scientific reports

Check for updates

OPEN

SEGCN: a subgraph encoding based graph convolutional network model for social bot detection

Feng Liu[1,2], Zhenyu Li[2]✉, Chunfang Yang[2]✉, Daofu Gong[2], Haoyu Lu[2] & Fenlin Liu[2]

Message passing neural networks such as graph convolutional networks (GCN) can jointly consider various types of features for social bot detection. However, the expressive power of GCN is upper-bounded by the 1st-order Weisfeiler–Leman isomorphism test, which limits the detection performance for the social bots. In this paper, we propose a subgraph encoding based GCN model, SEGCN, with stronger expressive power for social bot detection. Each node representation of this model is computed as the encoding of a surrounding induced subgraph rather than encoding of immediate neighbors only. Extensive experimental results on two publicly available datasets, Twibot-20 and Twibot-22, showed that the proposed model improves the accuracy of the state-of-the-art social bot detection models by around 2.4%, 3.1%, respectively.

With the rapid development of Internet technology, cybersecurity incidents in cyberspace are frequent and have a great impact. Countries around the world regard network security situation awareness as the key to the strategic layout of cybersecurity. As an indispensable part of cyberspace, the security of online social networks has aroused widespread concern. Social bots pose a great challenge to online social network security. Social bots are social accounts that controlled by automated programs[1]. Bot manipulators use bots to perform various malicious activities in social networks (e.g., spreading rumors[1,2], polarizing online discussions[3], amplifying popularity[4,5], etc.) seriously jeopardize the security of cyberspace and cause adverse effects on society. Social bots have been found in different domains, including politics[6], health[7], and business[8,9].

As social networks become increasingly connected to people's lives, we are vulnerable to potential manipulation by bots. For example, in Mumbai, social bot spread rumors on social media that the vaccines were a plot by the government to sterilize Muslim children, which led to that only 50% of those who were expected to be vaccinated actually got the vaccine[7]. During the 2016 U.S. election bots posted numerous smear tweets against their competitors, swaying their supporters[6].

Social platforms and researchers proposed a series of social bot detection models to minimize the impact of malicious social bots, with early success. These detection methods can be grouped into two categories—account feature-based methods and graph structure-based methods.

Existing feature-based detection methods for social bot use many hand-crafted features from different categories of information, such as profiles, content, networks, properties and train machine learning models to separate the bots from benign users. However, many of the existing features are effectless when facing the manually aided created profile property and scheduled activities of social bot generated by complex stochastic algorithms. For instance, the rapid development of deep forgery techniques allows social bots to have identical profile information as normal accounts and automatically establish social relationships with other accounts, interspersing small amounts of malicious information with many neutral ones, which is very different from the traditionally considered bot behavior[10]. The study on Twitter bots[11,12] indicates that current social bots can more delicately disguise themselves as normal accounts and work in concert to achieve certain specific purposes, such as spreading rumors, posting advertisements.

To address these challenges, several studies used the interactions of accounts in social networks to construct social graphs which are then divided into cohesive subgraphs using graph mining techniques. This type of approach usually considers only leveraging the links of social bots in online social networks but misses the automated cues embedded in the text, time, and profile information. Therefore, these methods are unable to detect social bots that have successfully established enough attack edges (links) with normal users[13].

[1]School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450002, China. [2]Henan Provincial Key Laboratory of Cyberspace Situational Awareness, Zhenzhou 450001, China. ✉email: li1989zhenyu@126.com; chunfangyang@126.com

Inspired by graph neural network (GNN) models that utilize both structural and property features of nodes, some researchers used GCN, graph attention networks (GAT), and other Message passing neural networks (MPNN) models to integrate account property features and structural features for social bot detection[11,14], with promising results. However, MPNN's expressive power is upper-bounded by the 1st-order Weisfeiler–Leman (WL) isomorphism test[15]. The WL algorithm can be k-dimensional, which considers the k-tuple of the vertices when calculating the graph isomorphism problem. If only one vertex's characteristics (such as labels, properties, etc.) are considered, then it is a 1-dimensional WL (1-WL) algorithm. The 1-WL algorithm results in a unique set of features on most graphs, which means that each node on the graph has a unique role positioning. Therefore, for most irregular graph structures, the features obtained using 1-WL algorithm can be used as the basis for determining whether the graph is isomorphic, that is 1st-order Weisfeiler–Leman test. Importantly, researchers found that such method cannot capture basic graph structure features such as cycles and triangles[16,17]. Yang et al.[18] proposed that cycles or triangles are important features in social bot detection tasks.

To improve the expressive power of GCN, capture the basic structure in the graph, and improve the detection performance of the model, we design a subgraph encoding-based GCN model for social bot detection.

Motivation

Bot operators are easily aware of the property features used by the bot detection model and they tend to evade detection by avoiding these features[10,18]. Social bot detection models using purely structural features are unable to detect social bots that have successfully established sufficient attack edges (links) with ordinary users[13]. The MPNN-based social bot detection model proposed by Feng et al.[11] achieved good results in the social bot detection task, but it ignored the limitations of the expressive power of the MPNN.

In general, motivation can be summarized into the following two points:

- Basic graph structure features, such as rings and triangles, are important to detecting social bots. However, these features cannot be captured by directly using messaging neural networks in the entire social graph.
- Considering various types of features, rather than one type of features, can boost the social bot detection performance.

Contributions

To address the above problem, we propose an end-to-end social bot detection model with combined account semantic features, property features and structural features. Specifically, first, we vectorize the semantic and property information of the account and concatenate them into the initial representation vector of the nodes. Then, a random walk is used to extract a fixed-length subgraph of each node, and the final representation of the node is obtained using subgraph encoding. Finally, Softmax is used to identify machine accounts and human accounts.

- A GCN-based social bot detection model is proposed. The model detects social bots using semantic features, property features, and structural features of accounts simultaneously.
- Improve the expressive power of the GCN by using subgraph encoding to capture differences in the basic structure (e.g., cycles or triangles, etc.) between accounts.
- We analyze the impact of different types of features on model performance. Extensive experimental results show that the proposed model achieves better performance compared to the state-of-the-art models.

Related work

The earliest work on social bot detection dates back to 2010[19], honeypot traps were designed to detect social bots. Over time, the development of social bots has shown two main trends: single-account feature-based social bot detection and groups-based one. This section introduces the characteristics of these two categories of methods.

Single-account feature-based social bot detection

Early social bot detection methods were mainly based on feature engineering of account properties, using traditional classifiers for classification. The work from[20] filters social bots by analyzing Twitter account profiles. Specifically, it designed 16-dimensional features, for instance, screen name length, active days, the number of posted tweets, by analyzing account properties, tweet content, historical activity, and friend lists. Afterwards, it feeds these features into a random forest classifier to distinguish bots from humans, which is one of the foundational work on social bot detection based on individual accounts. Many follow-up studies continue to mine more features from accounts to improve the accuracy of model detection[18,21,22]. Some researchers, considering that social accounts should not be classified only as bots and non-bots due to the hijacking of human accounts in social networks, studied the differences between humans, bots and cyborgs in terms of tweets (number of tweets, time of posts) and account properties (external URL ratio, account reputation, etc.)[23]. This work laid down the idea of designing different classifiers for different types of bots. Cresci et al.[24] designed digital DNA, a string of characters that encodes the sequence of the accounts' action, to train different classifiers to detect different bots.

However, over time, bot operators gradually learned about classical bot detection features and managed to evade detection. The traces of the continuous evolution of bots can be found from[10,11,18]. In response to this trend, researchers continue to exploit individual account features. Yang et al.[18] mined 10 new features from the data, such as account clustering coefficients, two-way following ratio, and tweet similarity, to train classifiers against the evolution of bots. Beskow et al.[21] extracted differentiated account profile features (degree centrality, K-betweenness centrality, mean eigen centrality, etc.) and tweet features (mean/max mentions, number of

languages, etc.) from the collected data and used the random forest as the classifier. Subsequent researchers designed new features to combat the continuous evolution of bots and achieved good performance[25,26]. But it should be noted that the designed features are subject to the specific social platforms, which limits the generalization ability of these models.

