International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-024-02392-0

ORIGINAL ARTICLE

GNNRI: detecting anomalous social network users through heterogeneous information networks and user relevance exploration

Yangyang Li[1] · Xinyue Sun[2] · Renyu Yang[3] · Xiaoyang Sun[4] · Shiru Chen[5] · Shuhai Wang[6] · Md Zakirul Alam Bhuiyan[7] · Albert Y. Zomaya[8] · Jie Xu[4]

Received: 2 January 2024 / Accepted: 20 September 2024

Abstract
Detecting anomalous users in social networks is an imperative but challenging task. The increasing complexity of interpersonal behaviors and interactions further complicates the development of effective user anomaly detection techniques. Current state-of-the-art methods heavily rely on static personal features, making it difficult to quantify the hidden relevance of user behaviors through traditional feature engineering. This loss of accuracy is exacerbated by the rise of sophisticated camouflage and disguising techniques, which blur the distinction between anomalous and regular users. In this paper, we present GNNRI, an innovative framework for detecting anomalous users in social networks. Our approach leverages a network representation learning model and a heterogeneous information network (Hin) to explore hidden semantic connections from user metadata, tweets, and interaction information. We extract both user metadata and behavioral features to construct a Hin and introduce two distinct learning layers to explore explicit and implicit user relevance. First, we employ a relation-based self-attention layer to aggregate neighbor node closeness under specific relations and across different relationships. Subsequently, we apply graph convolution network-based convolutional learning layers, which enhance embedding effectiveness by capturing graph-wide node similarity. We evaluate GNNRI using real-world datasets, and our results demonstrate that it outperforms all other comparative baselines, achieving approximately 90% accuracy for user classification, with a 5–15% improvement over other GNN variants. Notably, even when using only 20% of the data for training, GNNRI achieves 87.8%, 86.57%, and 87.1% accuracy for detecting zombies, spammers, and bots, respectively.

Keywords Heterogeneous graph · Social network · Abnormal social user · Heterogeneous graph neural network

✉ Xinyue Sun
xysun@hit.edu.cn

✉ Renyu Yang
renyuyang@buaa.edu.cn

[1] Academy of Cyber, Beijing 100085, China

[2] School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China

[3] School of Software, Beihang University, Beijing 100083, China

[4] School of Computing, University of Leeds, Leeds LS2 9JT, UK

[5] Inspur Yunzhou Industrial Internet Co. Ltd., Beijing 100191, China

[6] School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang 050043, China

[7] Department of Computer Science, Fordham University, New York 10458, USA

[8] School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia

1 Introduction

Social networks such as Twitter, Facebook, Instagram, Weibo, etc., have become essential platforms for individual socialization and media engagement. Meanwhile, there are an ever-increasing number of anomalies mainly controlled by automated software in the form of zombie users, spammers, social bots, etc. [1, 2]. They substantially negatively impact social network platforms through malicious attempts, misinformation, and fraudulence. Specifically, zombie users are those fake followers who create an illusion of a high reputation and credibility via boosting the number of followers [3, 4]. Spammers and social bots attract public attention through active social behavior [5, 6]. Spammers aggressively post harmful content such as commercial or personal advertisements, adult advertisements, e-magazines, and serial letters [7–9] whilst social bots are designated to disseminate disinformation and propaganda that can sway public opinion [10]. Such deception can reach a vast community and lead to cascading and devastating consequences. [8].

Early approaches of social anomaly detection [11–13], simply relied on shallow classifiers based on straight-forward extraction of static account information, i.e., extracting the screen name, id, and the number of followers from metadata, assuming that early-stage anomalous users tend to adopt randomly-generated usernames. Unsurprisingly, the accuracy drastically drops when such static information is no longer the critical characteristic of mocked users. In reality, the evolving camouflage techniques such as adversarial text generation [14, 15] will make it even harder to discriminate between anomalous users and real human [16]. There are a huge body of deep neural network based approaches for spammer and bot detection. Refs. [17–19] perform tweet-granularity content analysis and detection where metadata is used as contextual information of the individual user. Ref. [20] design a GCN-based anti-Spam model that integrates both heterogeneous and homogeneous graph to capture the local and global context of a comment. Ref. [21] leverage convolutional mechanism for combining different single-view attributed graph embedding. However, all these works heavily exploit profile data and textual information of social users and rarely consider the hidden behaviors among relevant users. Graph neural networks (GNNs) based detection approaches [22–24] be widely applied to capture the explicit inter-connections between users (e.g., in Fig. 1, Bob posts a tweet and interacts with Alice in the comments. At the same time, Dan and a bot are involved in the same topic discussion). While neighbor information can be better leveraged for representation learning, the hidden and higher-order relational nature of social networks is still under-studied. Abnormal behaviors are often hidden within the substantial number of implicit interactions between users on social platforms. For example, in Fig. 1, Bot-1 and Bot-2 are independent without direct interactions, but they may share similar comment and retweet patterns and thus should be assimilated into the detection model. The third generation of bots since 2016, with a potent mixture of human operations and automated bot behaviors, managed to disguise themselves and survived platform-level detectors by using traditional classifiers or classic GNN models [25, 26]. Heterogeneous information network (HIN) has become the most helpful methodology to fuse various types of information and uncover the hidden semantics [27]. However, HIN-based approaches tend to have inherent limitations in meta-relations, i.e., meta-path and meta-graph, particularly when the data resources in real application scenarios cannot underpin the modeling of HIN.

In this paper, we present GNNRI, an innovative GNN-based anomaly detection framework enhanced by harvesting multiple Relations and Implicit connections between social users. The framework mainly explores and exploits the hidden semantic connectivity based on the whole bunch of user meta, tweets, and interaction information with the aid of network representation learning and HIN. In the data preparation and modeling phase, we extract a wide variety of user features – from meta-features to other behavioral operations – and transform the heterogeneous *entities* and their *relations* into nodes and edges of a HIN.

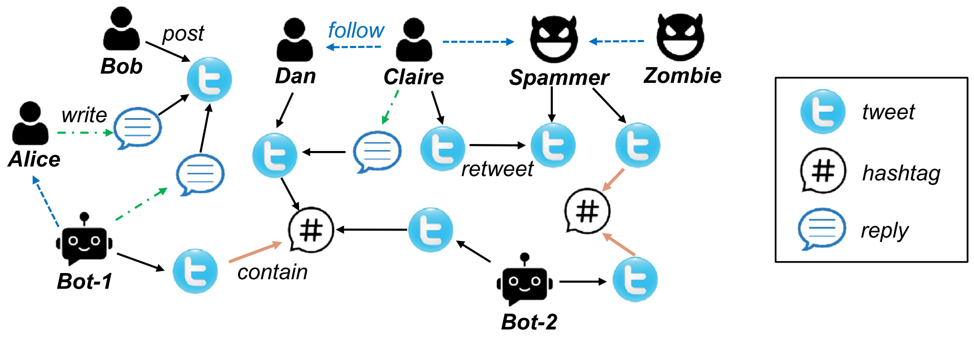


Fig. 1 An example of multiple entities and relations in social networks

At the core of GNNRI are the numerical embedding of user features and implicit relevance between users under different relations by integrating the advancement of graph attention network (GAT) and graph convolution network (GCN). We exploit rich semantic meta-structures, including meta-path and meta-graph based on the established Hin, to represent the implicit connections between different users. We introduce a relationship-based self-attention mechanism in the model (RSL) to explore the high-order implicit relevance between any pair of users and to aggregate initial information from different relations. The model includes a two-level attention aggregation procedure for fusing the feature embedding: first calculating neighbor users' closeness under a given relation and then consolidating the contributions made from different relations into the embedding. To make better use of users' adjacency in the social network and enhance further the implicit association of users, we stack additional convolutional embedding layers (ICL) upon the attention layer that can exploit the whole-graph wide user similarities based on the number of relationship instances between any two users. This procedure measures the weighted number of pathways across different meta-structures and a higher value indicates a closer relevance. ICL can therefore enhance the implicit association of users without explicit connections.

To evaluate the effectiveness and efficiency of GNNRI, we collect 10,316 user data and 21,360 tweet data from Twitter platform for initializing the Hin. We elaborately label the users in the dataset into four categories (normal users, zombies, spammers, and social bots) and compare our GNNRI against the state-of-the-art baselines. We conduct comprehensive experiments in terms of user classification and clustering tasks. Results show our approach can achieve roughly 90% accuracy for user classification, with about 5–5% improvement against other GNN variants. Particularly, even when using 20% data for training, GNNRI can achieve 87.8%, 86.57% and 87.1% accuracy for zombie, spammer and bot classification, respectively. For user clustering, our approach can achieve the highest NMI and ARI compared with other GNN-based approaches. In addition, GNNRI has the quickest convergence with less fluctuation, indicating the competitive efficiency in the model training.

In particular, we make the following contributions:

- For the first time, we propose a novel framework for categorizing social network users based on heterogeneous information network, which utilizes user meta features, behavior features, and textual semantic features (Sect. 3.3).
- We develop a relationship-based self-attention mechanism to measure user closeness within relationships, enhancing detection of key interactions in the network (Sect. 3.4).
- We introduce an advanced convolutional layer to analyze high-order interactions, improving anomaly detection by leveraging hidden connections among users (Sect. 3.5).
- We prove the superiority of our model through classification tasks and clustering tasks, and our model gains a higher accuracy in anomaly detection that the state-of-the-art methods. Meanwhile, our model is interpretative. In addition, we further illustrate parameter sensitivity experiments (Sect. 4).

The rest of the paper is organized as follows. Section 2 clarifies the problem, defines social network anomaly user detection, and lists the preliminaries used in the article. Secttion 3 presents the our framework GNNRI. Section 4 describes the proposed approach's experiments and performance results compared with the baseline methods. Section 5 lists the related work about our research. Section 6 summarizes the paper and outlines future work.

2 Preliminaries and problem definition

This section presents the background and preliminaries before introducing the problem scope and framework overview.

