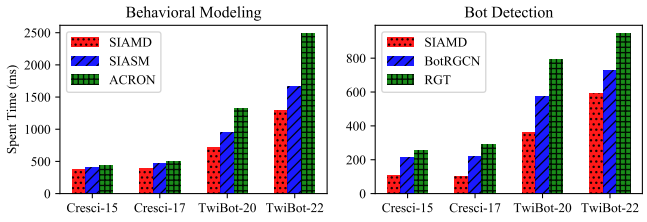

Fig. 9. Efficiency analysis of behavioral modeling and bot detection in the SIAMD framework, evaluated on Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22. We compare the runtime of SIAMD with two high-performing baseline detectors for both the modeling and detection stages.

representation stage) and save them as offline files for online retrieval during subsequent behavior modeling training, thus avoiding redundant community partitioning operations. The detailed results are shown in Figure 9.

In the modeling task, community partitioning based on structural entropy, combined with parallel computing techniques, improves the training efficiency of both SIASM and SIAMD. This efficiency advantage becomes more significant as the network size increases. We further analyze the time overhead of the activity determination and account selection stages on the challenging TwiBot-22 dataset. In the upper-layer activity determination phase, the SIAMD method has an average training cost of 435.3 ms per iteration, outperforming the comparison methods ACRON (720.5 ms) and SIASM (514.7 ms). In the lower-layer account selection phase, SIAMD narrows down the candidate account set through community partitioning and therefore exhibits a more significant efficiency advantage, with an average training cost of only 849.7 ms, much lower than ACRON (1769.9 ms) and SIASM (1144.5 ms). Furthermore, we quantify the single-inference cost of the SIAMD method in the activity determination and account selection phases, showing average costs of 0.4 ms and 2.1 ms, respectively, which validates the scalability and feasibility of the SIAMD method on large-scale datasets.

In the detection task, the SIAMD framework does not introduce additional computational overhead after the network evolution phase. Its efficiency primarily depends on the underlying graph convolutional network (GCN)–based detection algorithm. Therefore, compared to the two baseline models, SIAMD achieves higher computational efficiency with a simpler network structure and fewer parameters. Specifically, on the TwiBot-22 dataset, SIAMD achieves an average training cost of 594.1 ms, significantly outperforming the comparative methods BotRGCN (727.3 ms) and RGT (946.8 ms).

VII. Related Work

In this section, we review related work on social bot detection, including feature-based, content-based, and graph-based methods, as well as social bot modeling and structural information principles.

A. Social Bot Detection

Feature-based detection methods rely on extracting and engineering various user features, such as metadata [69], historical tweets [70], account descriptions [71], and follow relationships [22], to identify bot accounts on social platforms. Recent advances have focused on improving scalability for large-scale detection by using efficient feature selection algorithms [15] and integrating machine learning techniques to optimize the trade-off between precision and recall [72]. However, feature-based methods face an ongoing challenge: as bot operators continually adapt their tactics by modifying key features, these approaches struggle to keep pace with evolving bot strategies [19]. This adaptation makes it increasingly difficult to maintain high detection accuracy, as bot operators can effectively evade detection by tampering with engineered features [22].

Content-based detection methods employ natural language processing (NLP) techniques to analyze message content and account descriptions for bot identification. Approaches such as word embeddings [20] capture semantic relationships between terms in messages, while recurrent neural networks [69] and attention mechanisms [22] improve the detection of sequential patterns and contextual nuances in bot-generated content. Additionally, pre-trained language models [21] have enhanced the ability to understand complex linguistic structures. Despite these advancements, bots are increasingly adopting strategies to evade detection by blending harmful activity with legitimate content sourced from human users [19]. Furthermore, focusing solely on message content without incorporating additional behavioral and structural features has proven insufficient, underscoring the need for more holistic approaches that integrate multiple feature types to improve accuracy [23].

Graph-based detection methods treat social networks as graphs and leverage concepts from network science and geometric deep learning to identify bot accounts. Techniques such as node centrality [73], representation learning [74], and heterogeneous graph neural networks [23] have been successfully used to detect bots based on their network interactions and structural properties. Graph-based methods have shown significant potential in addressing challenges such as identifying bot communities and uncovering disguised bot behaviors [23], [27], [28]. At the same time, work on adversarial attacks against graph neural networks has demonstrated that even small, carefully designed perturbations to node features or edges can substantially degrade GNN performance [29], [37], and that such attacks can be formulated as influence maximization problems on the underlying graph [38]. Moreover, it has been shown that many popular GNN architectures struggle on low-homophily or heterogeneous graphs [32], which are characteristic of mixed human–bot ecosystems where bots strategically connect to legitimate users. Consequently, many existing graph-based bot detectors are inherently reactive: they are trained on historical graph patterns and are not designed to anticipate adaptive structural or behavioral changes by bots [12], [19], [33], [34]. This limitation motivates proactive approaches that explicitly model adversarial evolution rather than assuming a static bot population.

B. Social Bot Modeling

Unlike traditional detection methods that rely on static features and reactive mechanisms [15], [75], bot behavioral

modeling seeks to simulate the dynamic interactions and evolution of bots over time. Early work used evolutionary optimization and genetic algorithms to synthesize future generations of bots from existing ones, with the dual goal of stress-testing detectors and improving robustness [40]. These approaches highlight the arms race between bots and detectors and provide an explicit adversarial augmentation mechanism, but they largely operate at the account-feature level and do not fully capture the joint evolution of network structure and content.

Reinforcement learning (RL) has been introduced to model adversarial social bots as sequential decision makers. For example, [35] formulates the behavior of coordinated social bots as a multi-agent hierarchical RL problem, where bots learn policies that simultaneously maximize influence and avoid detection in large-scale networks. Related work on influence maximization and rumor propagation also leverages deep RL to optimize long-term impact of actions on social graphs [36], [76]. These RL-based models provide powerful tools for simulating adaptive adversaries and studying worst-case scenarios for existing detectors, but they are typically designed from the attacker's perspective and can become computationally expensive when modeling large populations of bots.

C. Structural Information Principles

Since their introduction [44], structural information principles have promoted the analysis of network complexities by introducing innovative metrics like structural entropy and partitioning trees. These methods have enhanced the understanding of network dynamics, including those in multi-relational graphs [77], and have enabled diverse applications across various domains, including graph learning [78], skin segmentation [79], and social network analysis [49], [80], [81]. In reinforcement learning, these principles have been crucial in defining hierarchical abstractions of actions and states using encoding trees [82], [83], resulting in significant advancements in robust decision-making.

Building on these principles, our work extracts the structural information embedded in multi-relational social interactions to identify hierarchical community structures of user accounts. By incorporating a generative model on top of this structural hierarchy, we jointly detect and model bot accounts in large-scale networks. Compared to prior adversarial augmentation and RL-based approaches, this perspective provides finer-grained insight into how coordinated bot communities are organized and evolve, offering a principled and scalable solution to the challenges of proactive bot detection and behavioral analysis.

VIII. Conclusion

This work proposes SIAMD, an adversarial framework that leverages structural information principles to model dynamic bot behaviors, enabling proactive bot detection. Multi-relational structural entropy is formally defined and optimized to reveal hierarchical communities within user accounts, enhancing the effectiveness of activity determination and account selection for behavioral modeling. By integrating large language models, we generate message content associated with bot behaviors, driving the network evolution in both structure and content. Comprehensive evaluations on well-established datasets demonstrate that SIAMD significantly and consistently outperforms feature-based, content-based, and graph-based detection baselines in terms of effectiveness, generalizability, robustness, and interpretability. Future work will focus on investigating the dynamic behaviors of human accounts and introducing additional detection baselines to evaluate our framework on more real-world datasets.