To address the challenge of generalization ability and design generic social bot detection models, some researchers[27] designed various classifiers for bots using different datasets and combined these classifiers into an ensemble; Botometer-v3[28], a social bot detection system that incorporates 1700-dimensional features to improve generalization, boosted a series of research works on social bots detection[28–30]; Some scholars used natural language processing methods to extract semantic differences from account tweets to detect social bots. For example, the work from[31] designed a long short-term memory network (LSTM) based model to extract content features and temporal features of tweets to distinguish bots and people. Pre-training models in natural language processing are also applied in social bot detection[11,32].

The confrontation between bot detectors and operators is a never-ending race. The properties of a single account are easy to be forged and tampered. Dealing with this challenge, researchers work on group-based social bot detection methods.

Group-based social bot detection

The group-based social bot detection method utilizes the structural differences between the social graphs generated by humans and bots. The relationships that are used to build the social graph are usually friend relationships[33], following/follower[34,35], retweet/retweeted[36]. The detection mechanism is to use the homogeneity of social networks, in another word, the neighbor nodes of the bot tend to be bots, and the neighbor nodes of the human tend to be humans[34,37–39]. A label-enhanced network integrates labels with social networks and uses the defined badness score based on the random walk of nodes to distinguish bot and human[39]. Wang et al.[38] proposed paired Markov random field models to estimate the posterior probability of each user by loopy belief propagation and predict the user's label based on the posterior probability. Moreover, they proposed a framework to unify random walk and loopy belief propagation in[37,40] to address the limitation of the method[39] that it cannot utilize the label of bot and human, meanwhile, avoiding the problems of the method[38] that it is not scalable and does not guarantee convergence. The study from[41] trained a local classifier to calculate the local trust scores of nodes and edges, and then the local trust scores used for prediction are propagated through the global network structure by a weighted random walk and loopy belief propagation mechanism.

These group-based social bot detection methods largely improve the generalization of the model and avoid manual feature engineering[42,43]. However, this type of method only utilizes the link information between accounts, and its detection performance is greatly reduced when enough attack links are established between accounts[13].

With the rise of GCN[44], it has been widely used in various occasions, such as link prediction, node classification, community division. Researchers introduce GCN to detect social bots because GCN can utilize the link information between accounts as well as lots of other information. Sun et al.[45] designed a GCN with trust mechanism. First, the method starts a short random walk from a known real node, and its walk probability is the trust score of the node. Then, it uses these trust scores as edge weights, and uses graph convolution operations to aggregate features from local graph neighborhoods onto a weighted graph for classification. This work[14] proposed a GCN-based spam bot detection model which utilizes both account property features and neighborhood features. Following this direction, researchers designed a social bot detection model using the semantic features and property features of the relational graph convolutional network (RGCN[46]). First, it vectorizes the property features and semantic features of the accounts and concatenates the two types of vectors together. Then, the spliced semantic vectors are fed into a neural network model for training to detect social bots. This method achieves state-of-the-art results on homogenous graph social bot detection.

Recently, RoSGAS[47] designed an adaptive search GNN structure for social bot detection model, which gets rid of the a priori of people designing GNN structures and searches for appropriate GNN structures through reinforcement learning. RF-GNN[48] utilized the idea of integrated learning to detect social bots by combining the Random Forest algorithm and GNN. They both directly aggregate information from the direct neighbors of the account, which may fails to capture the differences in the basic structures (rings or triangles, etc.).

Proposed approach

To address the above challenges, we design a subgraph encoding-based approach for social bot detection, dividing the social network into multiple subgraphs and coding each node in the subgraphs with a GCN, which significantly differs from the existing methods. Since a node may belong to multiple subgraphs, so there are multiple representation vectors of a node, which enhances the representation of the node. Compared to GCNs where the central node features originate from the aggregation of its immediate neighbors, subgraph encoding considers both immediate and non-immediate neighbors, making it capable of capturing basic structural information such as rings and triangles, and therefore, more suitable for social bot detection. This is the difference between our approach and existing social bot detection methods. The framework of our model is shown in Fig. 1 and the implementation details of the model are specifically described in the following subsections.

Input

Social accounts contain abundant data information, and existing social bot detection methods identify bot accounts by mining the information contained in social accounts. This paper proposes to use account semantic features, property features, and structure features for learning account representation. The semantic features are extracted from the account's descriptions and tweets. The account's profile, such as account ID, screen name,

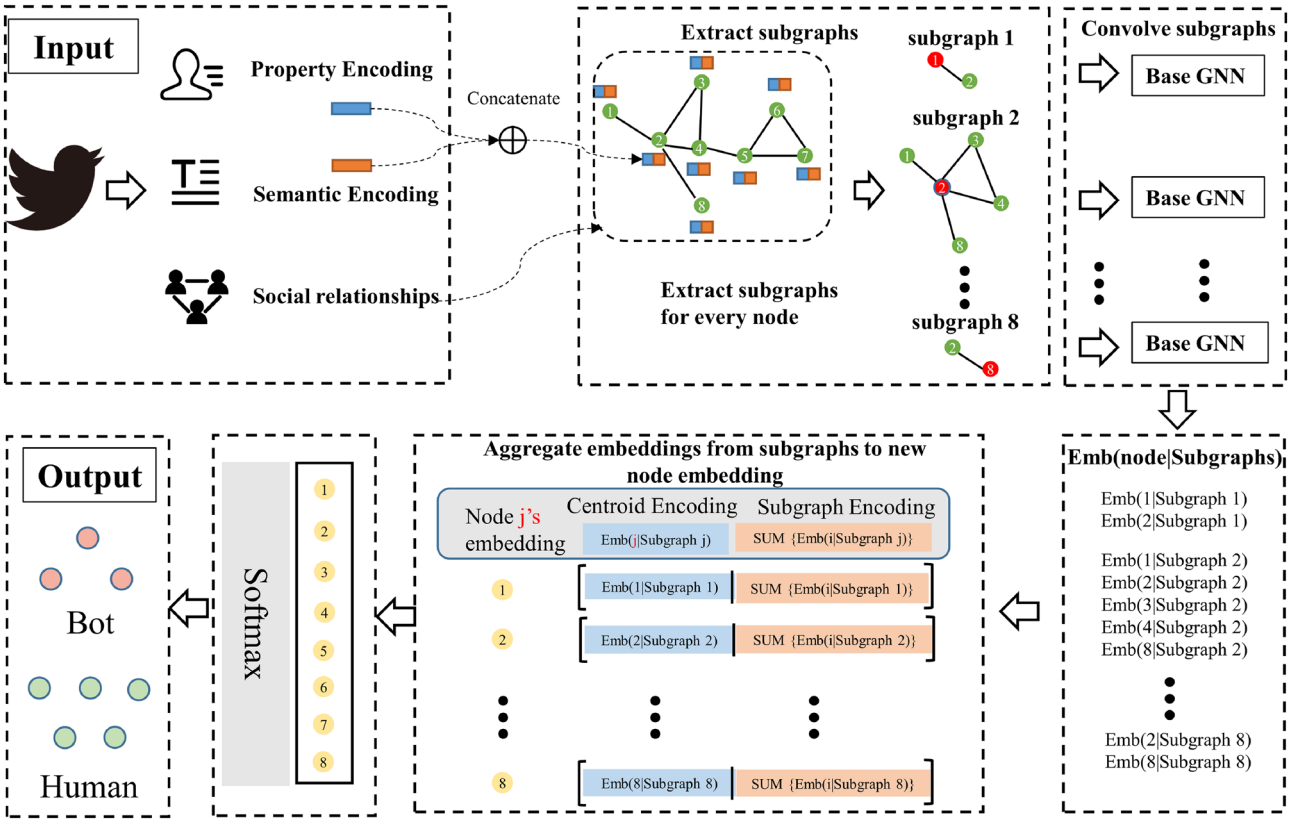


Figure 1. The framework of the proposed social bot detection model.

profile image, is the source of the property features. The social graph whose edge represents the following and follower relationship between accounts is the input for extracting the structure features.

