Computers, Materials & Continua CMC, vol.65, no.1, pp.617-638, 2020

Semi-GSGCN: Social Robot Detection Research with Graph Neural Network

Xiujuan Wang[1], Qianqian Zheng[1,*], Kangfeng Zheng[2], Yi Sui[1] and Jiayue Zhang[1]

Abstract: Malicious social robots are the disseminators of malicious information on social networks, which seriously affect information security and network environments. Efficient and reliable classification of social robots is crucial for detecting information manipulation in social networks. Supervised classification based on manual feature extraction has been widely used in social robot detection. However, these methods not only involve the privacy of users but also ignore hidden feature information, especially the graph feature, and the label utilization rate of semi-supervised algorithms is low. Aiming at the problems of shallow feature extraction and low label utilization rate in existing social network robot detection methods, in this paper a robot detection scheme based on weighted network topology is proposed, which introduces an improved network representation learning algorithm to extract the local structure features of the network, and combined with the graph convolution network (GCN) algorithm based on the graph filter, to obtain the global structure features of the network. An end-to-end semi-supervised combination model (Semi-GSGCN) is established to detect malicious social robots. Experiments on a social network dataset (cresci-rtbust-2019) show that the proposed method has high versatility and effectiveness in detecting social robots. In addition, this method has a stronger insight into robots in social networks than other methods.

Keywords: Social networks, social robot detection, network representation learning, graph convolution network.

1 Introduction

The great development of the Internet provides a large amount of real online user behavior data for the study of human behavior. In Q4 2019, the number of daily active twitter users reached 152 million, the number of monthly active users of Weibo reached 516 million, and the daily active users were 222 million. Gigantic userbases generate terabyte levels of data every day since they record the rich online behaviors of thousands of users. Social media has become an important part of people's life to acquire and share information [Gu, Wang and Yin (2019)]. In general, social media websites like twitter

[1] Information Technology Institute, Beijing University of Technology, Beijing, 100124, China.

[2] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[*] Corresponding Author: Qianqian Zheng. Email: zhengqianqian@emails.bjut.edu.cn.

Received: 23 April 2020; Accepted: 31 May 2020.

and Weibo bring unprecedented opportunities to study whether users' behaviors deviate from normal social patterns based on objective behavior data, so as to detect those users that damage network security.

At present, most people are willing to express their feelings, record their lives, and actively express their opinions on social mass media platforms. The entire social network has gradually become complex and diversified, and the reliability of social media content is becoming increasingly significant [Zhu, Zhao, Li et al. (2019)]. Unfortunately, social robots for various purposes (i.e., an automatic program to simulate the behavior of real normal users in social networks) have now emerged, and these robots are different from the intelligent robots that serve people in reality and have been widely studied to extend their popularization and application [Xiang, Shen, Qin et al. (2019)]. Social robots were initially created for the purpose of serving human beings and improving the quality of human life. However, the development of social robots has gradually grown out of control by human beings. Robots would pretend to be independent entities, create some false accounts, and carry out activities that cause harm to innocent users, such as stealing users' privacy, sending spam, spreading malicious links, and launching distributed denial of service (DDoS) attacks, and malicious robots as a key part of social engineering attacks have become a cancer of social networks, endangering their health [Hjouji, Hunte, Mesnards et al. (2018)]. According to a report by the U.S. Securities and Exchange Commission, more than 23 million active accounts on twitter in 2014 were actually social robots, which has become an important content production and communication power in social media. According to the 2019 bad bot report[3] released by distilling networks, the network traffic generated by robots accounted for 38% of all network traffic. Robots mentioned in the report often appear in the form of Botnet. Anonymous agents and other identity-hiding technologies are used to hide the origin of their traffic and disguise themselves as legitimate human beings. It is this feature that makes them difficult to prevent and control [Sneha and Ferrara (2018)]. Varol et al. [Varol, Ferrara, Davis et al. (2018)] point out that 9%–15% of active twitter accounts are robots, and detecting social robots is a very challenging and meaningful task.

Social robots are a kind of program that imitates human social behavior. The detection of negative users in social networks was mainly focused on the water arm, garbage users, and zombie fans in the early stage. With the emergence of robot users, the negative impact of malicious social robots was realized in all walks of life. Owing to the late emergence of social robots, the research on social robots is not relatively abundant. Several studies [Wang (2010); Zhang and Paxson (2011); Chu, Gianvecchio, Wang et al. (2012)] were an early attempt to identify malicious social robots, aimed at distinguishing robots and normal users in social networks. All of the schemes mentioned above are not totally suitable for complex social networks in different environments due to the following shortcomings: Most of the research is done to manually extract content features, behavior features, and structural features, and select features of differentiation, and then use algorithms to distinguish social robots from legitimate accounts. However, much hidden and complex information is lost by manual extraction, and the schemes employed have low versatility. Because social robots must keep in touch with other users to achieve certain communication purposes, the

[3] https://resources.distilnetworks.com/white-paper-reports/bad-bot-report-2019.

topological structure of social networks is stable over a certain period of time. Based on a simple network structure, structural features are extracted for detection without adding other complex content features and behavior features, which are more conducive to the application of the framework of social platforms, especially in different language environments. Better node representation can be obtained by comprehensively utilizing the connection information and global and local feature information about nodes. There is a lack of research on the detection of social robots by automatically learning the feature and structure information of graphs at the same time.

Therefore, in this paper social robot detection based on network topology is proposed, which mainly includes the following three aspects.

1. To improve the generality of the detection framework, graph embedding (GE) and a graph convolutional neural network (CNN) are combined in this work to detect social robots. Through the connection of social networks, GE obtains the local characteristics of graph data and learns the global structural information of graph via a graph convolution network (GCN).

2. It is considered herein that the weight coefficient between graph nodes reflects the degree of connection between nodes. The graph embedding algorithm (Graphsage), which is an inductive representation, is improved, and embedding of nodes obtained according to the relationship of node's neighbor. Therefore, to make full use of edge weight and node information, an attention mechanism is introduced and edge weight information considered to improve Graphsage to learn node representation.

3. Compared with a supervised learning algorithm, a semi-supervised learning algorithm reduces the cost of labeling, and use of a small amount of labeled data can achieve a better learning effect. To improve label utilization, GCN based on graph filtering is used to enhance the model learning of low label rate in this paper, and an end-to-end semi-supervised combination model (Semi-GSGCN) is established to detect malicious social robots.