Acknowledgments

This work is supported by the NSFC through grants U25B2029, 62322202, 62441612, 62432006 and 62476163, Beijing Natural Science Foundation through grant L253021, the Pioneer and Leading Goose R&D Program of Zhejiang through grant 2025C02044, Local Science and Technology Development Fund of Hebei Province Guided by the Central Government of China through grant 254Z9902G, National Key Laboratory under grant 241-HF-D07-01, Hebei Natural Science Foundation through grant F2024210008, Major Science Technology Special Projects of Yunnan Province through grants 202502AD080012 and 202502AD080006, and the Fundamental Research Funds for the Central Universities.

References

[1] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[2] V. S. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer, "The darpa twitter bot challenge," *Computer*, vol. 49, no. 6, pp. 38–46, 2016.

[3] M. Mendoza, E. Providel, M. Santos, and S. Valenzuela, "Detection and impact estimation of social bots in the chilean twitter network," *Scientific Reports*, vol. 14, no. 1, p. 6525, 2024.

[4] Z. Huang, Z. Lv, X. Han, B. Li, M. Lu, and D. Li, "Social bot-aware graph neural network for early rumor detection," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 6680–6690.

[5] R. Shao, T. Wu, J. Wu, L. Nie, and Z. Liu, "Detecting and grounding multi-modal media manipulation and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[6] T. Qiao, S. Xie, Y. Chen, F. Retraint, and X. Luo, "Fully unsupervised deepfake video detection via enhanced contrastive learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[7] S. Aral and D. Eckles, "Protecting elections from social media manipulation," *Science*, vol. 365, no. 6456, pp. 858–861, 2019.

[8] S. Rossi, M. Rossi, B. R. Upreti, and Y. Liu, "Detecting political bots on twitter during the 2019 finnish parliamentary election," in *Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2020, pp. 2430–2439.

[9] U. A. Ciftci, I. Demir, and L. Yin, "Fakecatcher: Detection of synthetic portrait videos using biological signals," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[10] E. Ferrara, W.-Q. Wang, O. Varol, A. Flammini, and A. Galstyan, "Predicting online extremism, content adopters, and interaction reciprocity," in *Social Informatics: 8th International Conference, SocInfo 2016, Bellevue, WA, USA, November 11-14, 2016, Proceedings, Part II 8*. Springer, 2016, pp. 22–39.

[11] W. Marcellino, M. Magnuson, A. Stickells, B. Boudreaux, T. C. Helmus, E. Geist, and Z. Winkelman, "Counter-radicalization bot researchusing social bots to fight violent extremism," 2020.

[12] K.-C. Yang, O. Varol, C. A. Davis, E. Ferrara, A. Flammini, and F. Menczer, "Arming the public with artificial intelligence to counter social bots," *Human Behavior and Emerging Technologies*, vol. 1, no. 1, pp. 48–61, 2019.

[13] B. Wu, L. Liu, Y. Yang, K. Zheng, and X. Wang, "Using improved conditional generative adversarial networks to detect social bots on twitter," *IEEE Access*, vol. 8, pp. 36 664–36 680, 2020.

[14] Y. Yang, R. Yang, H. Peng, Y. Li, T. Li, Y. Liao, and P. Zhou, "Fedack: Federated adversarial contrastive knowledge distillation for cross-lingual and cross-model social bot detection," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 1314–1323.

[15] K.-C. Yang, O. Varol, P.-M. Hui, and F. Menczer, "Scalable and generalizable social bot detection through data selection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1096–1103.

[16] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "Rtbust: Exploiting temporal patterns for botnet detection on twitter," in *Proceedings of the 10th ACM conference on web science*, 2019, pp. 183–192.

[17] D. M. Beskow and K. M. Carley, "You are known by your friends: Leveraging network metrics for bot detection in twitter," *Open Source Intelligence and Cyber Crime: Social Media Analytics*, pp. 53–88, 2020.

[18] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the 26th international conference on world wide web companion*, 2017, pp. 963–972.

[19] S. Cresci, "A decade of social bot detection," *Communications of the ACM*, vol. 63, no. 10, pp. 72–83, 2020.

[20] F. Wei and U. T. Nguyen, "Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings," in *2019 First IEEE International conference on trust, privacy and security in intelligent systems and applications (TPS-ISA)*. IEEE, 2019, pp. 101–109.

[21] D. Dukić, D. Keča, and D. Stipić, "Are you human? detecting bots on twitter using bert," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2020, pp. 631–636.

[22] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Satar: A self-supervised approach to twitter account representation learning and its application in bot detection," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3808–3817.

[23] S. Feng, H. Wan, N. Wang, and M. Luo, "Botrgcn: Twitter bot detection with relational graph convolutional networks," in *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2021, pp. 236–239.

[24] S. Feng, Z. Tan, H. Wan, N. Wang, Z. Chen, B. Zhang, Q. Zheng, W. Zhang, Z. Lei, S. Yang *et al.*, "Twibot-22: Towards graph-based twitter bot detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 254–35 269, 2022.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

[26] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer, 2018, pp. 593–607.

[27] S. Feng, Z. Tan, R. Li, and M. Luo, "Heterogeneity-aware twitter bot detection with relational graph transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 3977–3985.

[28] Y. Yang, R. Yang, Y. Li, K. Cui, Z. Yang, Y. Wang, J. Xu, and H. Xie, "Rosgas: Adaptive social bot detection with reinforced self-supervised gnn architecture search," *ACM Transactions on the Web*, vol. 17, no. 3, pp. 1–31, 2023.

[29] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2847–2856.

[30] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7693–7711, 2022.

[31] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv preprint arXiv:2003.00653*, vol. 10, no. 3447556.3447566, 2020.

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

[33] S. Cresci, M. Petrocchi, A. Spognardi, and S. Tognazzi, "The coming age of adversarial social bot detection," *First Monday*, 2021.

[34] B. He, Y. Yang, Q. Wu, H. Liu, R. Yang, H. Peng, X. Wang, Y. Liao, and P. Zhou, "Dynamicity-aware social bot detection with dynamic graph transformers," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, no. 646, 2024, pp. 5844–5852.

[35] T. Le, L. Tran-Thanh, and D. Lee, "Socialbots on fire: Modeling adversarial behaviors of socialbots via multi-agent hierarchical reinforcement learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 545–554.

[36] H. Li, M. Xu, S. S. Bhowmick, J. S. Rayhan, C. Sun, and J. Cui, "Piano: Influence maximization meets deep reinforcement learning," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 3, pp. 1288–1300, 2022.

[37] J. Jiang, X. Chen, Z. Huang, X. Li, and Y. Du, "Deep reinforcement learning-based approach for rumor influence minimization in social networks," *Applied Intelligence*, vol. 53, no. 17, pp. 20 293–20 310, 2023.

[38] J. Ma, J. Deng, and Q. Mei, "Adversarial attack on graph neural networks as an influence maximization problem," in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 675–685.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[40] S. Cresci, M. Petrocchi, A. Spognardi, and S. Tognazzi, "Better safe than sorry: an adversarial approach to improve social bot detection," in *Proceedings of the 10th ACM Conference on Web Science*, 2019, pp. 47–56.

[41] S. Najari, M. Salehi, and R. Farahbakhsh, "Ganbot: a gan-based framework for social bot detection," *Social Network Analysis and Mining*, vol. 12, no. 1, p. 4, 2022.