2.1 User anomaly in social network

An anomaly, aka. outlier, is referred to as *a case that does not follow the same model as the rest of the data* [28]. Abnormal users in social networks are those individuals or groups who fail to conform to the characteristics defined by standard patterns. It manifests in activities that differ from normal user behaviors, such as unusual views and behaviors. In this paper, we primarily focus on three types of abnormal users:

- **Zombie user.** Also known as fake followers, zombie users are used to boosting the number of followers to a target user for popularity promotion [29, 30]. Zombie followers merely follow users who want attention and behave harmlessly—most time keeping silent, with few interactions such as comments and retweets. However, they still have a non-negligible indirect societal and economic impact via changing public influence.
- **Spammer**. A spammer usually posts commercial content, tempts users to click malicious links, and redirects to spammer-controlled third-party domains [9]. Spammers interact with users by replying to comments or retweeting to enhance their credibility. They may post similar content through coordinated activities to increase their influence [31]. Moreover, spammers are engaged in topical discussions to disseminate their content.

- **Social bot**. Social bots sway public opinion by posting content with certain hashtags [32]. They make a huge attempt to imitate and change user viewpoints and behaviors by interactions [33]. Social bots disproportionately spread news and opinions with ulterior motives and typically increase the content in the early dissemination stages. Through *replying* and *mentioning* operations, they can even target influential users.

2.2 Preliminaries

Heterogeneous information network (Hin) [34]. A Hin is denoted as $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{K}, \mathcal{R}, \varphi, \phi)$, where $\mathcal{V}$ denotes the nodes set, $\mathcal{E}$ denotes the edges set, $\mathcal{K}$ denotes the node types set and $\mathcal{R}$ denotes the edge types set. In real-world settings, there may be multiple types of nodes or edges, i.e., $|\mathcal{K}| + |\mathcal{R}| > 2$. Each individual node $i \in \mathcal{V}$ is associated with a node type mapping function $\varphi : \mathcal{V} \rightarrow \mathcal{K}$. Each individual edge $e \in \mathcal{E}$ has an edge type mapping function $\phi : \mathcal{E} \rightarrow \mathcal{R}$. The Hin constructed in this paper encompasses different nodes and edges, indicating entities and their interactions. Figure 1 exemplifies the user behaviors that can be observed on a social network such as Twitter or Weibo. Users are inter-connected through four basic operations—*following, commenting/replying, posting/retweeting*, and *hashtagging*. Accordingly, there are four distinct entities—*user* `(U)`, *tweet* `(T)`, *comment* `(C)`, and *hashtag* `(H)`. Figure 1 exemplifies a typical use case in Twitter. For instance, Dan posts a tweet containing a hashtag, while Bot-1 and Bot-2 post tweets with the same hashtag. Claire retweets what a Spammer account originally tweets.

Meta-schema [34]. Given a HIN $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{K}, \mathcal{R}, \varphi, \phi)$, the network schema for network $\mathcal{G}$ can be denoted as $\mathcal{T}(\mathcal{K}, \mathcal{R})$, a directed graph with the node type set $\mathcal{K}$ and edge types set $\mathcal{R}$. In simple words, the meta schema comprehensively depicts the node types and their relations in a HIN. It provides a meta template to guide the exploration of node relationships and extract subgraphs from the HIN. Figure 2a summarizes the high-level relationships between entities. Therefore, we can employ the entities and operations as nodes and edges for constructing the Hin.

Meta-path and meta-graph [34, 35]. Given a meta schema, a meta-path $\mathcal{MP}$, denoted as $K_1 \xrightarrow{R_1} K_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{l-1}} K_l$, is a path on $\mathcal{T}(\mathcal{K}, \mathcal{R})$ that defines a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ between node type K_1 and K_l. Herein, $\circ$ indicates the relation composition operator. A path $p = v_1 - v_2 - \cdots - v_l$ between v_1 and v_l in the network G follows the meta-path $\mathcal{MP}$, if $\forall i$, v_i is of type S_i. We call p a *meta-path instance* of $\mathcal{MP}$. Figure 2b demonstrates the meta-paths defined on the meta-schema.

A meta-graph $\mathcal{MG}$, is a directed acyclic graph (DAG) with a single source node and a single target node. The nodes and edges in $\mathcal{MG}$ are confined to $\mathcal{K}$ and $\mathcal{R}$, respectively. We define the meta-graph instance in the same manner as the meta-path model. Intuitively, a meta-graph conveys more complex and richer relationships than a meta-path.

Mining such semantic relationship is the cornerstone of subsequent tasks such as classification, clustering, etc. In practice, based on the observations in the social network platforms, we extracted the four most valuable meta-paths s_1 to s_4 to outline that users can have indirect connectivity. For instance, the meta path s_4 `U-T-H-T-U` means two users post tweets that contain the same hashtag topic, although they may not have direct friendship (following and follower) or interactions (comments or replies). It is particularly effective when pinpointing social bots, as they typically lead a trend by posting substantial content related to the same hashtag in short order. Furthermore, we leverage meta graphs s_5 and s_6—composing a group of meta paths—to emphasize the importance of "retweet-reply" and "retweet-reply-hashtag" relationships between users simultaneously. For example, s_5 indicates the user posts a tweet retweeted and commented by another user.

2.3 Problem definition

This work aims to effectively detect abnormal users in social networks, depending on the personal features of users and relational features in social networks. Our work addresses two deficiencies facing abnormal user detection: loss of accuracy and ignorance of implicit relationships. Specifically, most existing studies tackle user anomalies in a unified way without differentiating their specific characteristics. Such approaches will result in a loss of accuracy in anomaly detection. On the other hand, there are numerous intrinsic

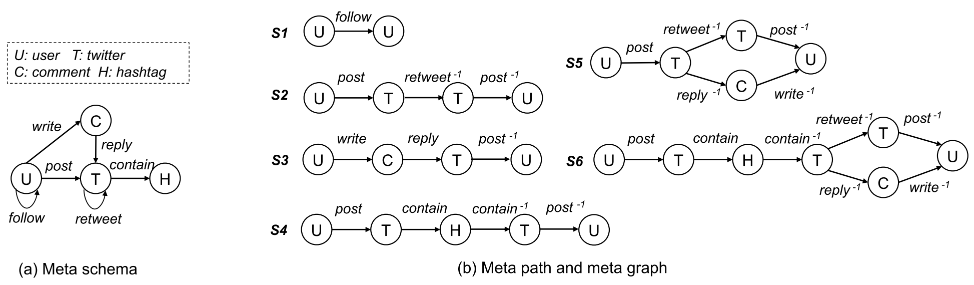


Fig. 2 **a** Meta schema and **b** meta structures (meta-paths and meta-graphs)

relationships, i.e., hidden relevance stemming from user behaviors, which can hardly be quantified by conventional feature engineering.

The detection procedure is typically regarded as a classification. Formally, we aim to take as input features $\mathcal{X}$ of social users and their previous labels (*standard/zombie/spammer/bot*) $\mathcal{T}$ to predict the type t of any target user. For the first time, we propose to solve the problem of abnormal user detection with heterogeneous information networks, which can take into account the relationships of social networks. In this work, our framework mainly relies on user meta-features and fully explores the explicit and implicit relationships hidden in social networks for abnormal user identification.

3 GNNRI: relation-aware attentive and convolutional embedding for heterogeneous user relevance

In this section, we first overview GNNRI and then detail how we design the two learning layers in GNNRI to project the user relevance into the embedding representation of each user.

3.1 Overview

Figure 3 describes the overview architecture. GNNRI offers an elaborated extraction of entities in a social network based on feature engineering—extracting data from Twitter via the Twitter API and selecting them to obtain the meta-features, behavior features, and textual semantic features of users. These features provide four types of entities, including social user, tweet, comment, and hashtag, and six interactions. GNNRI then builds up the HIN by organizing entities and the extracted interactions into nodes and edges within a HIN (Sect. 3.3). As *user* is the target object of the detection, we aim to generate a homogeneous graph that only contains user entities from the HIN. To do so, GNNRI primarily employs meta structures, including meta path and meta graph, to extract the relational connectivity among users.

At the detector's core is the numerical embedding of nodes and edges in the HIN. In learning the numerical embedding, it must fully exploit node affinities within a given relationship and properly aggregate the information about homogeneous graphs of different relationships. To fully exploit the individual contribution of relationships, which are expressed in meta structures, *Anomaly Detector* involves two distinct layers: relationship-based self-attention layer (RSL) and a GCN-based convolutional embedding layer (ICL). We design separate strategies to learn the embedding: In RSL, we firstly calculate the closeness of explicit neighbor users under a given relationship at node level (i.e., intra-relationship) (Sect. 3.4.1) and consolidate the contributions made from different relationships at semantic level (i.e., inter-relationship) (Sect. 3.4.2). We employ ICL to enhance further the implicit association of users without explicit connections. It exploits the whole-graph-wide user similarities based on the in-between number of relationship instances of any two users. This procedure measures the weighted number of pathways across different meta-structures, and a higher value indicates a closer relevance (Sect. 3.5). We then feed the similarity matrix and the preliminary embedding result from RSL into a GCN model to further improve the numerical embedding. Eventually, *Anomaly Classifier* digests the learned vector embeddings to learn a classification model and determines if the given user behavior is typically or malicious. General purpose techniques including Random Forest, Logistic Regression, SVM, etc. can be adopted for implementing the classifier (Sect. 3.6).

3.2 Main pipeline

Since node embedding is the crucial procedure of GNNRI, we first outline how the embedding procedure is broken down into pipelines. As shown in Fig. 3, the meta-feature matrix U will be streamed into GNNRI and passed to RSL where intra-relationship, and inter-relationship attention aggregation procedure will be carried out. Specifically, a $D \times D$ weighted matrix $\mathbb{W}^{s_i}$, i.e., the adjacency matrix, represents closeness of any two neighboring nodes in each relational meta-structure s_i. This aggregation mechanism finally yields an inter-mediate embedding matrix E^{s_i}, of shape $N \times D$ for all nodes, i.e., each row $E_x^{s_i}$ represents the embedding vector for node x (Sect. 3.4.1). Afterward, E^{s_i} will be applied further into a nonlinear transformation to consolidate the embedding results across all relational meta-structures. The trainable weight matrix $\mathcal{W}^{s_i}$ is used to transform the node features and other weight parameters. RSL will eventually produce the $N \times D$ embedding matrix T (Sect. 3.4.2).

Subsequently, ICL will take over the control of embedding to dig out long-distance user connections. The essence of ICL is to uniformly and generally calculate the structure proximity between any two users, even without direct connections. To generalize the similarity between any two users, we incorporate all adjacency matrices into an overall similarity matrix M. The intention is to add all possible pathways between the two users within all graphs about all relational meta-structures. M and T will be assembled further, with the aid of a GCN convolutional process, generating the final embedding matrix H, which is of shape $N \times D'$, where D' is the final embedding dimension (Sect. 3.5). Table 1 summarizes all notations used in the paper.