2 Related work

2.1 Social robots

Research on social robot detection began relatively late, and the main focus of such research was based on the dynamic content sent by social robots and the social relationship graph around robots. The detection steps included pre-processing collected data and then using the content information and behavior information to select some representative and differentiated features. To achieve better classification results, most robot detection methods were based on supervised robot learning and involved manually labeled data [Ferrara, Varol, Davis et al. (2016); Alothali, Zaki, Mohamed et al. (2018)]. In earlier research by Wang [Wang (2010)], three user features and three content-based attributes in tweets based on a graph model were extracted and algorithms designed to identify robots in twitter. Zhang et al. [Zhang and Paxson (2011)] devised a method to detect the robots based on the publishing time of each tweet, and the results indicated that approximately 16% of the active accounts in twitter had higher automatic behavior. Chu et al. [Chu, Gianvecchio, Wang et al. (2012)] established a classification system from the perspective of user behavior, twitter content, and account attributes, and divided twitter users into robots, human users, and semi-robots. Dickerson et al. [Dickerson, Kagan and Subrahmanian

(2014)] studied a set of network, language, and application-oriented features that were used to distinguish humans from robots. Many emotion-related factors were the key to identify the robot especially and improved the recognition rate. Main et al. [Main and Shekokhar (2015)] selected a decision tree (DT) algorithm to train the classifier in and constructed a model from five aspects: the interval of publishing blogs, spam word detection, repeated blog detection, social score, and device of publishing blog. The training results were compared and analyzed by using two main attributes (interval of publishing blog and spam word detection), and the effect of the classifier was completely different when using five attributes. The results indicated that the interval was an important feature of robot users and exhibited a better detection effect. Bacciu et al. [Bacciu, Morgia, Eugenio et al. (2019)] detected bot and gender from two languages (English and Spanish). An integrated architecture (AdaBoost) was used to solve the bot detection problem for accounts written in English, while a single support vector machine (SVM) was used for accounts written in Spanish. The accuracy of the final model in the bot detection task exceeded 90%. Efthimion et al. [Efthimion, Payne and Proferes (2018)] proposed a new complex robot learning algorithm that used a series of features, including user name length, re-login rate, time pattern, emotional expression, the ratio of followers to friends, and the variability of messages. Logistic regression (LR) effectively detected robots with an error rate of 2.25%.

In recent years, deep learning methods have become more popular. Cai et al. [Cai, Li and Zengi (2017)] combined convolutional neural networks (CNNs) with long and short term memory (LSTM) model to explore semantic information and a potential time model. This method used content information and behavior information to transform user content into temporal text data to reduce the workload of feature determination. Sneha et al. [Sneha and Ferrara (2018)] proposed a deep neural network based on context LSTM, which used content and account metadata to detect robots at the level of tweets. This method extracted context features from user metadata as auxiliary input of a deep network to process tweet text. In addition, a technology based on combined minority oversampling (Synthetic Minority Oversampling Technology, SMOTE) was proposed to generate a large label dataset suitable for deep network training from the minimum number of label data (approximately 3,000 complex twitter robot examples), and experiments indicated that the method achieved high classification accuracy (AUC>96%) by one tweet in the process of separating robots from humans. Färber et al. [Färber, Agon and Lule (2019)] proposed a method of using CNN to identify twitter robots and selected word2vec to obtain tweet features. The performance of various embedding methods was compared in the experiment. The method proposed achieved an accuracy of 90.34% in the CLEF 2019 robot evaluation subtask (in English). Wu et al. [Wu, Liu, Dai et al. (2019)] used generated adversarial network (GAN) to expand the unbalanced dataset to improve the detection ability of social robots and finally used NN for classification. The experimental results showed that the proposed method achieved better detection performance compared with other algorithms. Stukal et al. [Stukal, Sanovich, Joshua et al. (2019)] developed a deep neural network (DNN) classifier to separate Russia's pro-government, anti-regime, and neutral twitter robots; applied the method to Russia's political twitter content; and showed that there were numerous robots on twitter.

Many research works have further promoted the practical application of a social robot detection framework. Earlier open frameworks in research of social robot detection

included those by Chavoshi et al. [Chavoshi, Hamooni and Mueen (2016); Davis, Varol, Ferrara et al. (2016)]. Botornot proposed by Davis et al. [Davis, Varol, Ferrara et al. (2016)] was the first open interface for social robot detection on twitter. The system considered six kinds of features, namely network, user, dating, time, content, and emotion, and extracted more than 1,000 kinds of attribute features for analysis to determine whether the user to be detected was a malicious social robot or a normal user. The method compared random forest (RF), AdaBoost, LR and DT, and the RF model had the best classification effect, with an accuracy rate of 95%. Yang et al. [Yang, Varol, Hui et al. (2019)] proposed a framework using least account metadata, which enabled effective analysis to be extended to real-time processing of all public tweets. To ensure the accuracy of the model, a rich set of label datasets for training and verification was constructed, and a subset of training data was elaborately selected. The results indicated that the method achieved better model precision and generalization ability than exhaustive training on all available data. Mazza et al. [Mazza, Cresci, Avvenuti et al. (2019)] proposed a robot detection technology based on forwarding behavior exploration called retweet buster (RTbust). The LSTM auto-encoder transformed the forwarding time series into the potential vectors with a compact and large amount of information, and then a hierarchical clustering algorithm was used to detect bots. An account belonging to the large cluster with malicious forwarding mode was marked as a bot, RTbust was applied to a forwarding large dataset, and two unknown active botnets with hundreds of accounts were found.

2.2 Graph deep learning

Graph data are a kind of non-Euclidean spatial data that have attracted increasingly more attention because of their universal existence. Graph data can naturally express the data structure in real life, such as network, worldwide web, and social network traffic. The local structure of each node in graph data is different from that of image and text data. The characteristic information and structure information of nodes should be considered at the same time in graph data, but the node information is extracted by hand, resulting in the loss of many hidden and complex patterns.

With the rise of deep learning, a large number of researchers began to consider introducing the model of deep learning into graph data. The representative research work includes graph embedding [Cai, Zheng and Chang (2018)], which is learning the expression of fixed length for each node by constraining the proximity of nodes, such as deepwalk [Perozzi, Rami and Skiena (2014)], line [Tang, Qu, Wang et al. (2015)], node2vec [Grover and Leskovec (2016)], and Graphsage [Hamilton, Leskovec and Ying (2017)]. Some of the representation learning algorithms only consider the local walk and local structure information of the network. The node embedding obtained by deep learning can show much hidden information, which is conducive to the improvement of the subsequent network task effect. When solving specific application problems, researchers usually divide the modeling into two stages. Taking node classification as an example, the first stage is to learn the unified length representation for each node, and the node representation is used as input to train the classification model to classify in the second stage. In recent years, researchers have paid more attention to not only graph embedding but also how to transfer the deep learning model to graph data and carry out end-to-end modeling. The graph neural network is that of the most concern in this area

[Wu, Pan, Chen et al. (2019)]. Graph embedding and graph neural network can automatically learn the feature and structure information of a graph at the same time, and graph embedding can obtain the local feature representation, while graph neural network can learn the global information feature of a graph. Both learning methods include unsupervised, semi-supervised and supervised. The graph neural network originated from the work [Scarselli, Gori, Tsoi et al. (2009)], and with the in-depth study of graph neural networks, the gate graph neural network (GAT) [Li, Tarlow, Brockschmidt et al. (2015)] and graph convolution neural network [Kipf and Welling (2017)] appeared. Graph convolution neural network is a semi-supervised learning method. There are few labeled graph data nodes in reality. Graph convolution neural network based on graph filtering [Li, Wu, Liu et al. (2019)] improved the utilization of a small number of labels. GCN has been widely used in various fields, such as citation network node classification [Kipf and Welling (2017)] and social network node prediction [Qiu, Tang, Ma et al. (2018)]. Qiu et al. [Qiu, Tang, Ma et al. (2018)] designed an end-to-end method to automatically discover hidden and predicted signals in social impact (this study focused on the prediction of user-level social impact). The influence of nodes was predicted by combining network embedding, graph convolution and graph attention mechanism into a unified framework. Based on this, it is worth studying the application of more efficient graph embedding and graph neural networks to detect large-scale social robots.

3 Social robot detection with semi-GSGCN

3.1 Problem description

The definition of social robots and social networks is introduced in this section.