[42] I. Dimitriadis, G. Dialektakis, and A. Vakali, "Caleb: A conditional adversarial learning framework to enhance bot detection," *Data & Knowledge Engineering*, vol. 149, p. 102245, 2024.

[43] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[44] A. Li and Y. Pan, "Structural information and dynamical complexity of networks," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3290–3339, 2016.

[45] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, 2015.

[46] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, " Social Fingerprinting: Detection of Spambot Groups Through DNA-Inspired Behavioral Modeling ," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 04, pp. 561–576, 2018.

[47] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Twibot-20: A comprehensive twitter bot detection benchmark," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 4485–4494.

[48] S. Feng, H. Wan, N. Wang, Z. Tan, M. Luo, and Y. Tsvetkov, "What does the bot say? opportunities and risks of large language models in social media bot detection," *arXiv preprint arXiv:2402.00371*, 2024.

[49] X. Zeng, H. Peng, and A. Li, "Adversarial socialbots modeling based on structural information principles," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 392–400.

[50] Y. Liu, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, vol. 364, 2019.

[51] A. Li, X. Yin, B. Xu, D. Wang, J. Han, Y. Wei, Y. Deng, Y. Xiong, and Z. Zhang, "Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy," *Nature communications*, vol. 9, no. 1, p. 3265, 2018.

[52] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[53] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *arXiv preprint arXiv:2402.07927*, 2024.

[54] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang *et al.*, "A survey on in-context learning," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 1107–1128.

[55] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[56] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey *et al.*, "Multitask prompted training enables zero-shot task generalization," in *International Conference on Learning Representations*.

[57] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*.

[58] W. Luo, W. Wang, X. Li, W. Zhou, P. Jia, and X. Zhao, "Tapo: Task-referenced adaptation for prompt optimization," *arXiv preprint arXiv:2501.06689*, 2025.

[59] D. M. Beskow and K. M. Carley, "Bot-hunter: a tiered approach to detecting & characterizing automated activity on twitter," in *Conference paper. SBP-BRiMS: International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation*, vol. 3, no. 3, 2018.

[60] Q. Guo, H. Xie, Y. Li, W. Ma, and C. Zhang, "Social bots detection via fusing bert and graph convolutional networks," *Symmetry*, vol. 14, no. 1, p. 30, 2021.

[61] T. Magelinski, D. Beskow, and K. M. Carley, "Graph-hist: Graph classification from latent feature histograms with application to bot detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5134–5141.

[62] K.-C. Yang, E. Ferrara, and F. Menczer, "Botometer 101: Social bot practicum for computational social scientists," *Journal of computational social science*, vol. 5, no. 2, pp. 1511–1528, 2022.

[63] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1150–1160.

[64] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[65] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 199–208.

[66] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[67] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the web conference 2020*, 2020, pp. 2704–2710.

[68] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[69] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, pp. 312–322, 2018.

[70] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Information Sciences*, vol. 260, pp. 64–73, 2014.

[71] K. Hayawi, S. Mathew, N. Venugopal, M. M. Masud, and P.-H. Ho, "Deeprobot: a hybrid deep neural network model for social bot detection based on user profile data," *Social Network Analysis and Mining*, vol. 12, no. 1, p. 43, 2022.

[72] F. Morstatter, L. Wu, T. H. Nazer, K. M. Carley, and H. Liu, "A new approach to bot detection: striking the balance between precision and recall," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 533–540.

[73] A. Dehghan, K. Siuta, A. Skorupka, A. Dubey, A. Betlen, D. Miller, W. Xu, B. Kamiński, and P. Prałat, "Detecting bots in social-networks using node and structural embeddings," *Journal of Big Data*, vol. 10, no. 1, p. 119, 2023.

[74] P. Pham, L. T. Nguyen, B. Vo, and U. Yun, "Bot2vec: A general approach of intra-community oriented representation learning for bot detection in different types of social networks," *Information Systems*, vol. 103, p. 101771, 2022.

[75] D. M. Beskow and K. M. Carley, "Its all in a name: detecting and labeling bots by their name," *Computational and mathematical organization theory*, vol. 25, pp. 24–35, 2019.

[76] S. Yang, Q. Du, G. Zhu, J. Cao, L. Chen, W. Qin, and Y. Wang, "Balanced influence maximization in social networks based on deep reinforcement learning," *Neural Networks*, vol. 169, pp. 334–351, 2024.

[77] Y. Cao, H. Peng, A. Li, C. You, Z. Hao, and P. S. Yu, "Multi-relational structural entropy," in *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024, pp. 4289–4298.

[78] J. Wu, X. Chen, K. Xu, and S. Li, "Structural entropy guided graph hierarchical pooling," in *ICML*. PMLR, 2022, pp. 24 017–24 030.

[79] G. Zeng, H. Peng, A. Li, Z. Liu, C. Liu, P. S. Yu, and L. He, "Unsupervised skin lesion segmentation via structural entropy minimization on multi-scale superpixel graphs," in *IEEE ICDM*, 2023.

[80] H. Peng, J. Zhang, X. Huang, Z. Hao, A. Li, Z. Yu, and P. S. Yu, "Unsupervised social bot detection via structural information theory," *ACM Transactions on Information Systems*, pp. 1–42.

[81] Y. Cao, H. Peng, Z. Yu, and P. S. Yu, "Hierarchical and incremental structural entropy minimization for unsupervised social event detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, 2024, pp. 8255–8264.

[82] X. Zeng, H. Peng, and A. Li, "Effective and stable role-based multi-agent collaboration by structural information principles," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, pp. 11 772–11 780, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26390

[83] X. Zeng, H. Peng, A. Li, C. Liu, L. He, and P. S. Yu, "Hierarchical state abstraction based on structural information principles," in *IJCAI*, 2023.

Xianghua Zeng is currently a Ph.D. candidate in the School of Computer Science and Engineering at Beihang University. His research interests include Reinforcement Learning and Structural Information Principle.

Hao Peng is a Professor at the School of Cyber Science and Technology at Beihang University. His research interests include representation learning, social network mining, and reinforcement learning. Dr. Peng has published over 150 research papers in top-tier journals and conferences, including IEEE TPAMI, TKDE, AIJ, JMLR, NeurIPS and ICML. He is the Associate Editor of the IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), International Journal of Machine Learning and Cybernetics (IJMLC), and Neural Networks.

Angsheng Li is a professor at the School of Computer Science, Beihang University. He has worked for the Institute of Software, Chinese Academy of Sciences since 1993 after he finished his Ph.D. to 2018 as an assistant professor (1993-1995), associate research professor (1995-1999), and full research professor (1999-2018), and has been working for the School of Computer Science, Beihang University from 2018. In 2003, he was awarded the Distinguished Young Investigator Award by the National Natural Science Foundation of China. In 2008, he was selected by the Hundred Talent Program of the Chinese Academy of Sciences. His research areas include Computability Theory, Computational Theory, and Information Science. His current interests focus on mathematical principles of the information world and structural information principles of machine learning and intelligent machinery.