Table 1 The variables and feature tags of social users

Symbol	Description
S	Relationship collection
s_i	One type of meta-structures, i.e. relationships
U	The raw node feature matrix of users
$\mathbb{W}^{s_i}$	Learnable weight matrix for closeness degree in RSL
$c_x^{s_i}(y)$	The importance (closeness) for node y to node x based on relationship s_i in RSL
$E_x^{s_i}$	The node representation of user x under relationship s_i in RSL
$\mathcal{W}^{s_i}$	Trainable weight matrix for aggregation across relationships in RSL
b^{s_i}	Bias under meta-structure s_i for aggregation across relationships in RSL
$\phi_x^{s_i}$	The importance of the relationship s_i for user x in RSL
$\alpha_x^{s_i}$	The weight for relationship s_i to node x in RSL
Ψ^{s_i}	Adjacency matrix to calculate the instances of connection between two users
ω_{si}	Weight for relationship s_i in ICL
$M(x, y)$	The similarity calculated between user x and user y in ICL
T	The output matrix of user embeddings of RSL
H	The output matrix of user embeddings of ICL, also the final embedding matrix

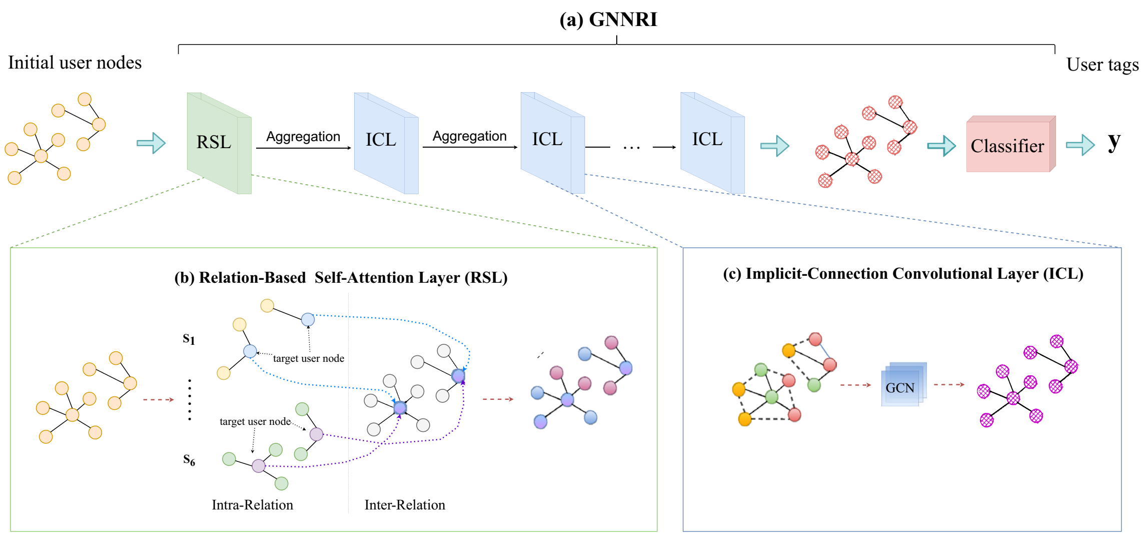


Fig. 3 The overall architecture of GNNRI

3.3 HIN construction

We exploit Twitter API to identify the pertaining entities and the in-between connectivity. We initialize the procedure by extracting meta-features before recognizing the entities and building up meta structures to represent user behaviors and ascertain their underlying semantics. Initially, the feature extraction module transforms the original information into a heterogeneous graph. The edges between nodes in the heterogeneous graph are established based on the account's friend relationships and interactions in the social network platform.

We retrieve each account's meta-features and description features as its initial node feature, and extra tweet features

Table 2 User features

Feature name	Type	Feature name	Type
id_str	String	followers_count	int
name	String	friends_count	int
screen_name	String	listed_count	int
url	String	favourites_count	int
description	String	statuses_count	int
created_at	String	default_profile	Boolean
verified	Boolean	protected	Boolean

and entities are extracted using NLPtool https://github.com/explosion/spaCy from the original tweets. The meta-features of users serve as the node features of the social user node and then as the initial input features of our framework. Table 2 demonstrates the metadata details extracted from the *user* object. We use the $N \times D$ raw feature matrix U to store the meta information, where N and D are the numbers of users and dimensions of meta-features, respectively.

Specifically, the feature engineering include the following steps:

- *Account nodes:* We encode the *description* field of the account metadata into a 64-dimension vector through a pre-trained language model Word2Vec. To enrich the original node features, we consider the number of followers and the number of friends since such numbers are the most representative social identity of a Twitter user. Additionally, binary value is used to indicate *default_profile*, *verified* and *profile_use_background_image*, count the length of *screen_name*, *name* and *description* fields and the number of digits in *screen_name* and *name* fields.
- *Tweet nodes*: We embed the text of the original tweet by the pre-trained language model Word2Vec into a 300 dimension vector. We incorporate the raw tweet context a variety of additional information – the number of retweets, the number of replies, the number of favorites, the number of mentioning of the original tweet, and the number of hashtags and the number of URLs involved in the tweet.
- *Hashtag and entity nodes:* Similarly we embed the text of hashtag and entity into a 300-dimension vector.

3.4 Relation-aware attention-based node embedding

One-hop connection, aka. direct link, is the most direct interpersonal connection between two users via a meta-path or a meta-graph. The critical procedure for obtaining the representation of a user is to aggregate all embeddings of its neighboring users. To exploit the intrinsic difference among different users and the impact disparity under different meta-structure (meta-path or meta-graph), we can break down the one-hop connection into two aspects: differentiating the proximity of different users within a meta-structure and adding up contributions from different meta-structures.

3.4.1 Node embedding under a given relationship

Closeness degree. Inspired by the self-attention mechanism [36], we use the *degree of closeness* between users to quantify the individual importance of neighbor users. More specifically, given a set of meta paths and meta graphs $S = \{s_1, s_2, \dots, s_m\}$, the objective is to calculate the intra-relationship node aggregation for each corresponding meta-structure. Assuming a pair of user nodes (x, y) is connected via a meta path s_i. We use $closeness(x, y; s_i)$ to denote the importance of y onto x:

$$closeness(x, y; s_i) = \sigma\left(u_x \mathbb{W}^{s_i} u_y^T\right), \tag{1}$$

where u_x and u_y denote the embedding vectors of user features (e.g., 64-dimension numerical vector) and $\mathbb{W}_{s_i}$ is a matrix only determined by the meta-structure s_i, while σ denotes the activation function. The closeness index is asymmetric.

We intend to calculate the structure connection across all users under different meta structures. Meta-paths and meta-graphs over types and structures indicate semantic-explainable similarities between two user nodes. We use the adjacency matrix Ψ^{s_i} to count the instances of connection between two user nodes. Given a meta-path s_i, $K_1 K_2 \cdots K_l$, the adjacency matrix can be calculated by:

$$\mathbb{W}^{s_i} = R_{K_1K_2} \cdot R_{K_2K_3} \cdots \cdot R_{K_{l-1}K_l}, \tag{2}$$

where $R_{K_jK_{j+1}}$ is the relation matrix between entity K_j and K_{j+1}. $\mathbb{W}_{s_i}(x, y)$ represents the count of meta-path instances between node x and node y.

As will be shown in Sect. 4, we collected 10,316 social users for the experiment where $N = 10{,}316$. $\mathbb{W}_{s_i}$ will be a $10{,}316 \times 10{,}316$ matrix and the embedding vector of users will be 64 dimensions. The overall embedding matrix E^{s_i} contains 10,316 rows and 64 columns. For meta path s_2 (U – T – T – U) in Fig. 2, $\mathbb{W}^{s_2} = R_{UT} \cdot R_{TT} \cdot R_{TU} = R_{UT} \cdot R_{TT} \cdot R_{UT}^T$. $\mathbb{W}_{i,j}^{s_2} > 0$ indicates that user i and user j are associated with each other, and higher value indicates more pathways that can connect the two users through the given semantic.

Likewise, for a given meta graph s_j consisting of a set of meta-paths, $\{s_{j1}, s_{j2}, \dots, s_{jJ}\}$, the node adjacency matrix is:

$$\mathbb{W}^{s_j} = \mathbb{W}^{s_{j1}} \odot \cdots \odot \mathbb{W}^{s_{jJ}}. \tag{3}$$

For example, $\mathbb{W}^{s_8} = R_{UT} \cdot ((R_{TT} \cdot R_{TU}) \odot (R_{TC} \cdot R_{CU}))$. For a given meta graph s_j consisting of a set of meta-paths, $\{s_{j1}, s_{j2}, \dots, s_{jJ}\}$, the matrix can be formulated as:

$$\mathbb{W}^{s_j} = \mathbb{W}^{s_{j1}} \odot \mathbb{W}^{s_{j2}} \odot \cdots \odot \mathbb{W}^{s_{jJ}}, \tag{4}$$

where $\odot$ is the operation of *Hadamard Product*. For example, for a pair of nodes (x, y) connected via the meta-graph s_5 (a combination of s_2 and s_3). The matrix is $\mathbb{W}_{s_5} = \mathbb{W}_{s_2} \odot \mathbb{W}_{s_3}$.

Node embedding. Within a meta-structure s_i, for each user node x, it is only the neighbor node y that counts. That means we measure the closeness with masked attention. We then normalize the closeness to obtain the weight coefficient $c_x^{s_i}(y)$ through softmax function:

$$c_x^{s_i}(y) = \mathrm{softmax}(closeness(x, y; s_i)). \quad (5)$$

Therefore, we calculate the node embedding of user x under s_i by carrying out weighted aggregation among all neighbor nodes:

$$E_x^{s_i} = \|_{h=1}^{H} \sigma\Big(\sum c_x^{s_i}(y) \cdot u_y\Big). \quad (6)$$

Specifically, we employ a multi-head attention mechanism for precisely incorporating the neighbors' embedding. To make the training procedure more stable, we repeat the computation H times and concatenate the results.