Node representation

Learning the node (account) representation is a very important process for downstream tasks, and the node representation directly affects the model performance.

Semantic representation

Tweets can largely reflect the characteristics of the accounts and are widely used by the existing bot detection methods. We use the RoBERTa[49] language model in Transformer.pipeline to encode account semantic information. The semantic feature vector $\boldsymbol{N}_s^u$ for a given account u consists of two components: the account description semantic vector $\boldsymbol{N}_d^u$, the tweet semantic vector $\boldsymbol{N}_t^u$. The account description is a paragraph set by Twitter users to briefly introduce themselves.

First, we use RoBERTa to learn the representation vector $\boldsymbol{N}_d^u$ of the u-th account description information (see as Eq. (1)),

$$\boldsymbol{N}_d^u = \sigma(W_d \cdot \frac{1}{n}\sum_{i=1}^{n} RoBERTa(\{d_i\}_{i=1}^{n}) + b_d), \tag{1}$$

where $d_i \in \mathbb{R}^{D_R \times 1}$ and $\{d_i\}_{i=1}^{n}$ is the u-th account description that consists of n words and i represents the index of the word in the description. D_R is the embedding dimension which is predefined in RoBERTa. W_d and b_d are learnable parameters. $\boldsymbol{N}_d^u \in \mathbb{R}^{D \times 1}$, D is the dimensionof the output vector of the MLP. σ is the activation function. In this paper, Leaky-ReLU[50] is used as the activation function.

The semantic vector of account tweets can be obtained in a similar method (see as Eq. 2)

$$\boldsymbol{N}_t^u = \sigma(W_s \cdot \frac{1}{M_u}\frac{1}{m_u}\sum_{i=1}^{M_u}\sum_{j=1}^{m_u} RoBERTa(\{w_j^i\}_{j=1}^{m_u} + b_s), \tag{2}$$

where $w_j^i \in \mathbb{R}^{D_R \times 1}$ and $\{w_j^i\}_{i=1}^{m_u}$ is the i-th word of the j-th tweet, and the tweet length is m_u. W_s and b_s are learnable parameters. M_u is the number of tweets from the u-th account. $\boldsymbol{N}_t^u \in \mathbb{R}^{D \times 1}$.

Combing the two parts obtained above, we can get the semantic feature vector of the u-th account, namely, $\boldsymbol{N}_s^u = [\boldsymbol{N}_t^u, \boldsymbol{N}_d^u], \boldsymbol{N}_s^u \in \mathbb{R}^{2D \times 1}$.

Property representation

Many early social bot detection studies were successful in distinguishing bot accounts from benign accounts based on the property features of the accounts[10,18]. In this paper, the properties of accounts are divided into statistical features (e.g., number of followers, likes, retweets) and category features (e.g., whether the account is authenticated, whether it uses default profile information, whether it displays location information). All the property features used for account representation are shown in Table 1. Concerning vectorization to the property features, we use Z-Score normalization for the numerical features and One-hot encoding for the category features. The processing details can be referred to[11]

The property feature vector N_p^u for the u-th account is combined as $\boldsymbol{N}_p^u = [\boldsymbol{N}_{P_n}^u, \boldsymbol{N}_{P_c}^u], \boldsymbol{N}_p^u \in \mathbb{R}^{2D\times 1}$.

Ultimately, the initial feature representation vector of account u can be expressed as $\boldsymbol{N}_{init}^u = [\boldsymbol{N}_s^u, \boldsymbol{N}_p^u], \boldsymbol{N}_{init}^u \in \mathbb{R}^{4D\times 1}$.

Subgraph encoding

The core idea of subgraph coding is to obtain more expressive structural features of the whole graph by encoding the subgraphs extracted from the graph, which is similar to the idea of word segmentation in natural language processing. In this paper, GCN is used as encoding model for the subgraphs.

Graph nodes contain rich structural and property information. In MPNNs, each node aggregates its neighbor features in a star pattern. Therefore, MPNNs cannot distinguish the non-isomorphic regular graphs with the same star structure[51]. However, two non-isomorphic graphs with the same star structure but their subgraphs may differ (see as Fig. 2). The star structure of node "1" in Fig. 2A,B are identical, but there are differences in their subgraph structures. Subgraphs retain basic structural features such as cycles or triangles.

Feature name	Description
Followers	The number of followers an account has
Followings	The number of accounts that the account follows
Favorites	The number of favorites or likes an account receives
Statuses	The number of statuses an account posts
Active days	The number of days from the account's registration to current
Screen name length	The length of the account's current screen name
Protected	Whether the account is currently protected
Contributors enabled	Whether contributors are enabled or not
Is translator	Whether there is a translator or not
Is translation enabled	Whether the translation is available or not?
Geo enabled	Whether the account is geo enabled or not
Background tile	Whether the account uses a background tile
Background image	Whether the account uses a background image
Extended profile	Whether the account has extended profile or not
Default profile	Whether the account uses a default profile
Default profile image	Whether the account uses a default profile image
Verified	Whether the account is verified or not

Table 1. Property features used in our model.

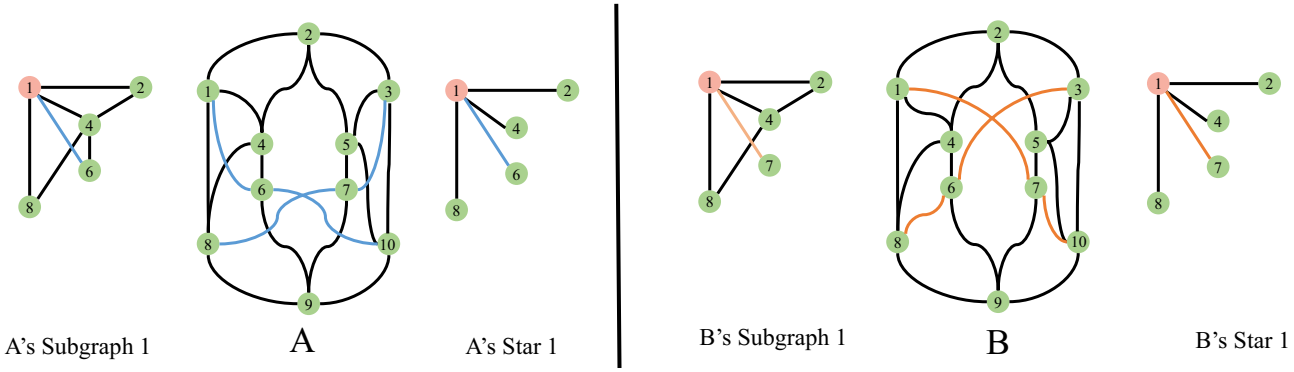


Figure 2. Illustration of the Two 4-regular graphs that cannot be distinguished by 1-WL. Colored edges are the difference between two graphs. There are differences in the first-order subgraph of some nodes in the graph.

Subgraph extraction. :n social networks, the k-hop egonet of a node as a subgraph may be too large[52]. Therefore, we use random walking to extract subgraphs that limit the subgraph size (see as Eq. (3)). In practice, We use the random walking rule in Node2vec [46].

$$G(N_{rw}(u)) = Random\ walk_{W_l}\ (u|u \in V), \tag{3}$$

where W_l is the random walking length. u is a subgraph root node. $N_{rw}(u)$ denotes the set of nodes visited by the random walker. $G(N_{rw}(u))$ denotes the subgraph whose root node is u.

Subgraph encoding. Subgraph encoding can improve the expressive power of GCN, and[51] demonstrated both theoretically and experimentally that subgraph encoding surpasses 1-WL and 2-WL and can be no weaker than 3-WL. The principle is similar to the convolution operation in convolutional neural networks.