Proactive Bot Detection Based on Structural Information Principles

Xianghua Zeng, Hao Peng, Angsheng Li

December 18, 2025

1 Appendix

1.1 Framework Details

1.1.1 Notation Glossary

Table 1: Summary of primary notations used throughout the paper about heterogeneous social network, multi-relational account graph, bot detection, bot modeling, and structural information

Notation	Description
$G_h; G_m; G$	Heterogeneous social network, Multi-relational account graph, Homogeneous graph
$M; U$	Set of social messages, Set of user accounts
$m; u$	Single message, Single account
$X_m; X_u$	Feature matrices of messages and accounts
$x_m; x_u$	Content embedding, Description embedding
E_h	Heterogeneous social interactions
$R; r$	Set of relations, Single relation
E_r^w	Set of weighted edges under single relation
$e_{i,j}^r$	Single directed edge under single relation
W	Weight function for multi-relational edges
$w_{i,j}^r$	Single edge weight under single relation
f_θ	Graph-based bot detection function
$y; \hat{y}$	Single truth label, Single predicted label
σ	Sigmoid function for binary classification
$V; v$	Vertex set, Single graph vertex
$T; K$	Encoding Tree, Maximal tree height
$\lambda; \nu; \alpha$	Root node, Leaf node, Non-root/non-leaf node
L_α	Number of non-leaf node's children
$\mathcal{H}$	One-/High-dimensional structural entropy
d_v	Degree of single graph vertex
$g_\alpha; \mathcal{V}_\alpha; g_{\alpha,\beta}$	Terms associated with structural entropy
$\mathcal{M}$	Markov Decision Process
$\mathcal{I}$	Set of multiple agents
$\mathcal{S}, \mathcal{A}$	State space, Action space
$\mathcal{P}, \mathcal{R}$	Transition function, Reward function
γ	Discount factor for expected reward
s, a, r	Single state, action, and reward
$\pi, \mathcal{Q}$	Policy network, Value function

1.1.2 Encoding Tree Optimization

Figure 1 demonstrates the core optimization operations on the encoding tree: *merge* (η_{mg}) and *combine* (η_{cb}). The *merge* operation collapses a pair of sibling nodes into a single parent node, while the *combine* operation introduces a new parent node above a pair of sibling nodes, restructuring the hierarchy without merging their subtrees. These operations are applied iteratively to optimize the encoding tree structure for minimizing high-dimensional structural entropy.

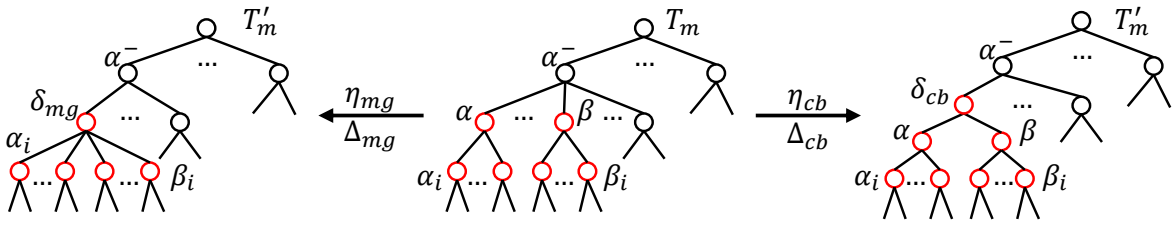

Figure 1: Illustration of the *merge* and *combine* operations on sliding encoding tree nodes. Given an encoding tree for the account graph, the *merge* operator collapses a pair of sibling nodes into a single node, whereas the *combine* operator inserts a new parent node above a pair of sibling nodes while preserving their subtrees.

1.2 Theorem Proofs

1.2.1 Proof of Theorem 1

The transition matrix P is derived from the non-negative graph G'_m, which implies that all entries of P are non-negative.

Furthermore, the strong connectivity of G'_m guarantees that the matrix P is irreducible. This means that, for any pair of accounts u_i and u_j, there exist a positive integer k such that:

$$(P^k)_{i,j} > 0, \tag{1}$$

which indicates that u_i can be reached from u_j in k steps with positive probability, thereby proving that the transition matrix P is irreducible.

Additionally, the elements of each row of the matrix P sums to 1, as shown below:

$$\begin{aligned}
\sum_{u_j \in U} P_{i,j} &= \sum_{u_j \in U} \sum_{r} P^{vr}_{i,r} P^{mr}_{i,j,r} \\
&= \sum_{r} \sum_{u_j \in U} P^{vr}_{i,r} P^{mr}_{i,j,r} \\
&= \sum_{r} \left[P^{vr}_{i,r} \sum_{u_j \in U} P^{mr}_{i,j,r} \right] \\
&= \sum_{r} \left[P^{vr}_{i,r} \sum_{u_j \in U} \frac{A^m_{i,j,r}}{\sum_{u_k \in U} A^m_{i,k,r}} \right] \\
&= \sum_{u_j \in U} \sum_{r} \frac{A^m_{i,j,r}}{\sum_{r' \in R} A^m_{i,j,r'}} \\
&= 1.
\end{aligned} \tag{2}$$

Given that P is non-negative, irreducible, and each row sums to 1, the Perron-Frobenius theorem guarantees the following properties:

- There exists an eigenvalue of P equal to 1, and the corresponding eigenvector π_u satisfies $P\pi_u = \pi_u$.
- The eigenvalue 1 is simple, meaning it has algebraic multiplicity 1.
- The corresponding eigenvector π_e can be chosen to have strictly positive components.
- Other eigenvalues of P have absolute values less than 1.

Since the maximal eigenvalue 1 is simple and the corresponding eigenvector π_u has strictly positive components, the stationary distribution π_u is unique. To ensure that π_u is a valid probability distribution, we normalize it by dividing by its total sum:

$$\pi_u = \frac{\pi_u}{\sum_{u_i \in U} \pi^u_i}, \tag{3}$$

where π^u_i is the stationary probability of account of $u_i \in U$.

Therefore, the stationary distribution π_u exists and is unique, and it corresponds to the eigenvector associated with the eigenvalue 1 of the adjacency matrix P.

1.3 Detailed Derivations

1.3.1 Derivation of One-dimensional Homogeneous Structural Entropy

The random walk on the irreducible, homogeneous graph G can be modeled as a Markov chain. Let the transition probability from vertices $v_i \in V$ to $v_j \in V$ be defined by:

$$p(v_j|v_i) = \frac{w_{i,j}}{d_{v_i}}, \tag{4}$$

where d_{v_i} represents the degree of vertex v_i, i.e., the total weight of the edges incident to v_i.

We now construct a probability distribution π based on the vertex degrees, as follows:

$$\pi_i = \frac{d_{v_i}}{\sum_{v_j \in V} d_{v_j}}, \tag{5}$$

where π_i corresponds to the probability of vertex v_i. After a single-step random walk, the probability being at vertex v_i is updated as follows:

$$\pi_i' = \sum_{v_j} \left[\pi_j \frac{w_{j,i}}{d_{v_j}} \right]. \tag{6}$$

Substituting the expression for π_j:

$$\begin{aligned} \pi_i' &= \sum_{v_j \in V} \left[\frac{d_{v_j}}{\sum_{v_k \in V} d_{v_k}} \frac{w_{j,i}}{d_{v_j}} \right] \\ &= \sum_{v_j \in V} \left[\frac{w_{j,i}}{\sum_{v_k \in V} d_{v_k}} \right]. \end{aligned} \tag{7}$$

Noting that the sum of the edge weights incident to vertex v_i, $\sum_{v_j \in V} w_{j,i}$, is exactly the vertex degree d_{v_i}, we have:

$$\pi_i' = \frac{\sum_{v_j \in V} w_{j,i}}{\sum_{v_k \in V} d_{v_k}} = \frac{v_i}{\sum_{v_j \in V} d_{v_j}} = \pi_i. \tag{8}$$

This confirms that π is a valid stationary distribution for the random walk on G.