3.4.2 Embedding aggregation across relationships

We transform every user vector in the R^{s_i} through a layer of nonlinear transformation including the trainable weight matrix $\mathcal{W}^{s_i}$ and bias b^{s_i}. The importance of the meta structure s_i for user x is calculated by:

$$\phi_x^{s_i} = (y^{s_i})^T \cdot \tanh\left(\mathcal{W}^{s_i} E_x^{s_i} + b^{s_i}\right), \quad (7)$$

where y_{s_i} denotes the weight vector for meta-structures. The importance score depends both on the meta-structure and the user node. $\mathcal{W}^{s_i}$, b^{s_i} and y^{s_i} are shared by all nodes in the same group. That means that even the same meta-structure has different scores for different nodes, but at the same time, these scores are closely related. We use softmax function to normalize the weight of each meta structure s_i, as following:

$$\alpha_x^{s_i} = \mathrm{softmax}\big(\phi_x^{s_i}\big). \quad (8)$$

We aggregate the embeddings of a node x under different meta structures:

$$T_x = \sum_{i=1}^{m} \alpha_x^{s_i} E_x^{s_i}. \quad (9)$$

Performing the aggregation on each user will eventually form the overall matrix T, and each row vector represents the individual embedding of each node.

3.5 ICL to explore implicit relevance

Following the similarity calculation presented in [37], we define a user similarity between user x and user y, primarily considering the meta-structure instances between these two users. Given a collection of meta structures $\{s_1, s_2, \dots, s_m\}$, the similarity is calculated by:

$$M(x, y) = \sum_{i=1}^{m} \omega_{s_i} \frac{2 \times \mathbb{W}^{s_i}(x, y)}{\mathbb{W}^{s_i}(x, x) + \mathbb{W}^{s_i}(y, y)}, \quad (10)$$

where $\mathbb{W}^{s_i}(x, y)$ is the number of s_i instances bridging between x and y, and ω_{s_i} represents the weight of s_i. Notably, we use a *learnable* parameter vector $\boldsymbol{\omega} = \{\omega_{s_1}, \omega_{s_2}, \dots, \omega_{s_m}\}$ to denote weights of all meta structures.

We can then use the calculated similarity to indicate the connectivity between any pair of user instances. We construct an $N \times N$ weighted adjacency matrix M to store the semantic similarity among N users. To form a GCN model, we firstly calculate $\tilde{M} = M + I_N$, $\hat{M} = D^{-\frac{1}{2}}\widetilde{M}D^{-\frac{1}{2}}$ where I_N is the identity matrix and D is a diagonal matrix such that $D_{ii} = \sum_j M_{ij}$. We apply two layers in the GCN:

$$H = H^{(1)} = \sigma_2\big(\hat{M}\sigma_1\big(\hat{M}H^{(0)}W^{(0)}\big)W^{(1)}\big), \quad (11)$$

where σ_1 and σ_2 denote different activation functions. $W^{(0)}$ and $W^{(1)}$ are learnable weight matrices between the input and hidden layer and between the hidden and output layer. $H^{(0)}$ is the input feature matrix, and $H^{(1)}$ is the output feature matrix after applying the first GCN layer. $H^{(0)}$ is set to be the embedding matrix T. Eventually, we use each row of the embedding matrix H, i.e., $H_1, H_2, \dots, H_N$, to numerically represent each user.

GNNRI allows for pipelining any arbitrary tiers of ICLs with RSLaccording to different requirements. There are many variants of GNNRI, e.g., GNNRI-RSL where we only retain RSL with ICL switched off and GNNRI-ICL where only convolutional layers are activated. We provide a configurable layer number for variable performance, and we will evaluate the impact of the layer number on the overall detection performance in Sect. 4.

3.6 Multi-anomaly detection and optimization

We adopt a *Logistic Regression* classifier to achieve the final step of anomaly detection. During the model training, the classifier absorbs the labels of the training nodes. It minimizes the cross-entropy between the ground truths and the predicted results to achieve the *multi-class* node classification. Specifically, the loss function F is formalized as follows:

$$F = -\sum_{\lambda \in V_\Lambda} y_\lambda \log P_\lambda, \quad (12)$$

where V_Λ is the index set of user nodes with labels while y_λ is a binary vector indicating the label of a given node y and P_λ is the probability of neural network prediction.

4 Experiment

In this section, we describe the experiments, including classification and clustering, and parameter sensitivity, and provide some analysis of the results.

4.1 Experiment setup

Platforms. Evaluation is conducted on a multi-core server with a 64-core Intel Xeon CPU @2.40 GHz with 512 GB RAM and 8×NVIDIA Tesla P100 GPUs. The server runs Ubuntu 20.04 LTS with Linux kernel 5.4.0. Our model is implemented using Python 3.5.2.

Dataset. We conduct experiments on two Twitter bot datasets Vendor-19 [38] and TwiBot-20 [39], by far the largest datasets in the public domain. Vendor-19 provided a collection of fake followers deriving from several companies and TwiBot-20 dataset exposes users' social relationships and enables the use of advanced graph representation-based algorithms. We mix the datasets with a dataset of benign accounts Verified [40] and tagged a totality of 10,316 social users, including 2507 normal users, 2683 zombie users, 2751 spammers, and 2375 social bots. We collected 21,360 tweets posted by these users. Table 3 describes the data statistics.

Methodology. To evaluate the effectiveness, we compare GNNRI with baselines based on traditional neural networks, random walk, and GNN and Botometer [41], the classic bot detection website. We perform node classification and node clustering representative tasks in network representation learning studies. The node classification distinguishes four types of users to demonstrate the effectiveness in detecting anomalous users (Sect. 4.2). We feed the obtained node embeddings to the K-means algorithm to perform clustering, setting the number of clusters the same as the ground truth classes (Sect. 4.3). For the ablation study, we evaluated GNNRI and its variables GNNRI-RSL and GNNRI-ICL, respectively. Furthermore, we experiment on parameter sensitivity to study the detailed impact of ICL layers and the dimension of embedding vectors on the accuracy (Sect. 4.4). We eventually showcase a comprehensive case study to discuss some key observations and findings from our experimental evaluation (Sect. 4.5).

Baselines. We compare our proposed framework with the following baselines, including the state-of-the-art heterogeneous network representation learning approaches and traditional social bot detection approaches. For all baseline models, we use the open-source implementations.

Botometer [41] is an online detection tool to check bots on Twitter but is not suitable for classifying anomalous users. Thus we only use it to distinguish anomalies from humans. Botometer is a representative method based on manual feature engineering [40].

MLP [42] is a multilayer perceptron that takes multiple datasets as inputs at the input layer. After complex calculations, the hidden layers take the vectors from the input layer and pass them to the output layer. There is a non-linear mapping framework of input and output vectors.

DeepWalk [43] takes the homogeneous network as input and generates the embeddings for all graph nodes. It leverages the random walk algorithm to aggregate the information of neighborhood nodes. We use DeepWalk to get the representations of the users.

Metapath2vec++ [44] constructs the neighborhood context for the node embedding by using a meta-path-based random walk and then generates node embeddings via the skip-gram framework. In our experiments, we report metapath performance with the best result.

GCN [45] is a semi-supervised homogeneous graph convolutional network model that retains the graph nodes' feature information and structure information. It uses the first-order approximation of the Chebyshev polynomial to complete an efficient graph convolution architecture.

GAT [46] is a semi-supervised homogeneous graph model that utilizes an attention mechanism to aggregate neighborhood information of graph nodes. To adapt the homogeneity of GAT, we ignore the differences between meta paths and meta graphs and execute GAT on the entire graph.

HAN [47] is a heterogeneous graph representation learning model that utilizes predefined meta paths and hierarchical attentions for node vector embedding.

HetGNN [22] uses a random walk with a restart strategy to sample strongly correlated heterogeneous neighbors and perform intra-node-type and inter-node-type aggregations but does not leverage the semantically meaningful meta-paths and meta-graphs.

Table 3 Statistics of Dataset

Features	Node pair (X–Y)	Number of node X	Number of node Y	Number of node pair	Type of user	Number
1386	User-User	10,316	10,316	3×951	Human	2507
	User-Tweet	10,316	21,060	21,060	Zombie User	2683
	Tweet-Tweet	21,060	21,060	9003		
	User-Comment	10,316	15,207	15,207	Spammer	2751
	Tweet-Comment	21,060	15,207	15,207	Social Bot	2375
	Tweet-Hashtag	21,060	27	11,068		

HGT [23] introduces an attention mechanism to the interactions of different types of nodes, and it conducts across-layer messages passing distinct kinds of neighbors to obtain high-order aggregation.

MAGNN [48] presents a meta-path based graph neural network aggregation for node embedding. It unifies different types of nodes and applies intra-metapath and inter-metapath aggregations. It neglects the comprehensively semantic meaningful meta-graphs.

RSHN [49] simultaneously captures the heterogeneous graph structure and implicit relation structural information to explore the interactions between edges and learn node and edge type representation. It is independent of meta-paths and meta-graphs.

RCRFR [50] combines the RoBERTa classifier and a random forest regressor with similarity analysis to evaluate tweet similarities and user profile features, employing a voting system to enhance detection accuracy.

BotBuster [51] uses a mixture of experts approach, where each expert analyzes a specific aspect of account information, such as the username, and combines their assessments to estimate the likelihood of an account being a bot, demonstrating superior performance across multiple datasets.

GNNRI-RSL and GNNRI-ICL are variants of GNNRI. GNNRI-RSL only retains RSL without ICL while only ICL is activated in GNNRI-ICL.

Model training. In general, we set the learning rate, the number of attention heads, and the dropout rate to be 0.0005, 8, and 0.4, respectively. For all the deep learning models, we employ the Adam optimizer with a learning rate of 0.005 and the weight decay of 0.001. We train them for 100 epochs with the dropout rate of 0.5 and utilize early stopping with patience of 5. We ascertain the best setting of parameters such as the network layer and embedding dimension, according to the study of parameter sensitivity which will be shown in Sect. 4.4. For a fair comparison, we set the final embedded vector dimension obtained by all the above methods to 64 and use the same splits of training, validation, and testing sets. We tune all baseline parameters through the policies developed in their works and report their performance with the optimal settings. More specifically, for Node2Vec and Metapath2Vec, we set the number of walks per node, the max walk length, and the window size to be 10, 100, 8, respectively. The number of negative samples is set to be 5, if applicable. For GNNs such as GCN, GAT, HAN, MAGNN, we set the parameters suggested by their original papers: all GNNs keep 3 layers with dropout rate 0.5. For RSHN, we use 2-layer HGNN coupled with 1-layer RLG-NN with 8 hidden units We set the number of attention heads to 8 and the dimension of the attention vector to 128. Note that all the above parameter settings also apply to all experiments.