The GCN is viewed as a kernel (GCN as kernel (GCN-AK)), and a new node representation vector is obtained by convolving it with the initial feature vector of the nodes in the subgraph. Specifically, the GCN is used as a subgraph encoder. Then, GCN-AK computes h_G by Eq. (4)

$$h_u^{l+1} = GCN^l(G^l(N_{rw}(u))), \quad l = 0, \cdots, L-1; h_G = pooling(h_v^l|u \in V), \tag{4}$$

where G is graph, $G = (V, E)$. $G(N_{rw}(u))$ is the subgraph generated by random walking from the root node u. $G^l(N_{rw}(u))$ is the subgraph with hidden features at the l-th layer. h_u^l denotes the hidden representation of node u in GNN-AK layer l. $u \in N_k(u), h_u^{(0)} = \boldsymbol{N}_{init}^u$.

We use $Subgraph^l(u)$ instead of $G^{(l)}(N_k(u))$ to denote the induced subgraph of u and use GCN to encode node i in subgraph j yields the representation vector $Emb(i|Subgraph^l(j))$. We consider the embedding of all $j \in V$ and all nodes of $i \in Subgraph^l(j)$. That the base GCN can have multiple convolutional layers, and Emb refers to the node embeddings at the last layer before global pooling $pooling_{GCN}$ that generates subgraph-level encoding.

$$h_u^{(l+1)|subgraph} = GCN^{(l)}\left(Subgraph^{(l)}(u)\right) := pooling_{GCN^{(l)}}\left(\{Emb\left(i|Subgraph^{(l)}(u)\right)|i \in N_k(u)\}\right), \tag{5}$$

We refer to the encoding of the rooted subgraph $Subgraph^{(l)}(u)$ in Eq. (5) as the subgraph encoding. Typical choices of $pooling_{GCN^{(l)}}$ are SUM and $MEAN$. As each rooted subgraph has a root node, $pooling_{GCN^{(l)}}$ can be additionally realized to differentiate the root node by self-concatenating its own representation, which is "centroid encoding", resulting in the following realization as each layer of GCN-AK:

$$h_u^{(l+1)} = FUSE(h_u^{(l+1)|centroid}, h_u^{(l+1)|subgraph}) \tag{6}$$

where $h_u^{(l+1)|centroid} := Emb(u|(Subgraph^{(l+1)}(u)$[40]. $FUSE$ is concatenation.

To improve the scalability of the model, we use the subgraph drop strategy, more details refer to. The final node representation vector is N^u.

$$\boldsymbol{N^u} = \sigma(W \cdot h_u^{(l)} + b) \tag{7}$$

where $\boldsymbol{N}^u \in \mathbb{R}^{\psi \times 1}$, and W and b are learnable parameters. ψ is the final output dimension of the model.

Output

The node representation vector $\boldsymbol{N}^u$ is obtained based on the processing in the previous subsection, and our model classifies node u as a social bot or human by the softmax layer (see as Eq. (8)).

$$\hat{y}_i = SoftMax(W \cdot \boldsymbol{N}^u + b), \tag{8}$$

where W and b are learnable parameters.

The loss function used in model training is the cross-entropy loss function which is commonly used in classification tasks. The proposed model is named as SEGCN and its pseudo-code is given as Algorithm 1.

Input: Social network: $G = (V, E)$,
Semantic information: $N_s = \{N_d, N_t\}$,
Property information: $N_p = \{N_n, N_c\}$
Output: Node representation vector: **N**

initialization **N, N$_s$, N$_p$;**
//Enconding
for $N^u \in V$ **do**
  $\boldsymbol{N}_d^u \longleftarrow RoBERTa(N_d^u)$;
  $\boldsymbol{N}_t^u \longleftarrow RoBERTa(N_t^u)$;
  $\boldsymbol{N}_{P_n}^u \longleftarrow Z\text{-}ScoreNormalization(N_{P_n}^u)$;
  $\boldsymbol{N}_{P_c}^u \longleftarrow One\text{-}hotEncoding(N_{P_c}^u)$;
  $\boldsymbol{N}_{init}^u = torch.cat((\boldsymbol{N}_d^u, \boldsymbol{N}_t^u, \boldsymbol{N}_{P_n}^u, \boldsymbol{N}_{P_c}^u))$;
end
//Subgraph extraction
while $u \in V$ **do**
  while $distance(u, u_l) < W_l$ **do**
    $Sub(u) \longleftarrow RandomWalk(u)$;
  end
  //Subgraph embedding
  while *GCN has not converged* **do**
    $Loss(\boldsymbol{N}^u) \longleftarrow GCN(\boldsymbol{N}_{init}, Sub(u))$;
  end
end
Update: $\boldsymbol{N} \longleftarrow BackPropagate(Loss(\boldsymbol{N}^u))$;
return N;

Algorithm 1. SEGCN

Experiments

In this section, we perform extensive experiments on two benchmark datasets to validate the performance of the proposed model. All experiments are conducted on a server with Intel (R) Xeon (R) Gold 6234 CPU (4×8 cores, 128 GB, 3.3 GHz) and RTX 3090 (2×24 GB) GPU running Ubuntu 20.04 (64-bit).

Datasets

The experiments are based on two different publicly available datasets, namely, the TwiBot-20 dataset[53] and the TwiBot-22 dataset[54]. The TwiBot-20 dataset is a social bot dataset made public by Feng et al.[53] in 2020, which includes 229,573 Twitter users, 33,488,192 tweets, 8,723,736 user property items and 455,958 following relationships. The TwiBot-22 dataset is a larger social bot dataset made public by Feng et al.[54] in 2022, which includes 1,000,000 Twitter users (human: 860,057, bot: 139,943), 86,764,167 tweets and 170,185,937 following relationships. An overview of the datasets is presented in Table 2.

Baseline methods

In this section, we give a brief introduction of the baseline bot detection models compared with our model.

Deepwalk[55]
Deepwalk is a graph embedding algorithm that combines random walk and word2vec, which is able to represent the nodes in a graph as a vector containing potential information. It is widely used in downstream tasks such as node classification, link prediction, and community discovery.

Datasets	Total account	Bot account	Human account
TwiBot-20[53]	229,573	5273	6589
TwiBot-22[54]	1,000,000	139,943	860,057

Table 2. Overview of the benchmark dataset.

Node2vec[56]
Node2vec is a graph embedding model that integrates node structure equivalence and neighbor similarity. Specifically, it introduces breadth-first search (BFS) and depth-first search (DFS) to capture the homogeneity and structural equivalence of nodes, and can be seen as the Deepwalk model that combines BFS and DFS random walks.

GCN[44]
GCN is a kind of MPNN. MPNN aggregates the information of neighboring nodes to update the information of central nodes, and it extends the convolution operator to the field of irregular data to realize the connection between the graph and the neural network. It has been widely used for tasks such as node classification, and link prediction.

GAT[57]
GAT follows the same message-passing paradigm, which introduces an attention mechanism that takes into account the differences in the influence of neighboring nodes on the central node. It is also widely used for downstream tasks such as link prediction, node classification and graph clustering.

Bot2vec[34]
Bot2vec is a social bot detection algorithm using only structural features proposed by Pham et al. in 2021. It is an improved version of Node2vec that introduces community detection algorithms to capture the structural equivalence of nodes.

SATAR[32]
SATAR is a self-supervised Twitter account representation model combining account semantic information, property information and neighbor information proposed by Feng et al.[32]. It achieved very good results in the task of detecting novel bots.

BotRGCN[11]
BotRGCN is an RGCN-based social bot detection model and it is similar to GCN following the message passing paradigm. Compared to GCN which aggregates on undirected graphs, it can aggregate information about surrounding neighbors in a directed graph format.

RFGNN[48]
RFGNN is a method that combines Random Forest and GNNs, which employs GNNs as the base classifiers to construct a random forest, effectively combining the advantages of ensemble learning and GNNs to improve the accuracy and robustness of the model. We use the best-performing RF-RGCN model in RF-GNN as our comparison method. Notably, this method utilizes the BERT model to extract semantic features from tweets and account descriptions.

RFGNN-R
RFGNN-R, in comparison to RFGNN, uses the RoBERTa model to extract semantic features, meaning that its method of feature extraction beyond structural features remains consistent with that of GCN, GAT, SATAR, BotRGCN, and our model.