The one-dimensional structural entropy of G, as given by Equation **??**, can be written as follows:

$$\begin{aligned} \mathcal{H}^1(G) &= -\sum_{v_i \in V} \frac{d_{v_i}}{\sum_{v_j \in V} d_{v_j}} \cdot \log_2 \frac{d_{v_i}}{\sum_{v_j \in V} d_{v_j}} \\ &= -\sum_{v_i \in V} \pi_i \cdot \log_2 \pi_i. \end{aligned} \tag{9}$$

1.3.2 Derivation of Multi-Relational Entropy Variation

For each non-leaf node $\alpha \in T_m$, the corresponding account subsets of its child nodes satisfy:

$$\bigcap_{i=1}^{L_\alpha} U_{\alpha_i} = \emptyset, \quad \bigcup_{i=1}^{L_\alpha} U_{\alpha_i} = U_\alpha. \tag{10}$$

The terms $\mathcal{V}_\alpha$ in Equation **??** and g_α in Equation **??** can be rewritten as follows:

$$\begin{aligned}
\mathcal{V}_\alpha &= \sum_{u_i \in U} \sum_{u_j \in U_\alpha} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right] \\
&= \sum_{k=1}^{L_\alpha} \left[\sum_{u_i \in U} \sum_{u_j \in U_{\alpha_k}} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right] \right] \\
&= \sum_{k=1}^{L_\alpha} \mathcal{V}_{\alpha_k},
\end{aligned} \tag{11}$$

$$\begin{aligned}
g_\alpha &= \sum_{u_i \notin U_\alpha} \sum_{u_j \in U_\alpha} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right] \\
&= \sum_{k=1}^{L_\alpha} \left[\sum_{u_i \notin U_\alpha} \sum_{u_j \in U_{\alpha_k}} \sum_{r \in R} \left[P_{i,j,r}^{mr} P_{i,r}^{vr} \pi_i^u \right] \right] \\
&= \sum_{k=1}^{L_\alpha} \left[g_{\alpha_k} - \sum_{k' \neq k}^{L_\alpha} g_{\alpha_{k'}, \alpha_k} \right] \\
&= \sum_{i=1}^{L_\alpha} g_{\alpha_i} - \sum_{i \neq j}^{L_\alpha} g_{\alpha_i, \alpha_j}.
\end{aligned} \tag{12}$$

One *merge* operation (η_{mg}) on non-leaf sibling nodes α and β is executed as follows:

$$\begin{cases}
\delta_{mg}^- = \alpha^- = \beta^-, & \\
{\alpha_i}^- = \delta_{mg} & 1 \leq i \leq L_\alpha, \\
{\beta_i}^- = \delta_{mg} & 1 \leq i \leq L_\beta,
\end{cases} \tag{13}$$

where δ_{mg} is the new tree node added as a result of *merge* operation. Before the *merge* operation, the total entropy of the nodes α, β, and their child nodes is given by:

$$\begin{aligned}
&\mathcal{H}_m^{T_m}(G'_m; \alpha) + \mathcal{H}_m^{T_m}(G'_m; \beta) = \\
&-\frac{g_\alpha}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}} - \frac{g_\beta}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_\beta}{\mathcal{V}_{\beta^-}},
\end{aligned} \tag{14}$$

$$\begin{aligned}
&\sum_i^{L_\alpha} \mathcal{H}_m^{T_m}(G'_m; \alpha_i) + \sum_i^{L_\beta} \mathcal{H}_m^{T_m}(G'_m; \beta_i) = \\
&\sum_i^{L_\alpha} \left[-\frac{g_{\alpha_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha_i}}{\mathcal{V}_\alpha} \right] + \sum_i^{L_\beta} \left[-\frac{g_{\beta_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\beta_i}}{\mathcal{V}_\beta} \right].
\end{aligned} \tag{15}$$

After the *merge* operation, the total entropy of the merged node and its children is:

$$\mathcal{H}_m^{T'_m}(G'_m; \delta_{mg}) = -\frac{g_{\delta_{mg}}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_{\alpha^-}}, \tag{16}$$

$$\begin{aligned}
&\sum_i^{L_\alpha} \mathcal{H}_m^{T'_m}(G'_m; \alpha_i) + \sum_i^{L_\beta} \mathcal{H}_m^{T'_m}(G'_m; \beta_i) = \\
&\sum_i^{L_\alpha} \left[-\frac{g_{\alpha_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha_i}}{\mathcal{V}_{\delta_{mg}}} \right] + \sum_i^{L_\beta} \left[-\frac{g_{\beta_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\beta_i}}{\mathcal{V}_{\delta_{mg}}} \right].
\end{aligned} \tag{17}$$

Therefore, the entropy variation $\Delta_{mg}(T_m, \alpha, \beta)$, which measures the change in multi-relational entropy after the *merge* operation, is calculated as follows:

$$\begin{aligned}\sum_{i}^{L_\alpha} \mathcal{H}_m^{T_m}(G'_m; \alpha_i) - \sum_{i}^{L_\alpha} \mathcal{H}_m^{T'_m}(G'_m; \alpha_i) = \\ \sum_{i}^{L_\alpha} \left[-\frac{g_{\alpha_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\alpha} \right] = -\frac{\sum_i^{L_\alpha} g_{\alpha_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\alpha},\end{aligned} \tag{18}$$

$$\begin{aligned}\sum_{i}^{L_\beta} \mathcal{H}_m^{T_m}(G'_m; \beta_i) - \sum_{i}^{L_\beta} \mathcal{H}_m^{T'_m}(G'_m; \beta_i) = \\ \sum_{i}^{L_\beta} \left[-\frac{g_{\beta_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\beta} \right] = -\frac{\sum_i^{L_\beta} g_{\beta_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\beta},\end{aligned} \tag{19}$$

$$\begin{aligned}\mathcal{H}_m^{T'_m}(G'_m; \delta_{mg}) = -\frac{g_{\delta_{mg}}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_{\alpha^-}} \\ = \frac{g_{\alpha,\beta} + g_{\beta,\alpha} - g_\alpha - g_\beta}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_{\alpha^-}},\end{aligned} \tag{20}$$

$$\begin{aligned}\mathcal{H}_m^{T_m}(G'_m; \alpha) + \mathcal{H}_m^{T_m}(G'_m; \beta) - \mathcal{H}_m^{T'_m}(G'_m; \delta_{mg}) = \\ \frac{g_\alpha}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\alpha} + \frac{g_\beta}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\beta} + \frac{g_{\alpha,\beta} + g_{\beta,\alpha}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{mg}}},\end{aligned} \tag{21}$$

$$\begin{aligned}\Delta_{mg}(T_m, \alpha, \beta) = \frac{g_{\alpha,\beta} + g_{\beta,\alpha}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{mg}}} - \\ \frac{\sum_{i \neq j}^{L_\alpha} g_{\alpha_i,\alpha_j}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\alpha} - \frac{\sum_{i \neq j}^{L_\beta} g_{\beta_i,\beta_j}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{mg}}}{\mathcal{V}_\beta}.\end{aligned} \tag{22}$$

On the other hand, one *combine* operation (η_{cb}) on sibling nodes α and β is executed as follows:

$$\delta_{cb}{}^- = \alpha^-, \quad \alpha^- = \delta_{cb}, \quad \beta^- = \delta_{cb}, \tag{23}$$

where δ_{cb} is the new tree node added as a result of the *combine* operation. After the *combine* operation, the entropy sum of these nodes is given by:

$$\mathcal{H}_m^{T'_m}(G'_m; \delta_{cb}) = -\frac{g_{\delta_{cb}}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\delta_{cb}}}{\mathcal{V}_{\alpha^-}}, \tag{24}$$

$$\mathcal{H}_m^{T'_m}(G'_m; \alpha) + \mathcal{H}_m^{T'_m}(G'_m; \beta) = -\frac{g_\alpha}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\delta_{cb}}} - \frac{g_\beta}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_\beta}{\mathcal{V}_{\delta_{cb}}}, \tag{25}$$