Table 4 Descriptions of evaluation metrics

Metrics	Description
TP	The number of malicious Apps that are correctly identified
TN	The number of benign Apps that are correctly identified
FP	The number of benign Apps that are mistakenly identified
FN	The number of malicious Apps that are mistakenly identified
$F1$	$2 * Precision{*}Recall/(Precision + Recall)$
Accurate	$(TP + FN)/(TP + TN + FP + FN)$

Metrics. To quantitatively evaluate the performance of GNNRI, we use the following metrics in the experiments: (i) for the classification experiments, we use Accuracy and F1 scores (see Table 4), two high-is-better metrics, to evaluate the effectiveness of detection. (ii) to quantify the clustering effectiveness, we use *normalized mutual information (NMI)* [52] and *adjusted rand index (ARI)* [53] scores. We use ten-fold cross validation and calculate the average accuracy to provide an assurance of unbiased and accurate evaluation.

4.2 Classification effectiveness

Overall accuracy. We choose 20%, 40%, 60%, 80% of the data samples for training embedding models and the residual for testing. Table 5 illustrates the $F1$ and accuracy scores of each model when varying the training ratio. In all cases, the classification accuracy will intuitively increase when more samples are involved in the training procedure. In general, GNNRI-RSL or GNNRI-ICL can achieve competitive classification accuracy when compared with other graph-based models such as Metapath2Vec++, HAN, GCN, GAT, HetGNN, HGT, MAGNN, and RSHN. Similar results can be observed in the F1 score comparison. In fact, graph-based representation learning models manage to exploit relational information for effectively embedding any user information as a vector, and among them, it can be observed that the HIN-based approach obtains better results. Even more so, joint effort of both models into GNNRI can significantly improve the precision compared against all other baselines. This is because the embedding effectiveness is further enhanced by exploring the implicit relevance in ICL. In addition, limited deviation can be found in GNNRI compared with other baselines, indicating our approach can deliver more stable classification.

Accuracy of individual user category. Figure 4 demonstrates the classification effectiveness over different user categories and Table 6 shows the specific result when training ratio is set to be 80%. Obviously, GNNRI outperforms baselines over all cases, similarly to the overall accuracy comparison. Among different categories, the classification

Table 5 Classification precision comparison

Metrics	Accuracy				Micro-F1			
Training	20%	40%	60%	80%	20%	40%	60%	80%
MLP	0.5807 ± 0.0069	0.6069 ± 0.0050	0.6133 ± 0.0036	0.6285 ± 0.0083	0.5805 ± 0.0081	0.6091 ± 0.0078	0.6106 ± 0.0029	0.6269 ± 0.0052
Deepwalk	0.6008 ± 0.0026	0.6166 ± 0.0053	0.6397 ± 0.0030	0.6514 ± 0.0043	0.6056 ± 0.0036	0.6199 ± 0.0049	0.6363 ± 0.0052	0.6398 ± 0.0096
Metapath2vec	0.6499 ± 0.0089	0.6587 ± 0.0095	0.6679 ± 0.0033	0.6732 ± 0.0082	0.6483 ± 0.0094	0.6596 ± 0.0044	0.6683 ± 0.0096	0.6707 ± 0.0065
RCRFR	0.7055 ± 0.0045	0.7086 ± 0.0039	0.6697 ± 0.0043	0.7117 ± 0.051	0.6513 ± 0.0042	0.6902 ± 0.0054	0.6608 ± 0.0061	0.7100 ± 0.0033
GCN	0.7216 ± 0.0028	0.7202 ± 0.0060	0.7394 ± 0.0026	0.7507 ± 0.0075	0.7207 ± 0.0074	0.7193 ± 0.0027	0.7339 ± 0.0031	0.7501 ± 0.0031
GAT	0.7893 ± 0.0066	0.7959 ± 0.0013	0.8095 ± 0.0011	0.8352 ± 0.0087	0.7772 ± 0.0059	0.7902 ± 0.0062	0.8038 ± 0.0019	0.8193 ± 0.0026
HAN	0.8281 ± 0.0010	0.8322 ± 0.0086	0.8489 ± 0.0043	0.8633 ± 0.0070	0.8211 ± 0.0006	0.8302 ± 0.0071	0.8503 ± 0.0053	0.8621 ± 0.0038
HetGNN	0.8145 ± 0.0009	0.8238 ± 0.0072	0.8342 ± 0.0093	0.8539 ± 0.0055	0.8201 ± 0.0069	0.8270 ± 0.0021	0.8404 ± 0.0026	0.8597 ± 0.0074
HGT	0.8269 ± 0.0059	0.8306 ± 0.0071	0.8495 ± 0.0083	0.8601 ± 0.0022	0.8312 ± 0.0047	0.8395 ± 0.0061	0.8493 ± 0.0032	0.8619 ± 0.0021
MAGNN	0.8290 ± 0.0063	0.8364 ± 0.0089	0.8477 ± 0.0065	0.8639 ± 0.0039	0.8221 ± 0.0056	0.8292 ± 0.0117	0.8513 ± 0.0079	0.8672 ± 0.0094
RSHN	0.8366 ± 0.0034	0.8467 ± 0.0016	0.8604 ± 0.0106	0.8701 ± 0.0052	0.8359 ± 0.0037	0.8424 ± 0.0038	0.8533 ± 0.0081	0.8714 ± 0.0005
BotBuster	0.8419 ± 0.0057	0.8604 ± 0.0044	0.8885 ± 0.0073	0.8671 ± 0.0047	0.8362 ± 0.0033	0.8824 ± 0.0045	0.8489 ± 0.0071	0.8870 ± 0.0022
GNNRI-RSL	0.8306 ± 0.0064	0.8389 ± 0.0032	0.8564 ± 0.0094	0.8698 ± 0.0071	0.8356 ± 0.0047	0.8419 ± 0.0041	0.8524 ± 0.0078	0.8733 ± 0.0053
GNNRI-ICL	0.7255 ± 0.0072	0.7401 ± 0.0080	0.7597 ± 0.0029	0.7743 ± 0.0066	0.7231 ± 0.0073	0.7411 ± 0.0085	0.7502 ± 0.0031	0.7774 ± 0.0074
GNNRI	**0.8826 ± 0.0019**	**0.8892 ± 0.0036**	**0.8938 ± 0.0053**	**0.9016 ± 0.0076**	**0.8807 ± 0.0022**	**0.8851 ± 0.0026**	**0.8898 ± 0.0032**	**0.8927 ± 0.0034**

The values in bold represent the results of our proposed GNNRI method

over normal users can target the best effectiveness: the values in Fig. 4a is higher than values in other heatmaps while the classification on spammer data samples experiences the lowest effect.

Impact of increasing epoch on accuracy. Figure 5 shows the change procedure of accuracy score over all baselines with the increment of training epoch. Observably GNNRI has the quickest convergence speed to reach a decent accuracy (roughly 90%). In comparison, other bases either have larger accuracy fluctuation or need more epoch and time for convergence.

4.3 User clustering

We evaluate the user representations obtained by the above algorithms through the clustering task. After the training of these algorithms, we can get the embeddings of all user nodes. We perform node clustering using K-Means with a cluster number of four, i.e., the number of user types: human, zombie user, spammer, and social bot. The process is repeated for 20 times and the final result is shown in Fig. 6. Overall, graph-based representation learning models perform competitively in clustering tasks. GNNRI performs better than all baselines. Therefore, it can be assumed that GNNRI makes effective improvements in the clustering task to obtain good results.

4.4 Impact of parameters

Impact of ICL **layers**. Figure 7a shows the impact of the number of ICL layers on the detection accuracy. Increased ICL layers achieve limited accuracy improvement—the accuracy reaches the peak when carrying out two ICL layers but goes down if more layers are added on. This is because too many aggregations would reduce the difference between categories. Therefore, we only double ICL for relation aggregation.