Definition 1. Users in a social network can be divided into human users, normal robots, and malicious robots. Normal robots are less likely to engage in malicious behavior, their behavior is more similar to that of normal users, and they are significantly different from malicious robots. Therefore, normal robots can be defined as normal users. Detection of malicious robots is regarded as two classification problems: If a user is not a malicious robot, the user is a normal user. Supposing that the user set is $V=\{v_1,v_2,...,v_n\}$, the category set is $C=\{C_m,C_b\}$, in which C_m represents the normal user set, and C_b the malicious robot set. The problem of malicious robot identification is a classification function, as follows:

$$F(v_i,c_j)=\begin{cases}0,v_i\in C_m\\1,v_i\in C_b\end{cases},1\le i\le|V|,j\in\{m,b\} \tag{1}$$

Among them, $F(v_i,c_j)\in\{0,1\}$ is a binary function, 0 indicates that the user v_i is a normal user, and 1 indicates that the user v_i is a malicious robot.

Definition 2. A social network can be defined as a graph $G=(V,E,L)$. Then, V represents all user node sets, E is the edge relationship set between user nodes, and l is the node label set. The detailed symbol definitions involved in this social network are shown in Tab. 1.

Table 1: Symbol meanings

Symbol	Description
G	Social network graph G is composed of nodes and relationships among nodes.
V	Nodes set of G, $\|V\|$ represents number of nodes.
E	Edges set of G, $\|E\|$ represents number of edges.
L	Label set of nodes in G
X	Set of features of nodes in G
A	Adjacency matrix of G
v_i	Node i in G, $v_i \in V$
$e_i = (v_i, v_j)$	Edge i in G, $e_i \in E$
l_{v_i}	Label distribution of node v_i
x_{v_i}	Feature of node v_i
ω_{v_i, v_j}	Edge weight between nodes v_i and v_j

3.2 Detection framework

The social robot detection scheme in this paper is designed as shown in Fig. 1, comprising the following components.

(a) A social network platform, like twitter.

(b) Processing the original dataset and extracting the topological relationship.

(c) Extracting the important attributes and self-features of nodes (see Section 3.3).

(d) The input of the embedding layer, which consists of mini-batches.

(e) Graph embedding in the latter layer mines new information from the social network that contains a large amount of hidden information. In this paper, Graphsage is chosen and this method is improved, which is more suitable for a weighted forwarding relationship, and it also learns the embedding of each node in an inductive way. Each node is represented by its neighborhood aggregation (see Section 3.4 for details).

(f) The input of the GCN layer inputs the feature of a graph and features of nodes, which include (e) the embedding learned and the feature in (c).

(g) The improved GCN (see Section 3.4 for details). The number of labeled nodes in the graph is small in reality, and semi-supervised learning is more applicable, which mainly considers how to fully use a small number of labeled samples for training and classification. Compared with a supervised learning algorithm, semi-supervised learning reduces the cost of labeling, uses a small amount of labeled data, and can achieve better learning results. Therefore, an improved GCN is very suitable for the classification of graph nodes with a small amount of labeled information.

(h) The output layer of semi-supervised learning to classify graph nodes.

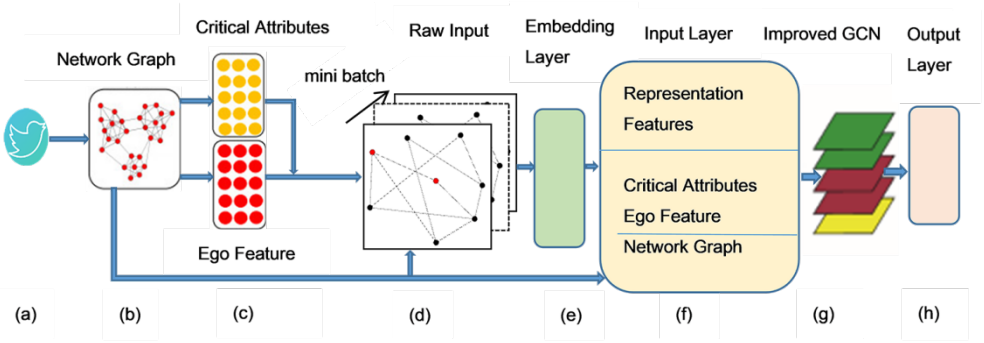


Figure 1: Social robot detection framework

3.3 Features extraction

The features extracted in this paper include self-features and structural features. The self-features are generally the user information that can be obtained directly, such as the user's basic information. In this paper, the basic information provided by the public dataset is used, including the length of the screenname, number of lower-case letters, number of upper-case letters, number of numbers, and number of special characters.

In social networks, the more the interactions between different users, the closer the relationships between them. The activities of users in social networks improve their influence to a large extent and are important information sources to describe the behavior preferences of nodes. Therefore, the structural features are very important in distinguishing the nature of users. The analysis and calculation results in the following important attributes are used to measure the importance of users based on the topological relationships.

1. Degree Centrality: The centrality of social networks can be used to measure the average influence of a node on its neighbor nodes and reflects the scale of the user's direct contact with other users. The higher the value, the greater the influence of the user. Generally, social robots will improve their degree centrality for some purpose. The calculation is

$$DC_i = \frac{\deg(i)}{n-1} \tag{2}$$

where $\deg(i)$ represents the degree of the node, and n is the total number of nodes.

2. Closeness Centrality: Closeness centrality is mainly used to calculate the indirect influence of the current node to other nodes, or the distance from the node to other nodes. The larger the value, the closer the distance between nodes, and the faster the information will be spread. The value can be considered as the closeness of the connection with other nodes in social networks. Generally, social robots will transmit information through forwarding and other behaviors to improve the compactness between itself and other nodes. The calculation is

$$CCi = (n-1) \Big/ \sum_{j=1}^{n} s_{ij} \tag{3}$$

where s_{ij} is the shortest path between nodes *i* and *j*.

3. Betweenness Centrality: This indicator is used to measure the importance of the node's position in the network structure, indicating the amount of information passing through the node. The larger the value, the greater the node's influence in the information dissemination. More nodes must pass through this node to connect with other nodes, which is similar to social elites in social networks, and the purpose of social robots is to become social elites. The centrality of social robots will be significantly greater than that of normal users. The calculation is

$$BC_i = \sum_{i \neq j \neq r} \frac{k_{jr}(i)}{k_{jr}} \tag{4}$$

where k_{jr} is the shortest path number from node *j* to node *i*, and $k_{jr}(i)$ the number of nodes passing through the shortest path.

4. Local Clustering Coefficient: Because the users in social networks are closely connected, there is a strong trend to form communities, and this trend can be measured by clustering coefficient, which indicates that there is node *i*, and that there is the possibility of any two neighbor nodes of this node to have a connection. Generally, the behaviors of normal users are mutual, while social robots are blind in finding their goals. The relationship between social robots and other users is likely to be unidirectional; for example, robots forward normal user information, while normal users rarely forward social robots. Therefore, the local clustering coefficient of social robots is smaller than that of normal users. The calculation is

$$LC_i = \frac{K}{|N_i| * (|N_{i-1}|) / 2} \tag{5}$$

where the neighbor node set of node *i* is N_i, the number of edges in the network composed of N_i is *k*, and the number of neighbor nodes of node *i* is $|N_i|$.