For each child α_i of node α, the following equality, which represents its entropy remains unchanged during the *combine* operation, is satisfied:

$$\mathcal{H}_m^{T'_m}(G'_m; \alpha_i) = -\frac{g_{\alpha_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha_i}}{\mathcal{V}_\alpha} = \mathcal{H}_m^{T_m}(G'_m; \alpha_i). \tag{26}$$

Similarly, for each child β_i of tree node β, the following equality holds:

$$\mathcal{H}_m^{T'_m}(G'_m; \beta_i) = -\frac{g_{\beta_i}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\beta_i}}{\mathcal{V}_\beta} = \mathcal{H}_m^{T_m}(G'_m; \beta_i). \tag{27}$$

Therefore, the entropy variation $\Delta_{cb}(T_m, \alpha, \beta)$, which measures the change in entropy after the *combine*

operation, is calculated as follows:

$$
\begin{aligned}
&\Delta_{cb}(T_m, \alpha, \beta) = \left[\mathcal{H}_m^{T_m}(G'_m; \alpha) + \mathcal{H}_m^{T_m}(G'_m; \beta)\right] \\
&- \left[\mathcal{H}_m^{T'_m}(G'_m; \delta_{cb}) + \mathcal{H}_m^{T'_m}(G'_m; \alpha) + \mathcal{H}_m^{T'_m}(G'_m; \beta)\right] \\
&= \frac{g_\alpha}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{cb}}} + \frac{g_\beta}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\beta^-}}{\mathcal{V}_{\delta_{cb}}} - \frac{g_{\delta_{cb}}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{cb}}} \\
&= \frac{g_\alpha + g_\beta - g_{\delta_{cb}}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{cb}}} \\
&= \frac{g_{\alpha,\beta} + g_{\beta,\alpha}}{\mathcal{V}_\lambda} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\delta_{cb}}}.
\end{aligned}
\tag{28}
$$

1.4 Experimental Setup

1.4.1 Dataset Description

We provide detailed descriptions of each publicly available social bot dataset in the Bot Repository—Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22—as follows:

• **Cresci-15**: The Cresci-15 dataset consists of both legitimate and fake Twitter accounts, including sub-datasets such as E13 and TFP. E13 contains 1481 manually verified human accounts, while TFP includes 469 verified human accounts, representing diverse social classes and backgrounds. Additionally, FSF, INT, and TWT are datasets of fake accounts purchased from online marketplaces, containing 1169, 1337, and 845 fake accounts, respectively.

• **Cresci-17**: The Cresci-17 dataset includes both genuine and fake user accounts from Twitter, with several sub-datasets for different types of fake accounts. The genuine dataset includes 3474 verified human accounts. It also features various types of social spambots, such as social spambot1 (991 accounts related to the 2016 Rome mayoral election), social spambot2 (3457 accounts related to the TaInts mobile app), and social spambot3 (464 accounts advertising products on Amazon). Additionally, the Cresci-17 dataset contains fake followers (3351 accounts purchased from online marketplaces) and traditional spambots (1000 accounts exhibiting classic spambot behavior).

• **TwiBot-20**: The TwiBot-20 dataset focuses on studying social bots and their influence on political discourse, particularly during the 2020 U.S. presidential election. It consists of both bot accounts and genuine user accounts that participated in political discussions. The bot accounts were identified using a combination of machine learning algorithms and social graph analysis, displaying behaviors such as automated posting, retweeting, and interaction patterns mimicking human activity.

• **TwiBot-22**: The TwiBot-22 dataset focuses on studying disinformation campaigns and automated propaganda within the context of the 2022 global political landscape. It includes a mix of traditional spambots and social bots that spread misinformation on various political and public health issues. The dataset also contains genuine user accounts, enabling comparative analysis of bot and human behavior on Twitter.

We present the statistical details of four selected datasets for bot detection-Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22-in Table 2. This table highlights the number of human accounts, bot accounts, user accounts, social messages, and interactions between these entities, to quantify the size and scope of each dataset, each representing data from different time periods and social networks.

Table 2: Statistics of the four benchmark bot detection datasets: Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22. For each dataset, we report the number of human accounts, bot accounts, total user accounts, social messages, and interactions.

Dataset	Cresci-15	Cresci-17	TwiBot-20	TwiBot-22
human account	1,950	3,474	5,237	860,057
bot account	3,351	10,894	6,589	139,943
user account	5,301	14,368	229,580	1,000,000
social message	2,827,757	6,637,615	33,488,192	86,764,167
interaction	7,086,134	6,637,615	33,716,171	170,185,937

1.4.2 Baseline Description

We describe the feature-based, content-based, and graph-based baseline methods for bot detection as follows:

• **BotHunter**: BotHunter extracts account, network, content, and timing features, applying random forest classifiers to achieve bot detection.

• **SGBot**: SGBot leverages account metadata and derived features, such as tweet frequency, to identify bots using a random forest classifier.

• **BGSRD**: BGSRD employs BERT and GCN to embed words and user descriptions for achieving bot detection.

• **RoBERTa**: RoBERTa leverages the pre-trained language model RoBERTa to encode account tweets and descriptions, distinguishing bots from human.

• **SATAR**: SATAR uses a co-influence module to aggregate semantic, property, and neighborhood data, enabling self-supervised learning for bot detection.

• **Botometer**: Botometer classifies bots by leveraging over 1,000 features derived from user metadata, interaction patterns, and content analysis.

• **SimpleHGN**: SimpleHGN enhances GAT through learnable edge-type embeddings, residual connections, and L2 regularization on node representations.

• **BotRGCN**: BotRGCN builds a heterogeneous graph of user relationships, applying R-GCN to learn user representations for bot detection.

• **RGT**: RGT employs graph transformers and semantic attention networks to capture both influence and relation heterogeneity in Twitter's social graph.

1.4.3 Large Language Model Configuration

To make the content evolution in SIAMD reproducible, we document here the configuration of the large language model (LLM) used to generate synthetic messages, including the backbone model, prompt construction, decoding setup, safety filters, rate limiting, and a data-card style summary of the resulting synthetic corpus.

Backbone Model. We base our system on the `Llama-2-70b-hf` checkpoint, a 70-billion-parameter variant of the Llama 2 family.[1] The model is instantiated using the `LlamaForCausalLM` architecture with a hidden size of 8192, 80 transformer layers, and 64 attention heads. This places the model in the class of large-scale, decoder-only transformer architectures optimized for causal language modeling, providing sufficient capacity to capture the linguistic and semantic patterns required for high-quality generation in research and professional communication scenarios.

Decoding and Sampling Setup. For text generation, we adopt nucleus (top-p) sampling, a standard stochastic decoding method that balances diversity and reliability. We use a low temperature of 0.1 to sharpen the output distribution, thereby reducing randomness and promoting more deterministic, stable responses, which is desirable for research-oriented writing and behavior simulation. The top-p value is set to 0.5, restricting sampling to the smallest set of tokens whose cumulative probability mass is at least 50%. This prunes low-probability continuations and yields text that is more focused and coherent, while still allowing a limited degree of variability to avoid repetitive or degenerate patterns. We additionally fix the maximum number of newly generated tokens (e.g., 256), and explicitly configure BOS, EOS, and padding tokens to ensure consistent handling of sequence boundaries and batching behavior during inference.