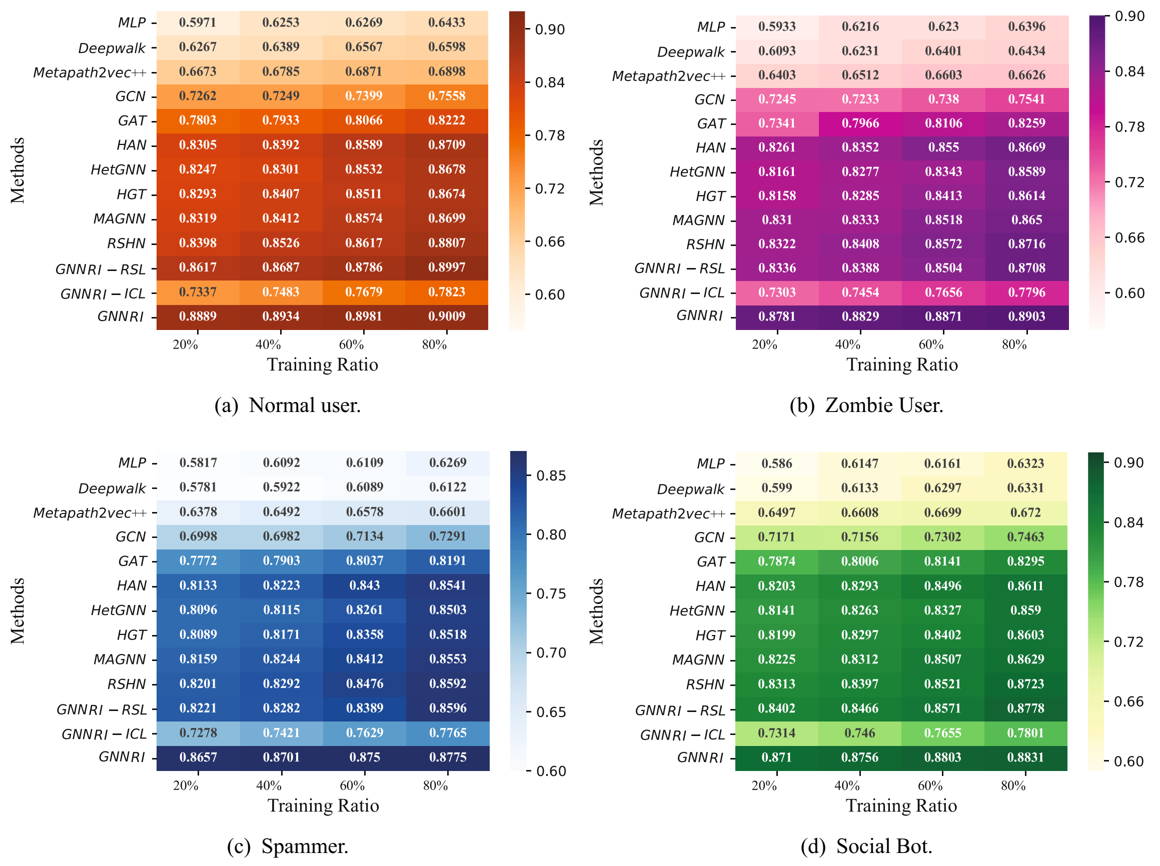


Fig. 4 Classifications accuracy under different user categories

Table 6 Classification precision in each user category

Categories	Human		Zombie user		Spammer		Social bot		All categories	
Metrics	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	Micro-F1
Botometer	–		–		–		–		0.8507	0.8569
MLP	0.6445	0.6433	0.6289	0.6396	0.6154	0.6269	0.6294	0.6323	0.62955	0.6283
Deepwalk	0.6577	0.6598	0.6443	0.6434	0.6279	0.6122	0.6465	0.6331	0.6441	0.6352
Metapath2vec	0.6872	0.6898	0.6712	0.6626	0.6589	0.6601	0.6756	0.672	0.673225	0.6707
RCRFR	0.7550	0.7376	0.6786	0.6857	0.7311	0.6608	0.7004	0.6984	0.7501	0.7356
GCN	0.7602	0.7558	0.7557	0.7541	0.7398	0.7291	0.7507	0.7463	0.7516	0.7439
GAT	0.8398	0.8222	0.8304	0.8259	0.8205	0.8191	0.8387	0.8295	0.83235	0.8211
HAN	0.8689	0.8709	0.8678	0.8669	0.8557	0.8541	0.8623	0.8611	0.863675	0.8632
HetGNN	0.8602	0.8678	0.8533	0.8589	0.8497	0.8503	0.8526	0.859	0.85395	0.8597
HGT	0.8659	0.8674	0.8564	0.8614	0.8512	0.8518	0.8587	0.8603	0.85805	0.8604
MAGNN	0.8693	0.8699	0.8612	0.865	0.8567	0.8553	0.8638	0.8629	0.86275	0.8634
RSHN	0.871	0.8807	0.8688	0.8716	0.8549	0.8592	0.8657	0.8723	0.8651	0.8711
BotBuster	0.8760	0.8948	0.8812	0.8866	0.8560	0.8705	0.8704	0.8780	0.8654	0.8809
GNNRI-RSL	0.8766	0.8997	0.8725	0.8708	0.8603	0.8596	0.8707	0.8778	0.870025	0.8771
GNNRI-ICL	0.7888	0.7823	0.7693	0.7796	0.7661	0.7765	0.7674	0.7801	0.7729	0.7798
GNNRI	**0.9055**	**0.9009**	**0.8944**	**0.8903**	**0.8897**	**0.8775**	**0.8902**	**0.8831**	**0.895**	**0.8897**

The values in bold represent the results of our proposed GNNRI method

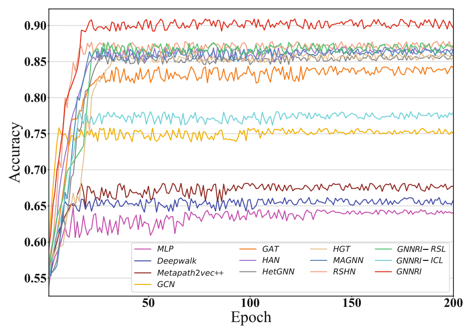


Fig. 5 Accuracy and epoch

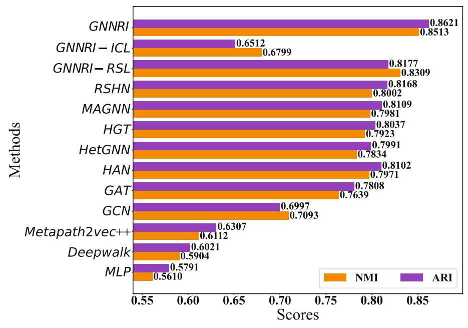


Fig. 6 NMI and ARI values

Impact of varying embedding dimensions. Figure 7b depicts the impact of the embedding dimensions on the accuracy. The increment of embedding dimensions leads to a picking-up trend until the dimension number reaches 64, followed by a constant reduction. This is because a large enough embedding dimension can provide the required information for the classification and redundancy will be introduced once the dimension surpasses a certain threshold. In addition, the dimension of the vector y observably impacts on the accuracy. As shown in Fig. 7c, as the dimension size grows, the performance of GNNRI grows accordingly and reaches the peaking accuracy when the dimension size is set to be 128. Similarly, the accuracy will drop drastically due to overfitting.

4.5 Case study

In this subsection, we conduct several case studies and explore the implications behind.

4.5.1 Implications of calculating closeness degree

We first randomly select several user instances under different user categories to investigate how the elements constitute the closest neighbors of a given user. This analysis can showcase the specific contribution of relevant users to the targeted user, correspondingly indicating their contributions to the numerical embedding and anomaly detection. It is worth noting that the targeted user itself will be included in the closeness calculation due to the self-attention mechanism in the RSL and we then visualize the residual top-5 closest neighbors of a given user.

Finding-1: normal users and social bots tend to be close to their own kind. Users of the same kind are often among the closest neighbors of a specific user. For instance, as shown in Fig. 8a, d, Four fifths of closest neighbors of the normal user U1037 are normal ones and all top-4 closest

Fig. 7 Impact of parameters

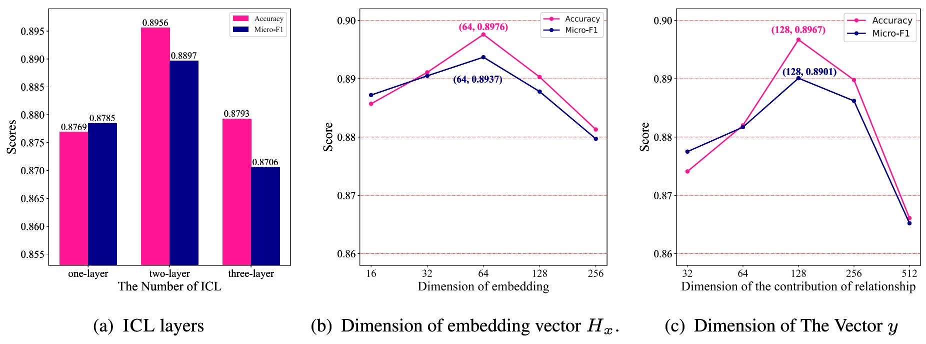
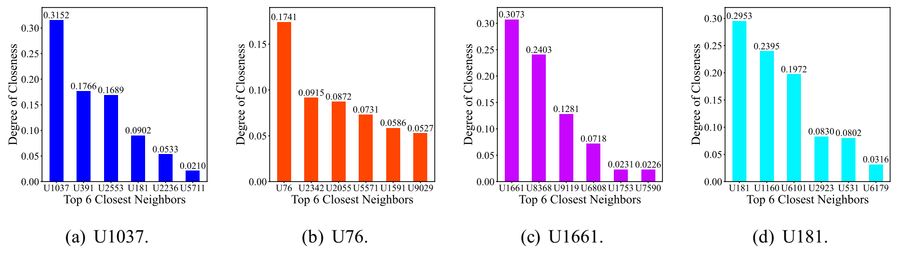


Fig. 8 The closest neighbors of U1037 (normal user), U76 (zombie user), U1661 (spammer) and U181 (social bot)



neighbors of the social bot U181 are social bots. This is simply because normal users and social bots are engaged in so many relationships that users involved in the same tweets via either retweet or comment will be heavily associated with each other. The advancement of bot technology makes it increasingly sophisticated for users, particularly the inexperienced ones, to differentiate the roles of social bots. This gives rise to certain social bots being included in the closest neighbors of a normal user, and vice versa.

Finding-2: Zombie users and spammers tend to own diverse close neighbors. One will be closely associated with zombie users as long as he would expand his social reputation, no matter what type of user he belongs to. For instance, U5571 (spammer) and U9029 (social bot) are among the closest neighbors of the zombie user U76. Similarly, spammers can be widely related to other users. Spammers involved in the same activities or coordinated to launch specific attacks—such as advertising to generate sales or disseminating viruses and phishing information, etc.—are likely strongly correlative in the proximity calculation. In addition, for the purpose of marketing promotion, spammers may be engaged in certain hashtags, wherein a close connection with social bots or normal users is established. This explains why the spammer U161 in Fig. 8c are closely related to U1753 (zombie user) and U7590 (social bot).

4.5.2 Analysis of importance of relationships in RSL

Figure 9a–d show the distinct contributions of each relationship to different types of users. We provide some findings below.

Finding-3: relationship s_2 (retweeting) and s_3 (commenting) substantially contribute to users under all categories. As the most frequent operations in social networks, retweeting and commenting are intrinsically of critical importance in social interactions, particularly for normal users (e.g., U1037) and spammers (e.g., U1661). Interestingly, some zombie users even occasionally retweet or comment other posts of normal users in order to flourish a targeted tweet and user, and camouflage themselves with such usual operations. This is aligned with the observations in Fig. 9b. Social bots are also keen to reply to comments or retweet to attract more attention from others and make them interested in the hashtag.

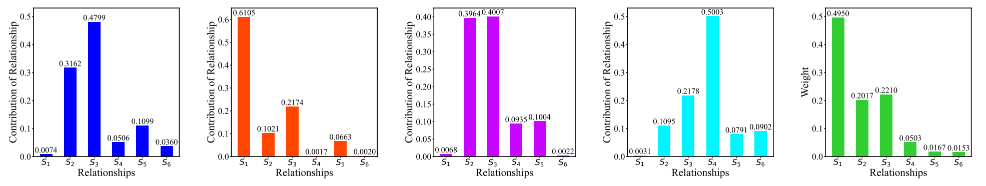
Finding-4: s_4 (hashtagging) plays a significantly important role in the social activities of social bots. Figure 9d indicates social bots are very likely to increase popularity of some specific topics by participating in the topic discussion with hashtagging. This is a unique characteristic of social bots, different from other types of users. Spammers are sometimes engaged in the popular hashtags, e.g., Twitter Trends, for boosting the marketing dissertation or sale promotion. This phenomenon is in line with the contribution of s_4 to the spammer U1661 shown in Fig. 9c.