5. PageRank: Katz centrality calculates the influence between two nodes by walking path. The farther a node from node *i*, the smaller the impact on Katz centrality of node *i*. Once a node becomes an authoritative node (high-centrality node), it will transfer its centrality to all its external connections, resulting in a high centrality of other nodes. Therefore, the centrality of each neighbor is divided by the out degree of the neighbor node during accumulation based on Katz centrality. This measure is called PageRank. Generally, social robots strive to improve the centrality for some purpose, and the corresponding PageRank will also improve and be higher than that of a normal user. The calculation is

$$PR_i = \alpha \cdot \sum_j A_{ij} \frac{PR_j}{k_j^{out}} + \beta \tag{6}$$

where α and β are constants, k_j^{out} is the degree of node *j*, and *A* is the adjacency matrix.

3.4 Algorithm definition

3.4.1 Graphsage

Hamilton et al. [Hamilton, Leskovec and Ying (2017)] proposed that a graph sampling aggregation network (Graphsage) is a kind of inductive learning method. A graph sampling aggregation network does random sampling for neighboring nodes, so that the number of neighboring nodes of each node is less than the given number of samples. A sampling aggregation network also proposes the training model of processing data by mini-batch, which only loads the local structure of the corresponding node under each batch of input data and avoids loading the entire network, which makes it possible to apply on the large-scale social dataset. Based on this advantage, in this paper two improved algorithms are proposed, NGraphsage and AGraphsage, to make full use of the network structure to learn the node representation.

NGraphsage: This improved algorithm considers that edge weight has a different contribution to the neighbor nodes when aggregating node features. The main process is that when a node updates node features through an aggregator, the features of neighbor nodes are multiplied by the corresponding edge weight first, and then the weighted neighbor features and node features are aggregated according to the aggregation function. The algorithm is shown in Tab. 2, and $\omega_{v,u}$ represents the weight between nodes *v* and *u*.

Table 2: NGraphsage embedding generation algorithm

Input: Graph *G(V,E,L)*; input features $\left\{x_{v_i}, \forall_{v_i} \in V\right\}$; depth *K*; weight matrices W^k, $\forall_k \in \{1,...,K\}$; non-linearity σ; differentiable aggregator functions AGGREGATE_k, $\forall_k \in \{1,...,K\}$; neighborhood function $N: v \to 2^V$

Output: Vector representations z_v for all $v \in V$

$h_v^0 \leftarrow x_v, \forall_v \in V$

for $k = 1,...,K$ do

for $v \in V$ do

$h_{N(v)}^k \leftarrow AGGREGATE_k\left(\left\{\omega_{v,u} \cdot h_u^{k-1}, \forall_u \in N(v)\right\}\right)$

$h_v^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(h_v^{k-1}, h_{N(v)}^k\right)\right)$

end

$h_v^k \leftarrow h_v^k \Big/ \left\|h_v^k\right\|_2, \forall_v \in V$

end

$z_v \leftarrow h_v^k, \forall_v \in V$

AGraphsage: Different from NGraphsage, this improved algorithm introduces an attention mechanism, where the weight between nodes is an attention coefficient other

than edge weight. As shown in Tab. 3, the parameter $e_{v,u}^{k}$ is used to calculate an attention score between nodes by cosine similarity when updating iteration K, and the cosine similarity between v and u is calculated by $con\sin\left(h_v^{k-1},h_u^{k-1}\right)=\left(h_v^{k-1},h_u^{k-1}\right)\Big/\left(\left\|h_v^{k-1}\right\|\cdot\left\|h_u^{k-1}\right\|\right)$. The parameter $\alpha_{v,u}^{k}$ obtains the attention weight coefficient between nodes through attention score normalization, and the graphical representation process is shown in Fig. 2. The attention coefficient shows more similarity between nodes than edge weight.

Table 3: AGraphsage embedding generation algorithm

Input: Graph *G(V,E,L)*; input features $\left\{x_{v_i},\forall_{v_i}\in V\right\}$; depth K; weight matrices W^k, $\forall_k\in\left\{1,...,K\right\}$; non-linearity σ; differentiable aggregator functions AGGREGATE_k, $\forall_k\in\left\{1,...,K\right\}$; neighborhood function $N:v\to 2^V$

Output: Vector representations z_v for all $v\in V$

$h_v^0\leftarrow x_v,\forall_v\in V$

for $k=1,...,K$ do

for $v\in V$ do

$e_v^k=\left\{con\sin\left(h_v^{k-1},h_u^{k-1}\right),\forall_u\in N(v)\right\}$

$a_v^k=\left\{\exp\left(e_{v,u}^k\right)\Big/\sum_{u\in N(v)}\exp\left(e_{v,u}^k\right),\forall_{e_{v,u}^k}\in e_v^k\right\}$

$h_{N(v)}^k\leftarrow AGGREGATE_k\left(\left\{\alpha_{v,u}^k\cdot h_u^{k-1},\forall_u\in N(v),\forall_{\alpha_{v,u}^k}\in a_v^k\right\}\right)$

$h_v^k\leftarrow\sigma\left(W^k\cdot CONCAT\left(h_v^{k-1},h_{N(v)}^k\right)\right)$

end

$h_v^k\leftarrow h_v^k\Big/\left\|h_v^k\right\|_2,\forall_v\in V$

end

$z_v\leftarrow h_v^k,\forall_v\in V$

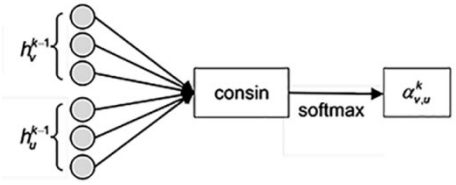


Figure 2: Graphical representation of the $\alpha_{v,u}^k$ calculation process

The two improved algorithms (NGraphsage and AGraphsage) are shown in Tabs. 2 and 3, respectively. The outer loop of these two algorithms represents the number of update iterations, while h_v^k represents the features of node v when updating iteration K, which is updated by the features of v and neighborhood of v in the previous iteration and the weight matrix W^k based on the aggregation function in each update iteration. There are four kinds of aggregation functions: mean aggregator, inductive aggregator, long short time memory (LSTM) aggregator and pooling aggregator (including meanpooling and maxpooling).

1. Mean aggregator: Each dimension in neighbor embedding is multiplied by the weight coefficient and then averaged, and then spliced with the target node embedding for nonlinear transformation. The mean aggregation calculation is

$$h_{N(v)}^k = mean\left(\left\{\omega_{v,u} or \alpha_{v,u}^k \cdot h_u^{k-1}, \forall_u \in N(v)\right\}\right) \tag{7}$$

2. Inductive aggregator: This aggregator takes the average of each dimension in the target node and all weighted neighbor embeddings directly, and then transforms it nonlinearly. Node embedding is updated according to

$$h_v^k = \sigma\left(W^k \cdot mean\left(\left\{h_v^{k-1}\right\} \cup \left\{\omega_{v,u} or \alpha_{v,u}^k \cdot h_u^{k-1}, \forall_u \in N(v)\right\}\right)\right) \tag{8}$$

3. LSTM aggregator: The LSTM function does not conform to the property of order invariants and must sort the neighbors randomly first, and then embed the random neighbor sequence into LSTM. The output is obtained as the aggregation result.