To explicitly compare with the detection models, we present an overview of the account features used by each model in Table 3. Deepwak/Node2vc/Bot2vec exploit the structural features of accounts, and GCN, GAT, SATAR, BotRGCN and our model all exploit the semantic features, property features, and structural features of accounts. "–" is None.

Model	Published	Semantic	Property	Structure
Deepwalk	2014	–	–	✓
Node2vec	2017	–	–	✓
Bot2vec	2022	–	–	✓
GCN	2017	✓	✓	✓
GAT	2018	✓	✓	✓
SATAR	2021	✓	✓	✓
BotRGCN	2021	✓	✓	✓
RFGNN	2023	✓	✓	✓
Ours	–	✓	✓	✓

Table 3. Overview of account information used by the compared models.

Implementation details

We conducted the experiments based on the source code provided by the authors. For model-specific parameters, we used the default configuration of the code, and we tried our best to ensure that the common parameters have the same configuration. The parameter configuration of all models in the experiments is shown in Table 4. "–" is None. The source code for these baseline models can be found in the original paper as well as in TwiBot-22.

Experimental results

To validate the performance of the models, we followed the data setting approach used in baseline models such as BotRGCN, SATAR and Bot2vec. The Deepwalk/Node2vec/Bot2vec models in both public datasets are trained with 90% of the data and tested with the remaining 10% . Both GCN/GAT/SATAR/BotRGCN/RFGNN and our model use 70% of the data as the training set, 20% of the data as the validation set, and the remaining 10% of the data as the testing set. The training of neural network is stochastic to some extent, so the learned model weights and errors can vary slightly after each iteration even with the fixed hyperparameters and data splits. In order to avoid the randomness in the training process, the models are trained and tested for 5 iterations but with the same partitioned data. The average performance over the repeated experiments is reported as the final result, which smooths out the random fluctuations and provides a more stable assessment of model effectiveness. Accuracy, F1-Score and Precision are used as evaluation metrics and experiments are conducted on three benchmark datasets. The experimental results are shown in Table 5, where the best results are in bold.

As seen in Table 5, the Accuracy of the social bot detection model (Deepwalk/Node2vec/Bot2vec) using only graph structure features is below 0.65 on the Twibot-20 dataset, which may be ascribed to the following reasons. Only 20 neighbor nodes (10 Following and 10 Followers) were extracted for each account in the Twibot-20 dataset, and the structural features of the accounts were impaired. Such models use only structural features which allow novel bots to evade detection. The social bot detection models that simultaneously utilize account property features, semantic features, and structural features all have an accuracy of over 74% on the Twibot-20 dataset, which improves the detection accuracy by more than 10% than purely utilize graph structural features, indicating the desirability of combining multiple types of features for social bot detection. Compared with the GCN model, GAT and SATAR introduced attention mechanisms, and the effect was improved by more than

Parameter	Deepwalk	Node2vec	Bot2vec	GCN	GAT	SATAR	BotRGCN	RFGNN	Ours
Network layers	–	–	–	2	2	1	2	2	2
Dropout value	–	–	–	0.3	0.3	0.6	0.3	0.3	0.3
Embedding size	128	128	128	128	128	128	128	128	128
Learning rate	–	–	–	0.001	0.001	0.01	0.001	0.001	0.01
Weight decay	–	–	–	0.005	0.005	0	0.005	0.005	0.005
Optimizer	–	–	–	AdamW	AdamW	SGD	AdamW	AdamW	AdamW
Epochs	–	–	–	100	100	100	100	100	100
Window size	7	7	7	–	–	–	–	–	–
Negative sampling	5	5	5	–	–	–	–	–	–
Walk length	30	30	30	–	–	–	–	–	30
Number of walks	20	20	20	–	–	–	—	–	–
Return parameter	–	1	1	–	–	–	–	–	–
In-out parameter	–	1	1	–	–	–	–	–	–

Table 4. Overview of models' parameter configuration.

	Twitbot-20				TwiBot-22			
Model	Accuracy	F1-Score	Precision	AUC	Accuracy	F1-Score	Precision	AUC
Deepwalk	56.31 ± 1.34	61.13 ± 0.87	53.27 ± 1.26	57.71 ± 0.73	51.87 ± 1.65	37.94 ± 1.41	49.17 ± 0.76	50.28 ± 0.91
Node2vec	60.66 ± 1.03	66.05 ± 1.17	59.23 ± 0.89	61.81 ± 0.87	57.11 ± 1.25	39.27 ± 1.31	55.97 ± 0.97	55.78 ± 1.07
Bot2vec	63.28 ± 0.87	71.47 ± 1.04	63.18 ± 0.71	60.37 ± 1.37	59.14 ± 0.83	41.08 ± 1.16	57.26 ± 0.73	49.81 ± 2.37
GCN	74.64 ± 0.24	77.03 ± 0.37	73.17 ± 0.17	83.07 ± 0.48	72.39 ± 0.51	44.80 ± 0.35	71.19 ± 0.27	72.78 ± 0.42
GAT	83.27 ± 0.32	85.25 ± 0.44	81.26 ± 0.29	84.63 ± 0.56	78.36 ± 0.41	55.86 ± 0.39	72.23 ± 0.25	73.47 ± 0.33
SATAR	84.02 ± 0.17	86.07 ± 0.24	81.50 ± 0.18	90.88 ± 0.27	78.71 ± 0.42	57.10 ± 0.26	74.07 ± 0.13	79.26 ± 0.67
BotRGCN	84.61 ± 0.38	87.07 ± 0.43	83.79 ± 0.24	91.46 ± 0.26	79.66 ± 0.14	57.50 ± 0.12	74.81 ± 0.21	78.21 ± 0.56
RFGNN	83.92 ± 0.21	83.37 ± 0.27	82.19 ± 0.44	88.67 ± 0.48	78.61 ± 0.32	55.67 ± 0.41	72.86 ± 0.56	76.87 ± 0.51
RFGNN-R	85.03 ± 0.69	87.96 ± 0.57	84.11 ± 0.51	91.67 ± 0.63	80.37 ± 0.46	57.97 ± 0.58	75.33 ± 0.51	79.61 ± 0.63
Ours	87.01 ± 0.08	88.74 ± 0.13	85.83 ± 0.06	93.79 ± 0.32	82.71 ± 0.16	59.31 ± 0.12	77.23 ± 0.17	82.31 ± 0.49

Table 5. Performance comparison of multiple social bot detection models on three benchmark datasets (%).

8.6%. BotRGCN divides the edge into Following edge and Follower edge, aggregates the surrounding neighbor information according to different relationships, and the accuracy is improved by about 9.9%; Our model uses subgraph encoding to improve accuracy by about 12.4%. These phenomena indicate that changing the node aggregation method affects the performance of the model. Compared with the BotRGCN, our model's detection accuracy improves by about 2.4%, compared with the RFGNN-R, our model's detection accuracy improves by about 2.0%, indicating that the design idea of the subgraph encoding-based graph convolutional network social bot detection model is feasible.