4.5.3 Analysis of implicit relevance exploration in ICL

Figure 9e shows the weight of the different relationships in one ICL. It can be seen that the proportion of s_1 is the highest, which is also consistent with the goal of the framework. In general, zombie users increase the popularity of normal users by following them. However, zombie users often have no direct connection with each other. Therefore, in the exploration of implicit relevance, mining the indirect relationships between zombie users through common friends is one of the main objectives. s_2 and s_3, on the other hand, exemplify the role of secondary retweets and multiple replies in interpersonal relationships. It is conceivable that these two relationships are also undetectable in RSL. s_4, s_5 and s_6 are of relatively marginal importance. Thus, it can be seen that the further aggregation of different indirect relationships in ICL can fully exploit the multiple relationships between users.

5 Related work

Anomalous social user detection. Anomalous social users are referred to those anomalies or outliers in social networks. We focus on three types of abnormal users: zombie users, spammers, and social bots. Most approaches employ basic personal features for the detection of zombie users. Basic



(a) U1037. (b) U76. (c) U1661. (d) U181. (e) Weights of Meta-Structures.

Fig. 9 Figures **a**–**d** show the contributions made by six relationships, s_1–s_6 to normal user U1037, zombie user U76, spammer U1661 and social bot U181, respectively. Figure **e**shows the weight values of each of the above six relationships in ICL

Bayesian classification that combines user characteristics and machine learning methods have proven efficient for zombie users detection. Alowibdi et al. [54] leverage user profile characteristics with a Bayesian classification algorithm to detect deception. Cresci et al. [3] further discuss the fake followers created for profit, with which the customers use numerous followers to draw the world's attention. However, they are rule-based and lightweight classifiers with fewer characteristics taken into account. Several studies use user features and text features for spammer detection. Ala'M et al. [55] extract ten standard features based on the user personal characteristics and textual information and leverage four machine learning classifiers for detection. Lee et al. [56] exploit the observable profile user information and textual features with a machine learning-based algorithm to identify spammers. Stringhini et al. [9] and Chen et al. [57] propose lightweight features to simplify the detection procedure. However, spammers can easily hide user information and textual features to avoid being identified. Benevenuto et al. [58] experimentally demonstrate classifiers that incorporate textual features and metadata can deliver better results compared with classifiers that use only user features. Alom et al. [59] conduct deep learning-based algorithms on metadata and textual data. All these works demonstrate the use of textual information in spammer detection. Social bots spread specific messages by posting many tweets with similar content on social networks [60]. Other works [11, 61, 62] consider substantial meta users and features in the elaborate machine learning models particularly for the consideration of scalability and generalization; however, they usually come with non-negligible time and computational cost.

There is also a huge body of detection approaches based on deep learning techniques. Ping et al. [18] devise CNN-based model to extract joint features and fuse temporal characteristics of the tweet information for detecting social bots. However, the the computation cost is prohibitively high. We make good use of a hashtag to simplify bot detection, where a hashtag can effectively represent the joint features of tweets and contains temporal information. Wang et al. [63] leverage GCN for fraudster detection in the online app review system. Li et al. [20] design a GCN-based anti-Spam model that integrates both heterogeneous and homogeneous graph to capture the local and global context of a comment. Zhang et al. [21] leverage convolutional mechanism for learning the embeddings of each single-view attributed graph and attention mechanism to fuse different embeddings. Yang et al. [64] propose a self-supervised GNN architecture search technique that can learn the scope of multi-hop neighborhood and the number of layers. However, these works heavily exploit the profile data and textual information of social users and rarely consider the hidden behaviors among relevant users. Unlike these studies that treat social users as independent individuals, our work focuses on mining the relational features of social networks through a series of meta-structures.

Recently, Aljabri et al. [65] provided a comprehensive review of the latest machine learning techniques for bot detection and classification across major social media platforms, covering supervised, semi-supervised, and unsupervised methods, as well as discussing challenges and future research directions. Yang et al. [40] proposed a framework that uses minimal account metadata for efficient and scalable real-time analysis of Twitter's public tweet stream, employing a rich collection of labeled datasets and a strict validation system to ensure model accuracy and generalization. Arin et al. [66] proposed a novel deep learning architecture that combines three LSTM models and a fully connected layer to capture complex social media activity of humans and bots, exploring three learning schemes to train each component effectively. Ng et al. [51] proposed BotBuster, a social bot detector using a mixture of experts approach, where each expert analyzes a specific aspect of account information, such as the username, and their combined assessments estimate the probability of an account being a bot. In contrast, our GNNRI framework integrates diverse data types and advanced graph neural network techniques, providing a more holistic and accurate detection of social bots by capturing complex user interactions and high-order relationships.

Graph neural networks. Graph neural networks (GNNs) [67] are proposed to extend existing neural networks to fit graph-structured data. GNNs aim to learn a representation for any node in a graph, primarily through aggregating neighborhood information. The delivered node embedding is fed into downstream tasks, such as clustering, classification, and linking prediction. Graph convolutional network (GCN) [45] generalizes the idea of convolutional neural networks (CNNs) to the graph domain. The convolutional mechanism is particularly useful to fuse the spatio-temporal features and external feature properties, and hence significantly critical in time-series prediction such as traffic flows prediction [68–72].

Graph attention network (GAT) [46] introduces the classical attention mechanism to GNNs. Unlike GCN, which aggregates all the neighbor nodes equally, GAT identifies more important neighbor nodes by assigning different importance scores to them. There are numerous variants of GNN; however, none of them can be directly used in heterogeneous graphs. In fact, such traditional approaches based on homogeneous information graphs merely encompass one type of node and edge, making it inefficient to depict various objects manifesting in the real world. To deal with this deficiency, heterogeneous information networks (HIN) are proposed [73]. Due to its ability to express multiple relationships between diverse entities, HIN is widely used in many areas including recommendation systems [74, 75], chemistry [76, 77], social network mining [37], and intelligent transportation systems [78], etc. Since social networks are intrinsically heterogeneous, we combine Hin with GNN-based approach to aggregate different information from neighbor nodes so that different impacts of neighbor nodes can be enforced.

6 Conclusion

We present GNNRI, a novel anomalous user detection framework for social networks based on heterogeneous information network and graph representation learning. In essence, numerical embedding of user features will be fed into a classifier to identify potential anomalies. To best express and capture relationships among entities, we exploit rich semantic meta-structures including both meta-path and meta-graph. GNNRI breaks down the inherent connectivity between users into the calculation of *explicit relevance* and *implicit relevance*. We leverage relationship-based self-attention mechanism and following-up GCN-based convolutional mechanism for precisely modelling the explicit and implicit user relevance. Experimental results showcase a remarkable performance increase for both user classification and clustering when compared to the state-of-the-art baselines.

In the future, we plan to extend GNNRI to support streaming data so that real-time anomaly detection can be enforced, protecting vulnerable users from anomalous and harmful behaviors. For example, on social media platforms like Twitter, where millions of tweets are posted every minute, GNNRI will handle real-time data inputs such as tweets, retweets, likes, and user interactions. The system will use the new incoming data to update the dynamic graph and detect anomalies as they occur.

Acknowledgements Yangyang Li and Xinyue Sun contribute equally to this work and are co-first authors. This work is supported partially by the National Natural Science Foundation of China (no. 62402024 and no. 62402137), the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20242205, the Fundamental Research Funds for the Central Universities (501QYJC2023121001), and the UK EPSRC Grant (EP/T01461X/1). For any correspondence, please refer to Xinyue Sun and Renyu Yang.

Data availability The datasets generated and analyzed during this study are available from the corresponding author upon reasonable request.

References

1. Jiang M, Cui P, Beutel A, Faloutsos C, Yang S (2014) Detecting suspicious following behavior in multimillion-node social networks. In: Proceedings of the WWW, pp 305–306
2. Zhao J, Liu X, Yan Q, Li B, Shao M, Peng H (2020) Multi-attributed heterogeneous graph convolutional network for bot detection. Inf Sci 537:380–393
3. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M (2015) Fame for sale: efficient detection of fake twitter followers. Decis Support Syst 80:56–71
4. Khalil A, Hajjdiab H, Al-Qirim N (2017) Detecting fake followers in twitter: a machine learning approach. Int J Mach Learn Comput 7(6):198–202
5. Jindal N, Liu B (2007) Review spam detection. In: Proceedings of the WWW, pp 1189–1190
6. Wald R, Khoshgoftaar TM, Napolitano A, Sumner C (2013) Predicting susceptibility to social bots on Twitter. In: Proceedings of the IRI. IEEE, pp 6–13
7. Grier C, Thomas K, Paxson V, Zhang M (2010) @ spam: the underground on 140 characters or less. In: Proceedings of the CCS, pp 27–37
8. Benevenuto F, Magno G, Rodrigues T, Almeida V (2010) Detecting spammers on twitter. CEAS 6:12
9. Stringhini G, Kruegel C, Vigna G (2010) Detecting spammers on social networks. In: Proceedings of the ACSAC, pp 1–9
10. Shao C, Ciampaglia GL, Varol O, Flammini A, Menczer F (2017) The spread of fake news by social bots. arXiv preprint arXiv:1707.07592 96, 104
11. Beskow DM, Carley KM (2019) Its all in a name: detecting and labeling bots by their name. Comput Math Organ Theory 25(1):24–35
12. Gilani Z, Kochmar E, Crowcroft J (2017) Classification of twitter accounts into automated agents and human users. In: Proceedings of the ASONAM, pp 489–496
13. Varol O, Ferrara E, Davis CA, Menczer F, Flammini A (2017) Online human–bot interactions: detection, estimation, and characterization. In: Proceedings of the ICWSM. AAAI Press, pp 280–289
14. Huang S, Xie J, Dai X, Jiajun C (2020) A reinforced generation of adversarial examples for neural machine translation. In: Proceedings of the ACL, pp 3486–3497
15. Bao M, Li J, Zhang J, Peng H, Liu X (2019) Learning semantic coherence for machine generated spam text detection. In: 2019 international joint conference on neural networks (IJCNN). IEEE, pp 1–8
16. Ferrara E, Varol O, Davis C, Menczer F, Flammini A (2016) The rise of social bots. Commun ACM 59(7):96–104
17. Kudugunta S, Ferrara E (2018) Deep neural networks for bot detection. Inf Sci 467:312–322
18. Ping H, Qin S (2018) A social bots detection model based on deep learning algorithm. In: Proceedings of the ICCT, pp 1435–1439
19. Stine ZK, Khaund T, Agarwal N (2018) Measuring the information-foraging behaviors of social bots through word usage. In: Proceedings of the ASONAM. IEEE Computer Society, pp 570–571
20. Li A, Qin Z (2019) Spam review detection with graph convolutional networks. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 2703–2711
21. Zhang Y, Fan Y, Ye Y, Zhao L, Shi C (2019) Key player identification in underground forums over attributed heterogeneous information network embedding framework. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 549–558
22. Zhang C, Song D, Huang C, Swami A, Chawla NV (2019) Heterogeneous graph neural network. In: Proceedings of the KDD, pp 793–803
23. Hu Z, Dong Y, Wang K, Sun Y (2020) Heterogeneous graph transformer. In: Proceedings of the WWW, pp 2704–2710
24. Dou Y, Liu Z, Sun L, Deng Y, Peng H, Yu PS (2020) Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM international conference on information and knowledge management, pp 315–324