4. Pooling aggregator: First, each dimension of embedding on the forward layer of each neighbor node is multiplied by the weight coefficient, and then the nonlinear transformation is carried out (this step to equivalent to a single full connection layer, each dimension representing the representation in a certain aspect). Then, the application of meanpooling (neighborhood aggregation calculation as shown by Eq. (9) or maxpooling [neighborhood aggregation calculation as shown by Eq. (10)] is carried out according to the dimension to capture the outstanding or comprehensive performance of the neighbor set to express the embedding of the target node:

$$h_{N(v)}^k = mean\left(\left\{\sigma\left(W_{pool}\left(\omega_{v,u} or \alpha_{v,u}^k \cdot h_u^{k-1}\right), \forall_u \in N(v)\right)\right\}\right) \tag{9}$$

$$h_{N(v)}^k = \max\left(\left\{\sigma\left(W_{pool}\left(\omega_{v,u} or \alpha_{v,u}^k \cdot h_u^{k-1}\right), \forall_u \in N(v)\right)\right\}\right) \tag{10}$$

Generally, embedding is obtained with forward propagation, and back-propagation is used to update parameters, and the loss-function definitions are defined as follows, and it is expected that adjacent vertices have similar vector representations (corresponding to the first term of the equation) but are not similar to "no intersection" nodes:

$$J(z_u) = -\log\left(\sigma\left(z_u^{\mathrm{T}} z_v\right)\right) - Q \cdot E_{v_n \sim p_n(v)} \log\left(\sigma\left(-z_u^{\mathrm{T}} z_v\right)\right) \tag{11}$$

where $z_u, \forall_u \in V$ represents the embedding of any node in the graph, v represents the node that appears together with node u in a fixed length random walk, $p_n(v)$ is the probability distribution of negative sampling, and Q is the number of negative samples.

3.4.2 GCN

The graph embedding algorithm obtains the local information in the social robot network relationship, and the acquired embedding is combined with the features in Section 3.3 as the input of the next classification algorithm. In this paper, GCN [Kipf and Welling (2017)] is used for the subsequent semi-supervised classification to detect the social robot. GCN captures the global information of the graph, and the local and global features of the network structure are effectively utilized by combining with GCN and graph embedding. Then, the target output of GCN is

$$f\left(X,A\right)=\sigma\left(\hat{A}\cdot\sigma\left(\hat{A}\cdot H^{(0)}\cdot W^{(0)}\right)\cdot W^{(1)}\right)=\sigma\left(\tilde{D}^{-1/2}\cdot\tilde{A}\cdot\tilde{D}^{-1/2}\cdot\sigma\left(\tilde{D}^{-1/2}\cdot\tilde{A}\cdot\tilde{D}^{-1/2}\cdot H^{(0)}\cdot W^{(0)}\right)\cdot W^{(1)}\right) \quad (12)$$

where the objective output function is *f*, the input *X* is the feature matrix, *A* is the adjacency matrix, *W* is the training weight, and σ is the activation function, which can be Relu, Tanh, Softmax, etc. Letting $H^{(0)}=X$ in the first layer, the activation function is generally Relu, and, in the second layer, Softmax can be selected for classification. Each node adds a self-ring to obtain a new adjacency matrix, $\tilde{A}=I_N+A$, $\tilde{D}=D+I_N$, $D_{ii}=\sum_j A_{ij}$, where *D* is the diagonal nodal degree matrix of *A*. After normalization of the new adjacency matrix, one obtains $\hat{A}=\tilde{D}^{-1/2}\cdot\tilde{A}\cdot\tilde{D}^{-1/2}$.

The above is the standard GCN. To improve the utilization of a small number of labels in semi-supervised learning, the GCN [Li, Wu, Liu et al. (2019)] under graph filtering is more conducive to the detection of social robots. The objective function of the improved GCN is

$$f\left(X,A\right)=\sigma\left(\hat{A}^{(k)}\cdot\sigma\left(\hat{A}^{(k)}\cdot X\cdot W^{(0)}\right)\cdot W^{(1)}\right) \quad (13)$$

where $\hat{A}^{(k)}$ is the filter, $\tilde{L}$ the regularized Laplacian matrix $\tilde{A}$ under the new adjacency matrix, and $\tilde{L}=\Phi\tilde{\Lambda}\Phi^{-1}$, and then

$$\hat{A}=I_N-\tilde{L}=\Phi\left(I_N-\tilde{\Lambda}\right)\Phi^{-1} \quad (14)$$

Therefore, the frequency response function $\hat{A}^{(k)}=\left(I_N-\tilde{L}\right)^k$ of the filter, and the strength of the filter can be easily adjusted by using the index *k* to improve the label efficiency. Finally, the parameters can be adjusted by minimizing the loss function, which is

$$Loss=-\sum_{l\in YL}\sum_{f=1}^{F}Y_{lf}\ln Z_{lf} \quad (15)$$

where *l* is the sample number, *f* the category number, *YL* the set with a label, *y* the actual standard set, and *Z* the prediction set.

4 Experiment

4.1 Data description

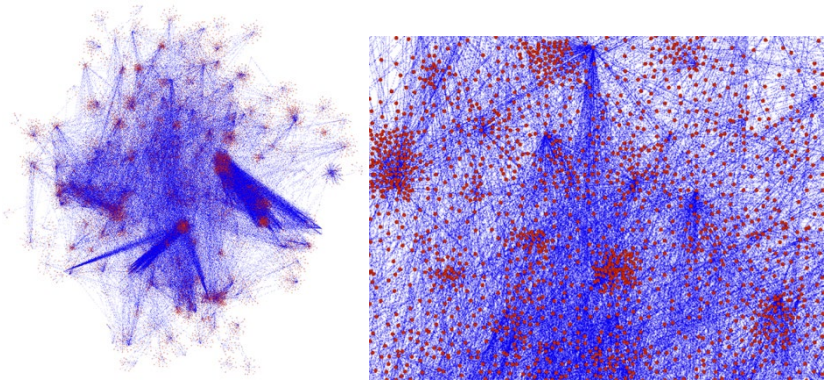
Currently, some scholars have collected several datasets and published them on the platform Bot Repository[4]. Several of the relevant data descriptions are shown in Tab. 4. Most of the datasets do not provide the topological graph of friend relationships, but one can obtain forwarding relationships between users in addition to friend and attention relationships. Many social behaviors (like concerns, comments, etc.) in social networks do not directly reflect the activities of social robots on the Internet, but forwarding can help spread information. Forwarding is the re-posting of social network information such as other people's posts or microblogs to one's own home page for one's friends to view. Because social robots must keep in touch with other users to achieve certain communication purposes, the topology of a social network is stable in a certain period of time.

For the work described in this paper, the cresci-rtbust-2019 dataset was selected to form a weighted forwarding topology in which 759 pieces of user data were selected as the center, and other nodes associated with these nodes were also selected. As shown in Fig. 3, the forwarding topology extracted in this study totaled 13,835 nodes, 759 of which are labeled. Botmeter published on Bot Repository is an open labeling website, and a python script automatically labels users through the twitter API, and botmeter will give a score within the range [0,5], as shown in Fig. 4. A higher score indicates that the user is likely a robot. Social robots are the minority of all 13,835 nodes. To maintain the positive and negative balance of samples, 2,000 nodes were labeled as robots (906): normal users (1094)=1:1.2.