Prompt Construction. As illustrated in our message content generator, each prompt is constructed for an ordered pair of accounts (u_s, u_r), where u_s is the *Account Sender* and u_r is the *Account Receiver*. The background part of the prompt is divided into three fields: *metadata*, *content*, and *structure*. The metadata field summarizes profile statistics for both accounts (e.g., follower and following counts, active years, and verification status). The content field lists the top-k historical tweets from u_s that are most relevant to u_r (e.g., three tweets in our implementation), providing topical context and writing style. The structure field describes the follower and following relationships of both accounts in natural language (e.g., typical categories of accounts they follow or are followed by), thus exposing their structural roles in the network.

Below this background, we append few-shot examples that demonstrate how the model should generate different interaction types. Each example consists of a natural-language **Task** specification (e.g., "Generate the type-Mention message that Account Sender is most likely to send to Account Receiver, based on their past interactions and contextual information.") followed by a corresponding **Output** illustrating a plausible mention or reply. Finally, we append a target **Task Description** of the same form, specifying the desired interaction type (e.g., Mention or Reply) between u_s and u_r and asking the model to generate the next message consistent with the provided background and the few-shot examples. This template is applied uniformly across all sampled account pairs and interaction types, together with the fixed decoding configuration described above.

[1] https://www.modelscope.cn/models/AI-ModelScope/Llama-2-70b-hf/summary

Safety Filters. To mitigate harmful or policy-violating outputs, the system incorporates safety checks both before and after LLM generation. Concretely, the API pipeline integrates a firewall module (LlamaFirewall) that applies PromptGuard2 and additional content filters at two stages:

- **Input filtering.** User prompts (including account descriptions and historical messages) are scanned to detect and block obvious jailbreak attempts, prompt injection patterns, and clearly harmful or disallowed requests before they reach the model.
- **Output filtering.** The raw model response is further inspected, and content that violates safety policies (e.g., clearly harmful, illegal, or disallowed topics) is intercepted or sanitized prior to being accepted as synthetic data.

This two-stage filtering strategy reduces the risk of generating unsafe content and improves robustness against adversarial prompt manipulation when deploying SIAMD in practical settings.

Rate Limiting. To ensure stable operation and prevent resource exhaustion, we apply request-level rate limiting using an in-memory token-bucket mechanism configured with a refill rate of 0.16 requests per second and a maximum bucket size of 2 requests. This configuration is sufficient for our research-oriented, batch-style generation regime, where the emphasis is on controllable, high-quality text and reproducible experiments rather than serving large-scale real-time traffic.

Synthetic Content Data Card. We summarize the synthetic message corpus generated by this prompting scheme as follows:

- **Purpose.** To simulate realistic mention and reply interactions between user accounts, conditioned on historical content, profile metadata, and local network structure, in order to study coordinated bot behavior and proactive detection.
- **Source signals.** For each account pair (u_s, u_r), we use (i) profile metadata for sender and receiver (follower/following counts, active years, verification status), (ii) the top-k historical tweets from u_s that are most relevant to u_r, and (iii) natural-language descriptions of their follower and following neighborhoods. These fields form the background section of the prompt.
- **Generation process.** Given the background, we construct prompts by appending few-shot Task/Output examples for different interaction types (Mention and Reply), followed by a final Task Description specifying the target type for the pair (u_s, u_r). Messages are then generated with the fixed Llama-2–70B decoding setup (temperature 0.1, top-$p = 0.5$, maximum new tokens, BOS/EOS configuration) and passed through the safety filters and rate limiter described above. Random seeds and configuration files are stored so that the same prompts and model settings can be replayed to regenerate the corpus.
- **Intended use and limitations.** The synthetic messages are intended solely for research on social bot modeling and proactive detection under controlled, reproducible conditions, and are not designed for direct deployment in real-world social platforms without additional auditing and alignment.

Our choice of Llama 2–70B as the backbone model is supported by empirical evaluation on grouped academic benchmarks such as code understanding, commonsense reasoning, world knowledge, and reading comprehension. Table 3 summarizes the reported performance and motivates our preference for Llama 2–70B over smaller variants in this work.

Table 3: Overall performance of LLaMA2-70B on grouped academic benchmarks, including Code, Commonsense Reasoning, World Knowledge, and Reading Comprehension. Reported scores are aggregated over representative tasks in each category.

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension
Llama 2	7B	16.8	63.9	48.9	61.3
Llama 2	13B	24.5	66.9	55.4	65.8
Llama 2	70B	37.5	71.9	63.6	69.4

1.5 Supplementary Experiments

1.5.1 Overall Performance

Table 4 summarizes the detection performance of SIAMD and representative feature-based, content-based, and graph-based baselines in terms of confusion-matrix entries and AUROC on four datasets: Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22. These results further highlight the detection performance advantage of our model and demonstrate that this advantage is robust to the choice of classification threshold, as reflected by the AUROC metric.

Table 4: Detection performance of SIAMD and baseline methods on four detection datasets (Cresci-15, Cresci-17, TwiBot-20, and TwiBot-22), reported in terms of confusion-matrix entries (TP, FP, TN, FN) and area under the ROC curve (AUROC).

Method	**Cresci-15**					**Cresci-17**				
	TP	**FP**	**TN**	**FN**	**AUROC**	**TP**	**FP**	**TN**	**FN**	**AUROC**
BotHunter	307	4	191	28	0.985	930	12	335	159	0.941
RoBERTa	315	8	187	20	0.992	1049	86	261	40	0.970
BotRGCN	332	16	179	3	0.995	1054	94	253	35	0.975
RGT	332	12	183	3	0.997	1070	71	276	19	0.988
SIAMD	333	4	191	2	0.999	1068	16	331	21	0.994
Method	**Cresci-15**					**Cresci-17**				
	TP	**FP**	**TN**	**FN**	**AUROC**	**TP**	**FP**	**TN**	**FN**	**AUROC**
BotHunter	571	213	311	88	0.897	3577	1674	73136	21613	0.729
RoBERTa	477	169	355	182	0.870	3098	1796	73014	22092	0.714
BotRGCN	594	109	415	65	0.951	11789	3971	70839	13401	0.866
RGT	600	91	420	59	0.955	7582	2527	72283	17698	0.792
SIAMD	631	91	433	28	0.982	13628	3536	71274	11562	0.880

1.5.2 Sensitive Analysis

To investigate the robustness of the proposed method with respect to LLM text generation, we conducted a sensitivity analysis on the temperature and top-p decoding parameters of the LLM using the Cresci-15 and Cresci-17 datasets. Specifically, we first fixed the top-p parameter at 0.5 and varied the temperature parameter within the range $[0.1, 0.5]$; then we fixed the temperature parameter at 0.1 and varied the top-p parameter within the range $[0.3, 0.7]$. As shown in Figure **??**, although gradually increasing the temperature and top-p values may introduce more randomness into the LLM-generated content and reduce the reliability of the generated text as input, our SIAMD method still exhibits strong active detection performance. Although there are some fluctuations, SIAMD still maintains a stable performance advantage compared with the baseline methods. This robustness stems from the fact that the generated messages are tightly anchored to real historical content, metadata, and structural context provided in the prompts and are used in conjunction with multi-relational structural signals rather than as a standalone source of supervision.