25. Cresci S (2020) A decade of social bot detection. Commun ACM 63(10):72–83
26. Cresci S, Di Pietro R, Petrocchi M, Spognardi A, Tesconi M (2017) The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In: WWW, pp 963–972
27. Shi C, Li Y, Zhang J, Sun Y, Philip SY (2016) A survey of heterogeneous information network analysis. IEEE Trans Knowl Data Eng 29(1):17–37
28. John GH (1995) Robust decision trees: removing outliers from databases. In: Proceedings of the KDD. AAAI Press, pp 174–179
29. Aggarwal A, Kumar S, Bhargava K, Kumaraguru P (2018) The follower count fallacy: detecting twitter users with manipulated follower count. In: Proceedings of the SAC, pp 1748–1755
30. Mehrotra A, Sarreddy M, Singh S (2016) Detection of fake twitter followers using graph centrality measures. In: Proceedings of the IC3I. IEEE, pp 499–504
31. Chu Z, Widjaja I, Wang H (2012) Detecting social spam campaigns on twitter. In: Proceedings of the ACNS. Springer, pp 455–472
32. Shao C, Ciampaglia GL, Varol O, Yang K-C, Flammini A, Menczer F (2018) The spread of low-credibility content by social bots. Nat Commun 9(1):1–9
33. Cai C, Li L, Zeng D (2017) Detecting social bots by jointly modeling deep behavior and content information. In: Proceedings of the CIKM, pp 1995–1998
34. Sun Y, Han J (2013) Mining heterogeneous information networks: a structural analysis approach. ACM SIGKDD Explor Newsl 14(2):20–28
35. Huang Z, Zheng Y, Cheng R, Sun Y, Mamoulis N, Li X (2016) Meta structure: computing relevance in large heterogeneous information networks. In: Proceedings of the SIGKDD. ACM, pp 1595–1604
36. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Proceedings of the NIPS, pp 5998–6008
37. Peng H, Li J, Gong Q, Song Y, Ning Y, Lai K, Yu PS (2019) Fine-grained event categorization with heterogeneous graph convolutional networks. In: Kraus S (ed) Proceedings of the IJCAI. AAAI Press, pp 3238–3245
38. Yang K-C, Varol O, Davis CA, Ferrara E, Flammini A, Menczer F (2019) Arming the public with artificial intelligence to counter social bots. Comput Hum Behav 1(1):48–61
39. Feng S, Wan H, Wang N, Li J, Luo M (2021) Twibot-20: a comprehensive twitter bot detection benchmark. In: CIKM, pp 4485–4494
40. Yang K-C, Varol O, Hui P-M, Menczer F (2020) Scalable and generalizable social bot detection through data selection. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 1096–1103
41. BotometerÂ® by OSoMe. https://botometer.iuni.iu.edu/#!/. Accessed 31 May 2023
42. Pal S, Mitra S (1992) Multilayer perceptron, fuzzy sets, and classification. IEEE Trans Neural Netw 3(5):683–697
43. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the SIGKDD. ACM, pp 701–710
44. Dong Y, Chawla NV, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the SIGKDD. ACM, pp 135–144
45. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the ICLR. OpenReview.net
46. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: Proceedings of the ICLR, pp 1–12
47. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: Proceedings of the WWW, pp 2022–2032
48. Fu X, Zhang J, Meng Z, King I (2020) Magnn: metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of the WWW, pp 2331–2341
49. Zhu S, Zhou C, Pan S, Zhu X, Wang B (2019) Relation structure-aware heterogeneous graph neural network. In: Proceedings of the ICDM. IEEE, pp 1534–1539
50. Chen Y, Bouazizi M, Ohtsuki T (2022) Social robot detection using roberta classifier and random forest regressor with similarity analysis. In: GLOBECOM 2022-2022 IEEE global communications conference. IEEE, pp 6433–6438
51. Ng LHX, Carley KM (2023) Botbuster: multi-platform bot detection using a mixture of experts. In: Proceedings of the international AAAI conference on web and social media, vol 17, pp 686–697
52. Estévez PA, Tesmer M, Perez CA, Zurada JM (2009) Normalized mutual information feature selection. IEEE Trans Neural Netw 20(2):189–201
53. Santos JM, Embrechts M (2009) On the use of the adjusted rand index as a metric for evaluating supervised classification. In: International conference on artificial neural networks. Springer, pp 175–184
54. Alowibdi JS, Buy UA, Philip SY, Stenneth L (2014) Detecting deception in online social networks. In: Proceedings of the ASONAM, pp 383–390 . IEEE
55. Ala'M A-Z, Alqatawna J, Faris H (2017) Spam profile detection in social networks based on public features. In: Proceedings of the ICICS. IEEE, pp 130–135
56. Lee K, Caverlee J, Webb S (2010) Uncovering social spammers: social honeypots+ machine learning. In: Proceedings of the SIGIR, pp 435–442
57. Chen C, Zhang J, Chen X, Xiang Y, Zhou W (2015) 6 million spam tweets: a large ground truth for timely twitter spam detection. In: Proceedings of the ICC. IEEE, pp 7065–7070
58. Benevenuto F, Magno G, Rodrigues T, Almeida V (2010) Detecting spammers on twitter. In: Proceedings of the CEAS, vol 6, p 12
59. Alom Z, Carminati B, Ferrari E (2020) A deep learning model for twitter spam detection. Online Soc Netw Media 18:100079
60. Wang P, Angarita R, Renna I (2018) Is this the era of misinformation yet: combining social bots and fake news to deceive the masses. In: Proceedings of the WWW, pp 1557–1561
61. Ferrara E (2017) Disinformation and social bot operations in the run up to the 2017 French presidential election. First Monday
62. Sayyadiharikandeh M, Varol O, Yang K-C, Flammini A, Menczer F (2020) Detection of novel social bots by ensembles of specialized classifiers. In: Proceedings of the CIKM, pp 2725–2732
63. Wang J (2019) Fdgars: Fraudster detection via graph convolutional networks in online app review system. In: Companion Proceedings of The 2019 World Wide Web conference, pp 310–316
64. Yang Y, Yang R, Li Y, Cui K, Yang Z, Wang Y, Xu J, Xie H (2022) Rosgas: adaptive social bot detection with reinforced self-supervised gnn architecture search. arXiv preprint arXiv:2206.06757
65. Aljabri M, Zagrouba R, Shaahid A, Alnasser F, Saleh A, Alomari DM (2023) Machine learning-based social media bot detection: a comprehensive literature review. Soc Netw Anal Min 13(1):20
66. Arin E, Kutlu M (2023) Deep learning based social bot detection on twitter. IEEE Trans Inf Forensics Secur 18:1763–1772
67. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
68. Ali A, Zhu Y, Zakarya M (2021) Exploiting dynamic spatio-temporal correlations for citywide traffic flow prediction using attention based neural networks. Inf Sci 577:852–870

69. Ali A, Zhu Y, Zakarya M (2022) Exploiting dynamic spatio-temporal graph convolutional neural networks for citywide traffic flows prediction. Neural Netw 145:233–247
70. Sun X, Ye Q, Hu H, Wang Y, Huang K, Wo T, Xu J (2023) Synthesizing realistic trajectory data with differential privacy. IEEE Trans Intell Transport Syst 24:5502–5515
71. Sun X, Ye Q, Hu H, Duan J, Xue Q, Wo T, Xu J (2023) Puts: Privacy-preserving and utility-enhancing framework for trajectory synthesization. IEEE Trans Knowl Data Eng 36:296–310
72. Sun X, Ye Q, Hu H, Duan J, Wo T, Xu J, Yang R (2024) Ldprecover: recovering frequencies from poisoning attacks against local differential privacy. In: 2020 IEEE 36th international conference on data engineering (ICDE). IEEE
73. Sun Y, Han J, Zhao P, Yin Z, Cheng H, Wu T (2009) Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In: Proceedings of the EDBT, pp 565–576
74. Zhao H, Yao Q, Li J, Song Y, Lee DL (2017) Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the SIGKDD. ACM, pp 635–644
75. Yu X, Ren X, Sun Y, Gu Q, Sturt B, Khandelwal U, Norick B, Han J (2014) Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the WSDM, pp 283–292
76. Chen H, Iyengar SK, Li J (2019) Large-scale analysis of drug combinations by integrating multiple heterogeneous information networks. In: Proceedings of the ACM-BCB, pp 67–76
77. Luo Y, Zhao X, Zhou J, Yang J, Zhang Y, Kuang W, Peng J, Chen L, Zeng J (2017) A network integration approach for drug–target interaction prediction and computational drug repositioning from heterogeneous information. Nat Commun 8(1):1–13
78. Hong H, Lin Y, Yang X, Li Z, Fu K, Wang Z, Qie X, Ye J (2020) Heteta: heterogeneous information network embedding for estimating time of arrival. In: Proceedings of the SIGKDD. ACM, pp 2444–2454