Table 4: Public social robot datasets

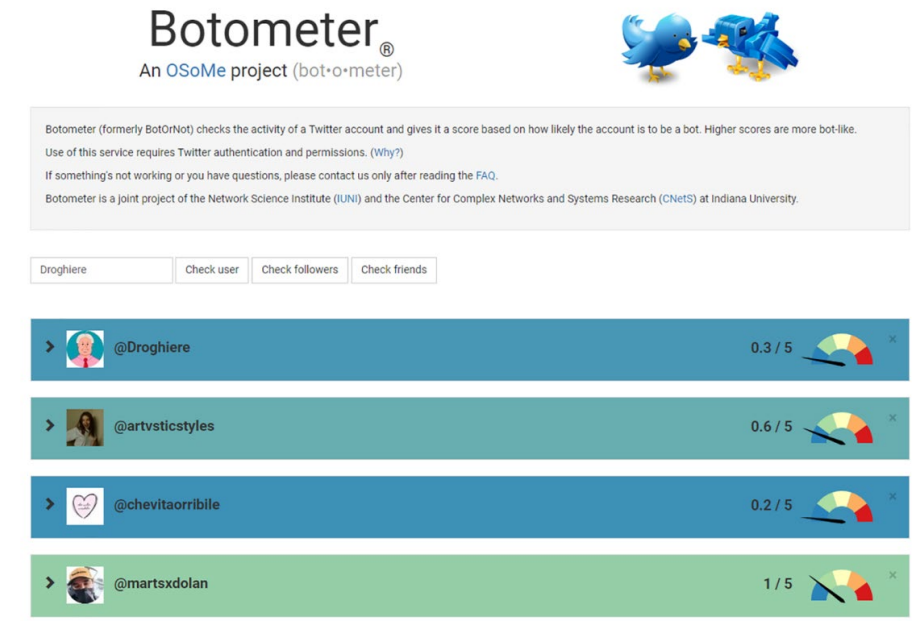
Dataset name	Bots	Human	Source
caverlee	22,179	19,276	[Lee, Eoff and Caverlee (2011)]
varol-icwsm	826	1,747	[Varol, Ferrara, Davis et al. (2018)]
cresci-17	10,894	3,474	[Cresci, Di Pietro, Petrocchi et al. (2017)]
pornbots	80,156	0	http://github.com/r0zetta/pronbot2
botometer-feedback-2019	143	386	[Yang, Varol, Davis et al. (2019)]
cresci-rtbust-2019	391	368	[Mazza, Cresci, Avvenuti et al. (2019)]

[4] https://botometer.iuni.iu.edu/.

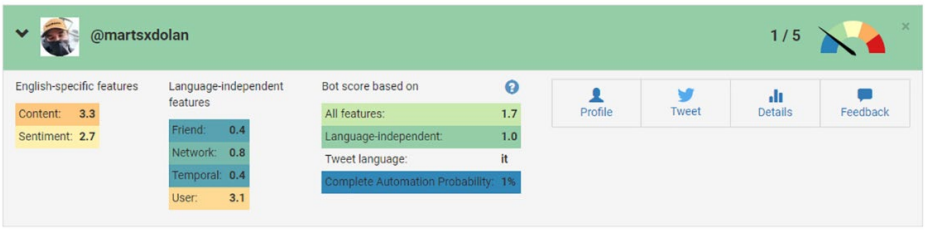


(a) (b)

Figure 3: Forwarding topology (a) Global structure (b) Local structure



(a)



(b)

Figure 4: Botometer detection social robot (a) Bot detection platform, which is detected by userscreen name or id (b) Visual result of detecting user

4.2 Experiment and results analysis

Self-features and structural features are used for the original feature input of graph embedding. To analyze the effectiveness of this method for social robot detection, the following three groups of experiments were set up. In experiments 1 and 2, the ratio of label training sets to verification set to test set was 2:1:1.

Experiment 1. This experiment verifies the influence of different feature dimensions learned from three embedding schemes on social robot detection. The experiment compares three schemes-Graphsage, NGraphsage, and AGraphsage-and the classical unsupervised embedding algorithm node2vec [Grover and Leskovec (2016)] was selected and compared with the unsupervised embedding in this paper at the end of this experiment. The embedding feature dimensions of the three schemes take values of 8, 16, 32, 128, and 256. GCN is used to test the effect of each embedding representation. In the embedding learning module of the three embedding schemes, different aggregators learn different embedding representations, as shown in Fig. 5, in which the horizontal axis represents the embedding dimension, and the vertical axis the accuracy. It can be seen in Fig. 5 that the performances of mean, LSTM, and inductive aggregators gradually increase when the dimensions change from 8 to 64, obtaining the best result when the dimension is 64 and then gradually decreasing when the dimension increases further. The accuracies of the three aforementioned aggregators under Graphsage are 66.40%, 66.80% and 66.20%, respectively; those under NGraphsage are 67.80%, 68.00%, and 67.00%, respectively; and those under AGraphsage are 68.8%, 68.2%, and 67.2%, respectively. However, maxpool and meanpool gradually increase when the dimensions change from 8 to 128 s, obtaining the best result when the dimension is 128 and then gradually decreases upon further increases. The accuracies of the two aforementioned aggregators under Graphsage are 67.80% and 66.40%, respectively; those under NGraphsage are 68.20% and 67.40%, respectively; and those under AGraphsage are 69.2% and 68.6%, respectively.

Fig. 6 illustrates the comparison of three embedding method effects under different dimensions, and it can be seen that, under different aggregators corresponding to each dimension, NGraphsage and AGraphsage obtain a better effect than Graphsage, and AGraphsage is better than Graphsage in most cases. In addition, NGraphsage and AGraphsage were compared with the classical unsupervised embedding algorithm node2vec, achieving an accuracy of 65.6%, which is worse than the improved methods NGraphsage and AGraphsage.

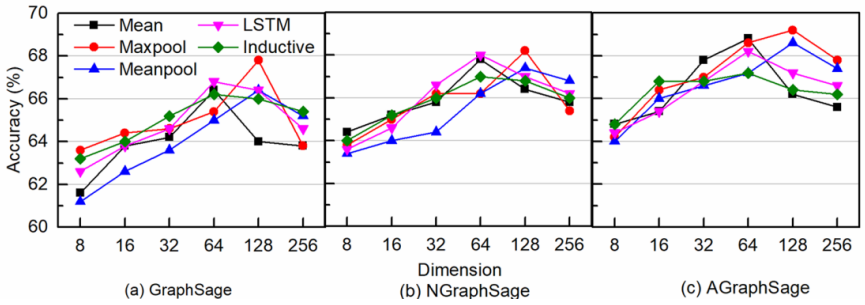


Figure 5: Influence of different dimensions on detection under five aggregators of three schemes studied (a) Graphsage (b) NGraphsage (c) AGraphsage