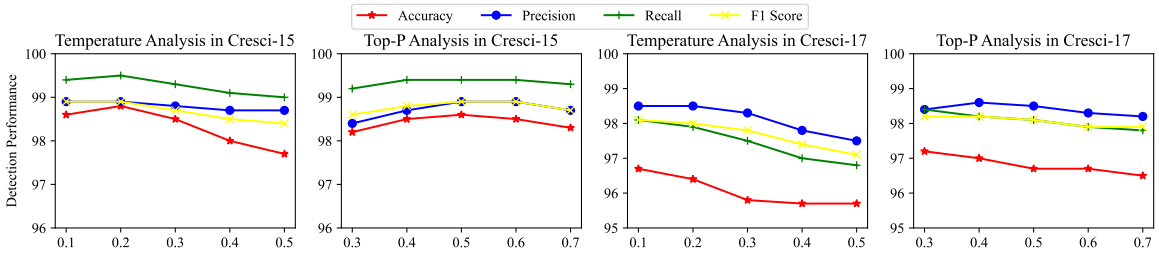

Figure 2: Sensitivity analysis of SIAMD with respect to LLM text generation parameters (temperature and top-p) on the Cresci-15 and Cresci-17 datasets. We report Accuracy, Precision, Recall, and F1 score under different decoding settings.

1.5.3 Example of LLM-based Message Generation

To make the text generation procedure more concrete, Figure 3 illustrates an example prompt constructed for a single modeled bot (Account *Sender*) and its target user (Account *Receiver*). The prompt given to the LLM is organized into three background blocks—metadata, content, and structure—followed by few-shot examples and the final task description.

Metadata Block. We first verbalize basic profile statistics of both accounts as short natural-language sentences, for example: *"Account Sender: follower count: 10, following count: 336, active years: 3, verified: false. Account Receiver: follower count: 23, following count: 650, active years: 3, verified: false."* This provides the model with coarse information about the size, activity level, and status of each account.

Content Block. Next, we insert the three tweets from the Sender that are most similar to the Receiver's embedding based on the inner product between their features, formatted as an ordered list: *"Top three tweets from Sender most relevant to Receiver are: (1) InMyDream I have an element, LIGHTNING!! #InMyDreamW (2) Que rico es acostarse a descansar después de un día tan duro ahora mañana a seguir con mi rutina diaria (3) WhatIFindAttractive #WhatIFindAttractive."* These examples expose the LLM to the Sender's typical topics, language choice, and style.

Structure Block. We then describe the local network neighborhood of both accounts by listing representative follower and following relationships in natural language, e.g.: *"Follower Relationship for Sender: (1) account 40: Andrea, 21, #untin CS student (2) account 291: mi guardo intorno e mahhhh (3) account 2880: No need of one ust make love to my follow button. Following Relationship for Receiver: (1) account 169: Del resto, è solo la curiosità che mi fa svegliare la mattina (2) account 824: Cittadino (3) account 3062: for people who are serious about starting or running an online home business."* This block summarizes how the two accounts are embedded in the surrounding social graph.

Few-Shot Examples. Using the same background information, we append several labeled examples that pair an interaction type with an appropriate historical message. For instance, one example specifies a *Retweet* interaction: *"Task: Generate the type-Retweet message that Account Receiver is most likely to send to Account Sender, based on their past interactions and contextual information. Output: RT @SenderAccount: InMyDream I have an element, LIGHTNING!! #InMyDream Me encanta la idea de soñar en grande! (I love the idea of dreaming big!)"* A second example specifies a *Mention* interaction with a different output message. These examples demonstrate to the LLM how background information and interaction types map to concrete message text.

Target Task For Generation. Finally, we append the target instruction whose output is unknown during inference, for example: *"Task: Generate the type-Relay message that Account Sender is most likely to send to Account Receiver, based on their past interactions and contextual information."* No output is provided for this last task in the prompt. The LLM receives the entire concatenated text—background blocks, few-shot examples, and the new task instruction—and generates a relay-type message that is consistent with the accounts' metadata, historical content, social neighborhood, and the patterns illustrated by the examples.

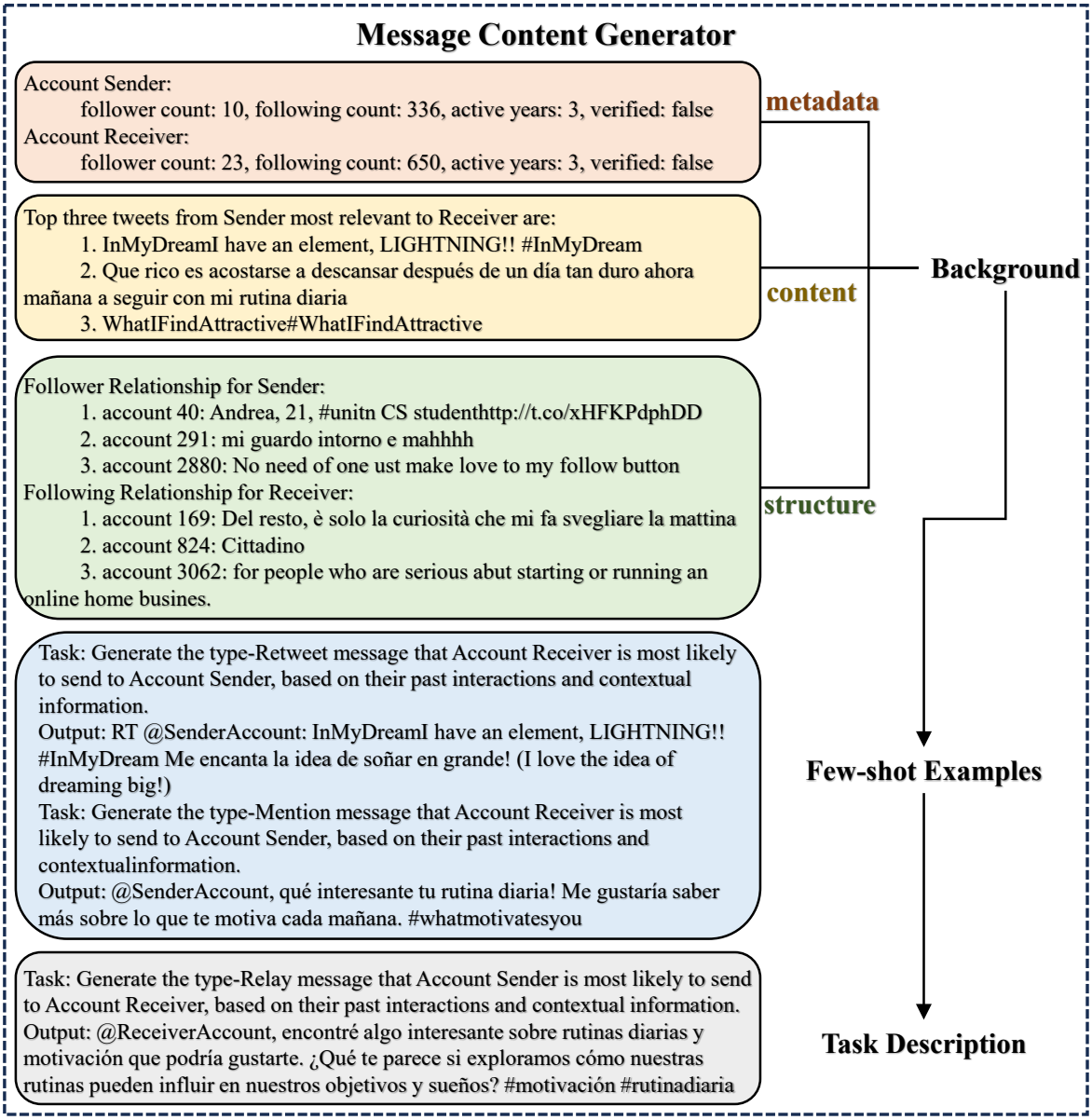

Figure 3: Illustrative example of prompt construction for LLM-based message generation for a single modeled behavior. The prompt for one modeled bot (Account Sender) and its target user (Account Receiver) is organized into three background blocks (metadata, content, and structure), followed by few-shot examples and the target generation task.