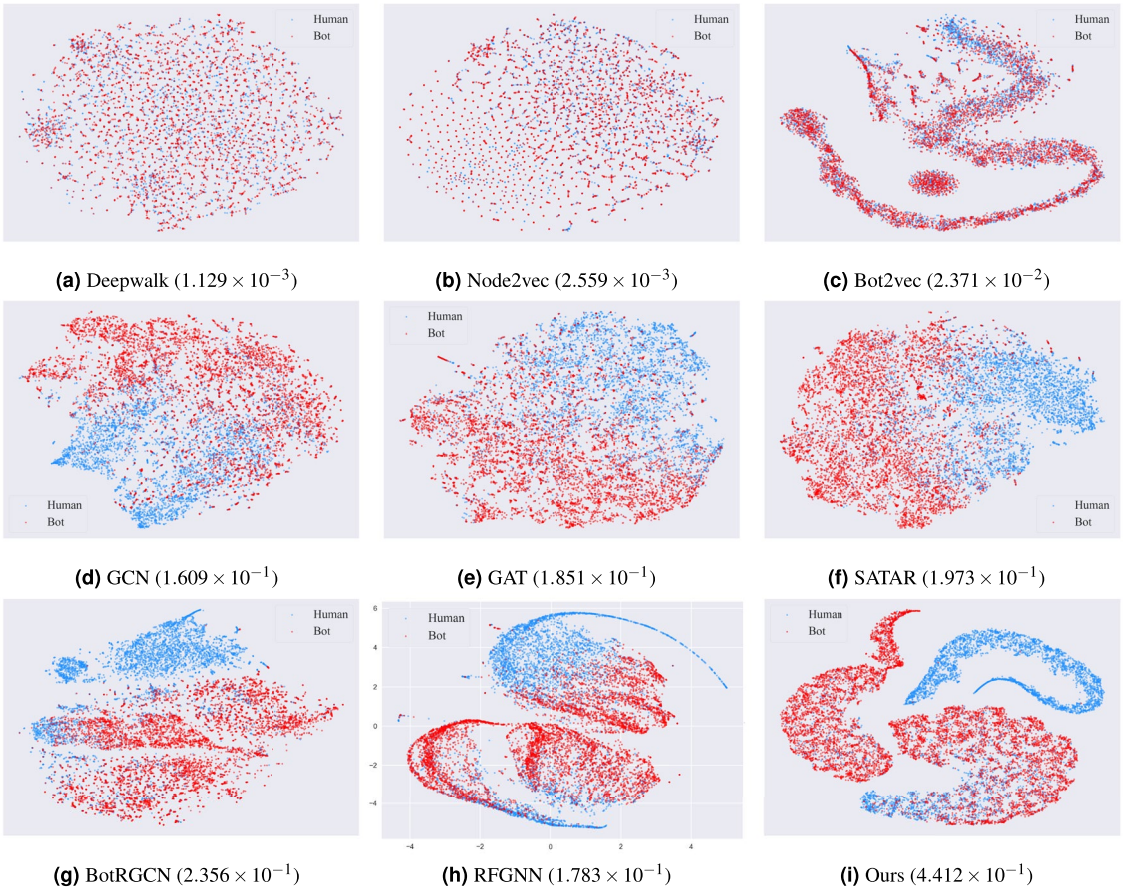
To justify why the proposed model has better performance, we use the t-SNE 2D visualization technique to visualize the embedding vectors and the corresponding homogeneity score obtained by each model on the TwiBot-20 dataset and TwiBot-22, as illustrated in Figs. 3 and 4. The t-SNE visualization results can reflect the quality of model training to a certain extent[11,12,32,34]. A higher homogeneity score means the samples are better clustered. It can be observed from Figs. 3 and 4 that our model achieves the highest homogeneity score and the embedding vector obtained from our model training is more beneficial for the social bot detection task.

In addition, we selected five representative models and plotted the ROC-AUC curves of each model on the Twibot-20 and Twibot-22 datasets based on the SVM classifier (Fig. 5). Observing the ROC-AUC curves, the one corresponding the node representation vectors learned by our model has the largest area under the curve, which indicates the proposed model has a stronger expressive power than the compared models.

Ablation experiment on features

To investigate the effect of different types of features on the detection performance of our model, we conducted feature ablation experiments on two datasets. After adding account description features (d), tweet semantic features (t), numeric features (n), and category features (c) to SEGCN, the detection accuracy of the model are shown in Fig. 6. By comparing "d", "t", "n" and "c", we can see that the category features have a greater impact on the model performance, which may be due to the fact that both datasets have more important category features such as whether they are authenticated or not. The accounts that are authenticated are usually human accounts. Most importantly, the best detection performance is achieved by "d+t+n+c", which validates that all of the four types of features are necessary for social bot detection.

In general, the use of subgraph encoding can capture the differences of structural features in subgraphs and improve the expressive power of GCN, and a large number of experiments showed the good performance of SEGCN. It should be noted that the proposed model is a general social bot detection framework, which is

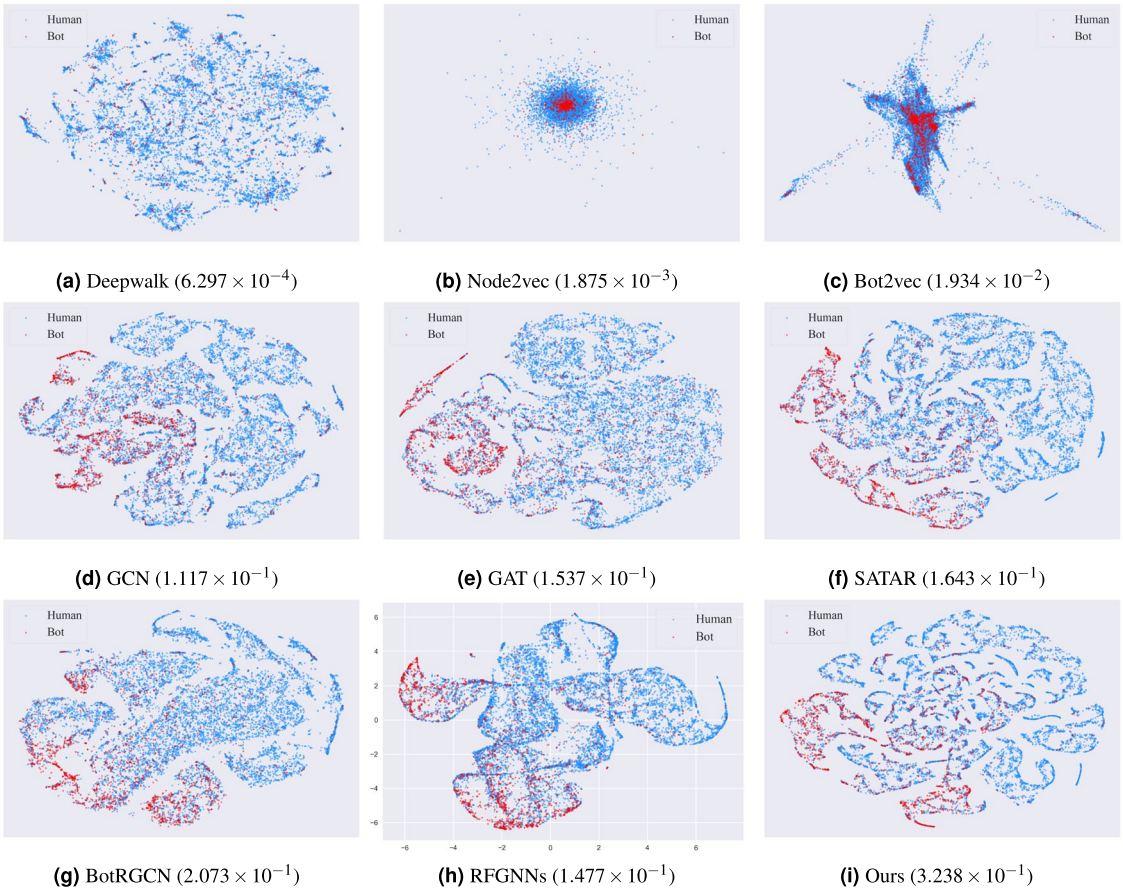


(a) Deepwalk (1.129×10^{-3}) **(b)** Node2vec (2.559×10^{-3}) **(c)** Bot2vec (2.371×10^{-2})

(d) GCN (1.609×10^{-1}) **(e)** GAT (1.851×10^{-1}) **(f)** SATAR (1.973×10^{-1})

(g) BotRGCN (2.356×10^{-1}) **(h)** RFGNN (1.783×10^{-1}) **(i)** Ours (4.412×10^{-1})

Figure 3. Visualization of human-bot user representations of the TwiBot-20 dataset by various models via t-SNE 2D projections and the corresponding homogeneity score.