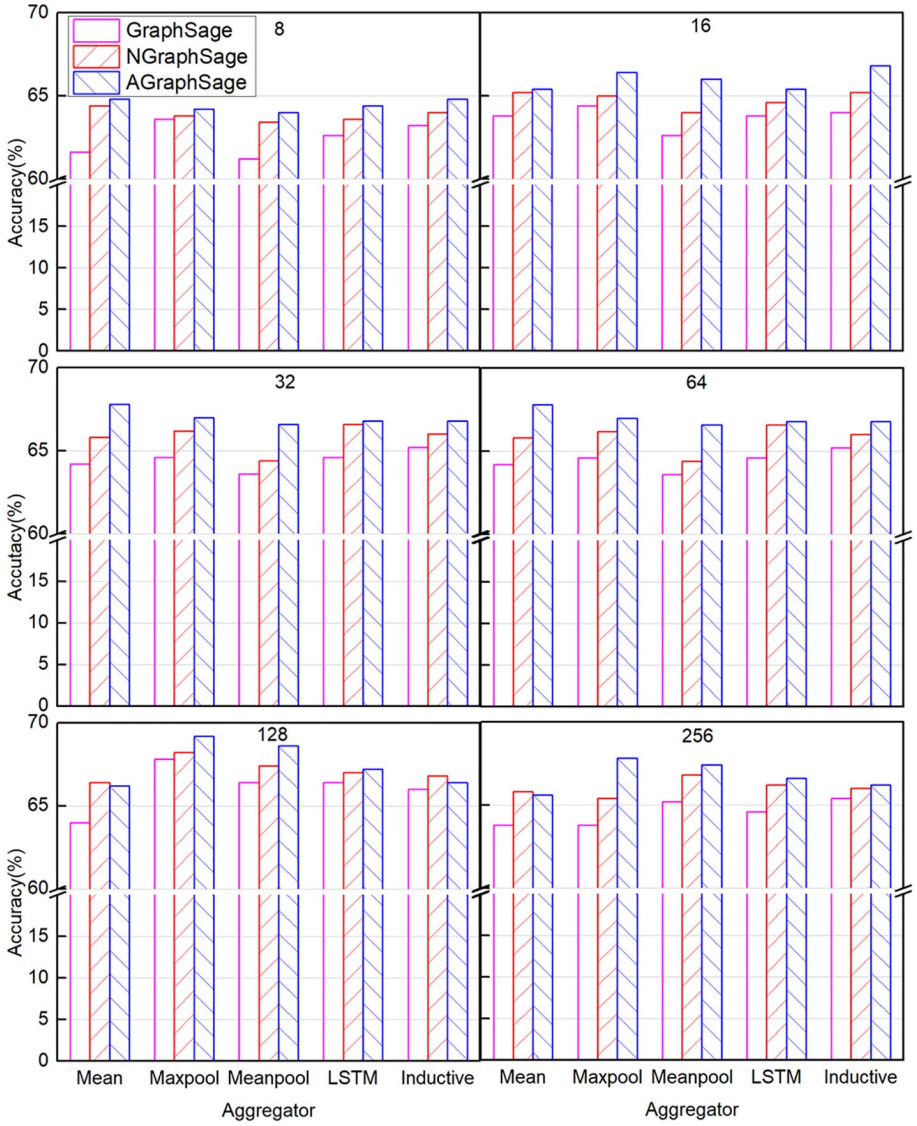


Figure 6: Comparison of effects of three schemes under different dimensions

Experiment 2. This experiment verifies the effectiveness of the method proposed in this paper by comparing different algorithms. According to experiment 1, the best detection effect is that with 128 dimensions obtained by the maxpool aggregator. In this experiment, the 128 dimensions obtained by maxpool aggregator under different embedding schemes are used as the feature input of each algorithm by combining self-features and structure features. According to the previous research, DT [Main and Shekokhar (2015)], SVM [Bacciu, Morgia, Eugenio et al. (2019)], LR [Efthimion, Payne and Proferes (2018)], MLP [Stukal, Sanovich, Joshua et al. (2019)], and RF [Davis, Varol, Ferrara et al. (2016)] were selected as comparison algorithms. The label propagation (LP) algorithm is a classical classification algorithm based on a graph and was also used as a comparison algorithm.

Since DT, SVM, LR, and RF are supervised algorithms, in this experiment all 2,000 nodes with labels were selected for model training and testing. Tab. 5 shows the detection effect of each algorithm under different graph embedding schemes. The detection effect of the framework proposed in this paper is better than that of other methods in three embedding schemes, but the detection time is higher than that of other methods, which reflects a disadvantage of the proposed method. The detection effect of each algorithm under the AGraphsage scheme is better than that of Graphsage and NGraphsage, except for RF, which performs the best in the NGraphsage scheme. The best effect of the framework proposed in this paper is 70.8%, which is better than that of the other algorithms. However, the detection effect of the framework proposed this paper is not particularly ideal. Because this study used the least original information and only a simple network structure, and the method does not extract complex content features and behavior features like other studies, it is more conducive to the application of the framework in various language and application environments. In addition, unsupervised embedding and semi-supervised learning are more practical. The final results of this paper verify the effectiveness of Semi-GSGCN and promote further research of social robot detection based on graphs.

Table 5: Detection effect of social robots using different methods (results in bold are for the proposed framework)

GE Method	Semi-supervised		Supervised (label 2,000 nodes)	
Graphsage	MLP	56.8(0.04256s)	LR	61(0.00595s)
	LP	64.6(0.00305s)	SVM	61.2(0.24459s)
	GCN	67.8(1.0322s)	DT	67.2(0.00996s)
	Semi-GSGCN	**69.4(2.60904s)**	RF	68.6(0.01001s)
NGraphsage	MLP	61 (0.03890s)	LR	62.6(0.00797s)
	LP	64.6(0.00459s)	SVM	67(0.25033s)
	GCN	68.2(1.04120s)	DT	68.2(0.00772s)
	Semi-GSGCN	**70.2(2.75511s)**	RF	70(0.01210s)
AGraphsage	MLP	61.8(0.03986s)	LR	63 (0.00698s)
	LP	64.6(0.00443s)	SVM	68(0.24740s)
	GCN	69.2(1.03624s)	DT	69(0.00699s)
	Semi-GSGCN	**70.8(2.87582s)**	RF	69.3(0.00997s)

Experiment 3. This experiment aims to analyze the influence of different label rates on the detection effect. The label training set sizes were 100, 300, 500, 700, and 1,000, and the corresponding label rates (the proportion of labeled training set data in the total dataset) are 0.72%, 2.17%, 3.60%, 5.10%, and 7.20%, respectively. The size of the verification and test sets is both 500. To compare the detection effects of GCN and Semi-GSGCN based on graph filtering under different training set sizes, the feature dimension with the best effect obtained by the AGraphsage algorithm according to experiment 2 was selected, and the self-feature and structural feature were combined as the feature input of the two methods. Fig. 7 shows the effects of the two methods, in which the horizontal axis represents the training set size, and the vertical axis shows the accuracy. According to Fig. 7, the accuracies of the GCN and Semi-GSGCN methods increase with increasing proportion of

labeled samples in the total dataset. In general, when the percentage of labeled samples is greater than 7%, the accuracies of the Semi-GSGCN and GCN methods are the highest, and the accuracy of the Semi-GSGCN method is higher than that of the GCN method. Therefore, the semi-supervised learning method proposed in this paper can form a model by training fewer samples to achieve the purpose of bot detection in a large-scale dataset. However, according to the cost in test time of the two methods shown in Fig. 8, the total test-time cost of the Semi-GSGCN method is greater than that of the GCN method under different training set sizes, and approximately 2.8 times that of the GCN method on average. In contrast, the method in this paper must be improved to reduce the test time and improve the detection speed.