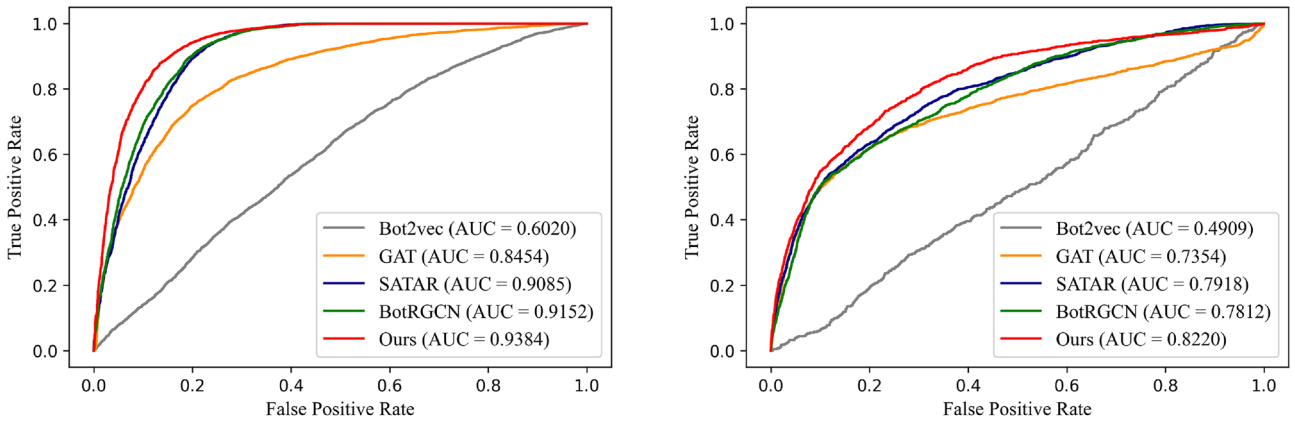


(a) Deepwalk (6.297×10^{-4}) **(b)** Node2vec (1.875×10^{-3}) **(c)** Bot2vec (1.934×10^{-2})

(d) GCN (1.117×10^{-1}) **(e)** GAT (1.537×10^{-1}) **(f)** SATAR (1.643×10^{-1})

(g) BotRGCN (2.073×10^{-1}) **(h)** RFGNNs (1.477×10^{-1}) **(i)** Ours (3.238×10^{-1})

Figure 4. Visualization of human-bot user representations of the TwiBot-22 dataset by various models via t-SNE 2D projections and the corresponding homogeneity score.



(a) ROC-AUC Curve on Twibot-20 **(b)** ROC-AUC Curve on Twibot-22

Figure 5. The ROC-AUC curve on two benchmark datasets.

applicable for furthermore meaningful features. It can also adjust the features dynamically according to the development of social bot, which has great potential for industrial applications.

Discussion

This section discusses the differences between our research and the existing ones. The investigation in[10] and extensive experimental results in "Experiments" shows that the evolution of social bots made social bot detection methods using only a single type of feature less effective in detecting novel bots. The existing social bot detection methods using multiple types of features have yielded promising results in detecting novel bot tasks, but they

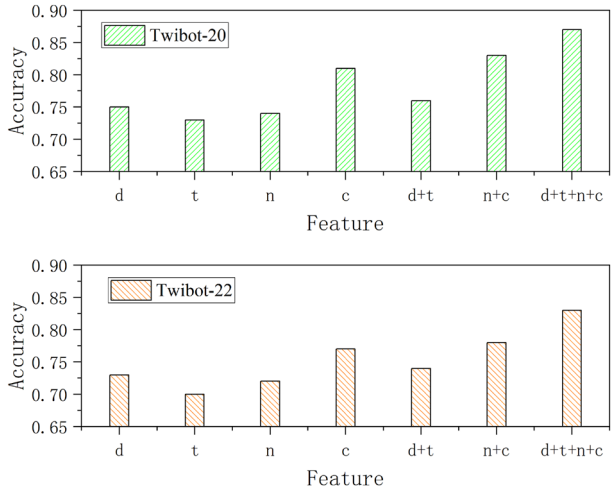


Figure 6. Illustration of accuracy when using various combination of the features for the training of the SEGCN model. The features used are accounts' description features (d), tweet feature (t), numerical features (n) and category features (c).

ignore the fact that the MPNN's expressive power is upper-bounded by the 1-WL isomorphism test[15]. The experimental results in Table 5 shows that compared with classical GCN, the subgraph coding can better capture the structural features of nodes in the social bot detection task, indicating that the subgraph encoding can improve the expression ability of GCN.

The most significant difference between our model and the existing ones is that subgraph coding method is introduced to improve the performance of social bot detection. To explicitly compare with the detection models, we present an overview of the account features used by each model in Table 3. The Deepwalk[55], Node2vec[56] and Bot2vec[34] utilize the structure features of the account. GCN[44], GAT[57], SATAR[32], BotRGCN[11], RFGNN[48] and our model all exploit the semantic features, property features, and structural features of the account. However, our model uses subgraph encoding to improve the expressiveness of the GCN.

Conclusion

In this paper, we propose a subgraph encoding based graph convolutional network model for social bot detection, named SEGCN, which uses subgraph encoding to improve the expressive power of graph convolutional networks and uses multiple types of features simultaneously for social bot detection. To the best of our knowledge, this is the first work using subgraph encoding based graph convolutional networks for social bot detection. Experimental results on two benchmark datasets show that the model achieves better performance than the SOTA approach and effectively improves the expressive power of GCN. However, the application of the proposed method in the real world social platform, for instance, Twitter (now called X), is facing more difficulty, because some of the data that needed to evaluate the social account is not free to access anymore. Nevertheless, our method provides a generalized framework for social bot detection, and social platforms and individuals can refer to this pipeline to detect the social bots. In the future, we will try to investigate the construction of heterogeneous graphs to detect social bots using accounts in social networks with multiple types of activity relationships.

Data availibility

The Twibot-20 dataset and the Twibot-22 dataset are used to support the findings of this study, which are available at "https://github.com/BunsenFeng/TwiBot-20" and "https://github.com/LuoUndergradXJTU/TwiBot-22", respectively.

Received: 22 July 2023; Accepted: 16 February 2024
Published online: 19 February 2024

References

1. Ferrara, E., Varol, O., Davis, C., Menczer, F. & Flammini, A. The rise of social bots. *Commun. ACM* **59**, 96–104 (2016).
2. Subrahmanian, V. S. *et al.* The darpa twitter bot challenge. *Computer* **49**, 38–46 (2016).
3. Stella, M., Ferrara, E. & De Domenico, M. Bots increase exposure to negative and inflammatory content in online social systems. *Proc. Natl. Acad. Sci.* **115**, 12435–12440 (2018).
4. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A. & Tesconi, M. Fame for sale: Efficient detection of fake twitter followers. *Decis. Support Syst.* **80**, 56–71 (2015).
5. Ratkiewicz, J. *et al.* Truthy: Mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th International Conference Companion on World Wide Web*, 249–252 (2011).
6. Chang, H.-C.H., Chen, E., Zhang, M., Muric, G. & Ferrara, E. Social bots and social media manipulation in 2020: The year in review. In *Handbook of Computational Social Science* Vol. 1 304–323 (Routledge, 2021).
7. Donovan, J. Stuck: How vaccine rumors start-and why they don't go away. *Nature* **583**, 680–681 (2020).

8. Cresci, S., Lillo, F., Regoli, D., Tardelli, S. & Tesconi, M. Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on twitter. *ACM Trans. Web* **13**, 1–27 (2019).
9. Noekhah, S., Binti Salim, N. & Zakaria, N. H. Opinion spam detection: Using multi-iterative graph-based model. *Inf. Process. Manage.* **57**, 102140 (2020).
10. Cresci, S. A decade of social bot detection. *Commun. ACM* **63**, 72–83 (2020).
11. Feng, S., Wan, H., Wang, N. & Luo, M. 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*, 236–239 (2021).
12. Feng, S., Wan, H., Wang, N. & Luo, M. Botrgcn: Twitter bot detection with relational graph convolutional networks. arXiv:2106.13092 (arXiv preprint) (2021).
13. Fazil, M., Sah, A. K. & Abulaish, M. Deepsbd: A deep neural network model with attention mechanism for socialbot detection. *IEEE Trans. Inf. Forensics Secur.* **16**, 4211–4223 (2021).
14. Ali Alhosseini, S., Bin Tareaf, R., Najafi, P. & Meinel, C. Detect me if you can: Spam bot detection using inductive representation learning. In *Companion Proceedings of The 2019 World Wide Web Conference*, 148–153 (2019).
15. Leskovec, K. X. W. H. J. & Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 1–9 (2018).
16. Chen, Z., Chen, L., Villar, S. & Bruna, J. Can graph neural networks count substructures?. *Adv. Neural. Inf. Process. Syst.* **33**, 10383–10395 (2020).
17. Arvind, V., Fuhlbrück, F., Köbler, J. & Verbitsky, O. On Weisfeiler–Leman invariance: Subgraph counts and related graph properties. *J. Comput. Syst. Sci.* **113**, 42–59 (2020).
18. Yang, C., Harkreader, R. & Gu, G. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Trans. Inf. Forensics Secur.* **8**, 1280–1293 (2013).
19. Yardi, S. *et al.* Detecting spam in a twitter network. *First Monday* **20**, 20 (2010).
20. Lee, K., Eoff, B. & Caverlee, J. Seven months with the devils: A long-term study of content polluters on twitter. *Proc. Int. AAAI Conf. Web Soc. Med.* **5**, 185–192 (2011).
21. Beskow, D. M. & Carley, K. M. Bot conversations are different: Leveraging network metrics for bot detection in twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 825–832 (IEEE, 2018).
22. Yang, K.-C., Varol, O., Hui, P.-M. & Menczer, F. Scalable and generalizable social bot detection through data selection. *Proc. AAAI Conf. Artif. Intell.* **34**, 1096–1103 (2020).
23. Chu, Z., Gianvecchio, S., Wang, H. & Jajodia, S. Detecting automation of twitter accounts: Are you a human, bot, or cyborg?. *IEEE Trans. Depend. Secure Comput.* **9**, 811–824 (2012).
24. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A. & Tesconi, M. Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling. *IEEE Trans. Depend. Secure Comput.* **15**, 561–576 (2017).
25. Rodríguez-Ruiz, J., Mata-Sánchez, J. I., Monroy, R., Loyola-González, O. & López-Cuevas, A. A one-class classification approach for bot detection on twitter. *Comput. Secur.* **91**, 101715 (2020).
26. De Nicola, R., Petrocchi, M. & Pratelli, M. On the efficacy of old features for the detection of new bots. *Inf. Process. Manage.* **58**, 102685 (2021).
27. Sayyadiharikandeh, M., Varol, O., Yang, K.-C., Flammini, A. & Menczer, F. Detection of novel social bots by ensembles of specialized classifiers. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 2725–2732 (2020).
28. Yang, K.-C., Ferrara, E. & Menczer, F. Botometer 101: Social bot practicum for computational social scientists. arXiv:2201.01608 (arXiv preprint) (2022).
29. Davis, C. A., Varol, O., Ferrara, E., Flammini, A. & Menczer, F. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, 273–274 (2016).
30. Miller, Z., Dickinson, B., Deitrick, W., Hu, W. & Wang, A. H. Twitter spammer detection using data stream clustering. *Inf. Sci.* **260**, 64–73 (2014).
31. Ping, H. & Qin, S. A social bots detection model based on deep learning algorithm. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, 1435–1439 (IEEE, 2018).
32. Feng, S., Wan, H., Wang, N., Li, J. & Luo, M. 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*, 3808–3817 (2021).
33. Karpov, Ilia & Glazkova, Ekaterina Detecting automatically managed accounts in online social networks: Graph embeddings approach. In *Recent Trends in Analysis of Images, Social Networks and Texts: 9th International Conference, AIST 2020, Skolkovo, Moscow, Russia, October 15–16, 2020 Revised Supplementary Proceedings* (eds van der Aalst, Wil M. P.. *et al.*) 11–21 (Springer International Publishing, 2021). https://doi.org/10.1007/978-3-030-71214-3_2.
34. Pham, P., Nguyen, L. T., Vo, B. & Yun, U. Bot2vec: A general approach of intra-community oriented representation learning for bot detection in different types of social networks. *Inf. Syst.* **103**, 101771 (2022).
35. Wang, Wenxian *et al.* Exploring the construction and infiltration strategies of social bots in sina microblog. *Sci. Rep.*.https://doi.org/10.1038/s41598-020-76814-8 (2020).
36. Zhang, J., Zhang, R., Sun, J., Zhang, Y. & Zhang, C. Truetop: A sybil-resilient system for user influence measurement on twitter. *IEEE/ACM Trans. Network.* **24**, 2834–2846 (2015).
37. Wang, B., Jia, J., Zhang, L. & Gong, N. Z. Structure-based sybil detection in social networks via local rule-based propagation. *IEEE Trans. Netw. Sci. Eng.* **6**, 523–537 (2018).
38. Wang, B., Gong, N. Z. & Fu, H. Gang: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, 465–474 (IEEE, 2017).
39. Jia, J., Wang, B. & Gong, N. Z. Random walk based fake account detection in online social networks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 273–284 (IEEE, 2017).
40. Wang, B., Zhang, L. & Gong, N. Z. Sybilscar: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, 1–9 (IEEE, 2017).
41. Gao, P. *et al.* Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *2018 IEEE Conference on Communications and Network Security (CNS)*, 1–9 (IEEE, 2018).
42. Zhao, J. *et al.* Multi-attributed heterogeneous graph convolutional network for bot detection. *Inf. Sci.* **537**, 380–393 (2020).
43. Lo, W. W., Kulatilleke, G., Sarhan, M., Layeghy, S. & Portmann, M. Xg-bot: An explainable deep graph neural network for botnet detection and forensics. *Internet Things* **22**, 100747 (2023).
44. Welling, M. & Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)* (2017).
45. Sun, Y., Yang, Z. & Dai, Y. Trustgcn: Enabling graph convolutional network for robust sybil detection in osns. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 1–7 (IEEE, 2020).
46. Schlichtkrull, M. *et al.* Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607 (Springer, 2018).
47. Yang, Y. *et al.* Rosgas: Adaptive social bot detection with reinforced self-supervised gnn architecture search. *ACM Trans. Web* **17**, 1–31 (2023).

48. Shi, S. *et al.* Rf-gnn: Random forest boosted graph neural network for social bot detection. arXiv:2304.08239 (arXiv preprint) (2023).
49. Liu, Y. *et al.* Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692 (arXiv preprint) (2019).
50. Xu, B., Wang, N., Chen, T. & Li, M. Empirical evaluation of rectified activations in convolutional network. arXiv:1505.00853 (arXiv preprint) (2015).
51. Zhao, L., Jin, W., Akoglu, L. & Shah, N. From stars to subgraphs: Uplifting any gnn with local structure awareness. arXiv:2110.03753 (arXiv preprint) (2021).
52. Kleinberg, J. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, 163–170 (2000).
53. Feng, S., Wan, H., Wang, N., Li, J. & Luo, M. Twibot-20: A comprehensive twitter bot detection benchmark. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 4485–4494 (2021).
54. Feng, S. *et al.* Twibot-22: Towards graph-based twitter bot detection. arXiv:2206.04564 (arXiv preprint) (2022).
55. Perozzi, B., Al-Rfou, R. & Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710 (2014).
56. Grover, A. & Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864 (2016).
57. Veličković, P. *et al.* Graph attention networks. In *International Conference on Learning Representations (ICLR)* (2018).

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 62002387, 61872448, 61772549, U1804263, 62002386), the Science and Technology Research Project of Henan Province (No. 222102210075), China and the Key Research and Development Project of Henan Province (No. 221111321200), China.

Author contributions

F.L. contributed the central idea, analysed most of the data, and wrote the initial draft of the paper. Z.L. provided detailed guidance on writing the manuscript and made meticulous revisions and polish to the manuscript. The remaining authors contributed to refining the ideas, carrying out additional analyses and finalizing this paper. All authors read and approved the manuscript.

Competing Interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Z.L. or C.Y.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2024