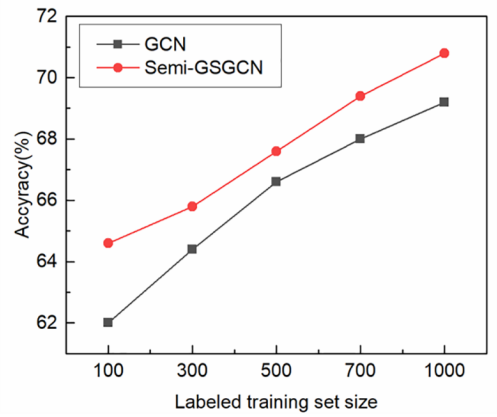


Figure 7: Effect of label training set size on the accuracy of the detection method

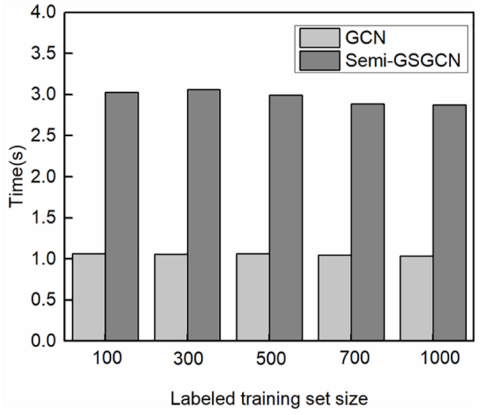


Figure 8: Efficiency of detection methods under different label training set sizes

5 Conclusions

Aiming at the problem of missing important information in the existing research of manual extraction of shallow features and low utilization rate of labels, in this paper a weighted social network detection technology is proposed. Based on the network structural information, this method extracts the local structure features of the network through the improved network representation learning algorithm, and a GCN algorithm

based on graph filtering is used to obtain the network global structural feature detection robot. In this paper, the results of experiments conducted on the public cresci-rtbust-2019 social robot dataset are reported, and then the impacts of different dimensions, algorithms, and label rates on the detection through generous experiments are analyzed. The experimental results show that the Semi-GSGCN method is an effective detection method that is superior to other algorithms, and has universal applicability and expansibility for network graph data structures.

Currently, there are few studies on the application of a graph neural network in social robot detection, and the method proposed in this paper has achieved a considerable amount of valuable results. In the future, a series of problems are planned to be explored, e.g., how to integrate more differentiated features to further improve the detection effect, how to apply the method to different language environments, and how to identify robot attributes in social robot detection. In addition to malicious social robot accounts, there are many social robot accounts that provide convenience in people's lives, and these useful robots must not be miss-detected.

Funding Statement: This research was funded by the National Key R&D Program of China [Grant Number 2017YFB0802703], Beijing Natural Science Foundation [Grant Number 4202002], the research project of the Department of Computer Science in BJUT [Grant Number 2019JSJKY004], Beijing Municipal Postdoc Science Foundation [No Grant Number] and Beijing Chaoyang District Postdoc Science Foundation [No Grant Number].

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Alothali, E.; Zaki, N.; Mohamed, E. A.; Alashwal, H. (2018): Detecting social bots on twitter: a literature review. *Proceedings of International Conference on Innovations in Information Technology*, pp. 175-180.

Bacciu, A.; Morgia, M.; Eugenio, N.; Valerio, N.; Alessandro, M. et al. (2019): Bot and gender detection of twitter accounts using distortion and LSA. *CLEF*.

Cai, C.; Li, L.; Zengi, D. (2017): Behavior enhanced deep bot detection in social media. *Proceedings of 2017 IEEE International Conference on Intelligence and Security Informatics*, pp. 128-130.

Cai, H.; Zheng, V.; Chang, C. (2018): A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616-1637.

Chavoshi, N.; Hamooni, H.; Mueen, A. (2016): DeBot: twitter bot detection via warped correlation. *Proceedings of the IEEE 16th International Conference on Data Mining*, pp. 817-822.

Chu, Z.; Gianvecchio, S.; Wang, H.; Jajodia, S. (2012): Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable & Secure Computing*, vol. 9, no. 6, pp. 811-824.

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. *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 963-972.

Davis, C.; Varol, O.; Ferrara, E.; Flammini, A.; Menczeret, F. (2016): Botornot: a system to evaluate social bots. *Proceedings of the 25th International Conference Companion on World Wide Web*, pp. 273-274.

Dickerson, J.; Kagan, V.; Subrahmanian, V. (2014): Using sentiment to detect bots on Twitter: are humans more opinionated than bots? *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 620-627.

Efthimion, P.; Payne, S.; Proferes, N. (2018): Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Science Review*, vol. 1, no. 2.

Färber, M.; Agon, Q.; Lule, A. (2019): Identifying twitter bots using a convolutional neural network. *CLEF*.

Ferrara, E.; Varol, O.; Davis, C.; Menczer, F.; Flammini, A. (2016): The rise of social bots. *Communications of the ACM*, vol. 59, no. 7, pp. 96-104.

Grover, A.; Leskovec, J. (2016): Node2vec: scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855-864.

Gu, K.; Wang, L. Y.; Yin, B. (2019): Social community detection and message propagation scheme based on personal willingness in social network. *Soft Computing*, vol. 23, no. 15, pp. 6267-6285.

Hamilton, W.; Ying, R.; Leskovec, J. (2017): Inductive representation learning on large graphs. *Proceedings of Neural Information Processing Systems*, pp. 1024-1034.

Hjouji, Z.; Hunter, D.; Mesnards, N.; Zaman, T. (2018): The impact of bots on opinions in social networks. arXiv: 1810.12398.

Kipf, T.; Welling, M. (2017): Semi-supervised classification with graph convolutional networks. *Proceedings of the 5th International Conference on Learning Representations*.

Lee, K.; Eoff, B. D.; Caverlee, J. (2011): Seven months with the devils: a long-term study of content polluters on twitter. *Proceedings of the Fifth International Conference on Weblogs and Social Media*, pp. 185-192.

Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. (2015): Gated graph sequence neural networks. *Computer Science*.

Main, W.; Shekokhar, N. (2015): Twitterati identification system. *Procedia Computer Science*, vol. 45, pp. 32-41.

Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; Tesconi, M. (2019): RTbust: exploiting temporal patterns for botnet detection on twitter. arXiv: 1902.04506.

Perozzi, B.; Rami, A.; Skiena, S. (2014): Deepwalk: online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701-710.

Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K. et al. (2018): DeepInf: social influence

prediction with deep learning. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2110-2119.

Scarselli, F.; Gori, M.; Tsoi, A.; Hagenbuchner, M.; Monfardini, G. (2009): The graph neural network model. *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61-80.

Sneha, K.; Ferrara, E. (2018): Deep neural networks for bot detection. *Information Sciences*, vol. 467, pp. 312-322.

Stukal, D.; Sanovich, S.; Joshua, A.; Bonneau, R. (2019): For whom the bot tolls: a neural networks approach to measuring political orientation of twitter bots in Russia. *Journal Indexing and Metrics*, vol. 9.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J. et al. (2015): Line: large-scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067-1077.

Varol, O.; Ferrara, E.; Davis, C.; Menczer, F.; Flammini, A. (2017): Online human-bot interactions: detection, estimation, and characterization. *Proceedings of 11th International AAAI Conference on Web and Social Media*, pp. 280-289.

Wang, A. (2010): Detecting spam bots in online social networking sites: a machine learning approach. *Data and Applications Security and Privacy XXIV*, vol. 6166, pp. 335-342.

Wu, B.; Liu, L.; Dai, Z.; Wang, X.; Zheng, K. (2019): Detecting malicious social robots with generative adversarial networks. *KSII Transactions on Internet and Information Systems*, vol. 13, no. 11, pp. 5594-5515.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C. et al. (2019): A comprehensive survey on graph neural networks. arXiv: 1901.00596v4.

Xiang, L.; Shen, X.; Qin, J.; Hao, W. (2019): Discrete multi-graph hashing for large-scale visual search. *Neural Processing Letters*, vol. 49, no. 3, pp. 1055-1069.

Yang, K.; Varol, O.; Davis, C.; Ferrara, E.; Flammini, A. et al. (2019): Arming the public with artificial intelligence to counter social bots. *Human Behavior & Emerging Technologies*, vol. 1, pp. 48-61.

Yang, K.; Varol, O.; Hui, P.; Menczeret, F. (2019): Scalable and generalizable social bot detection through data selection. arXiv: 1911.09179.

Zhang, C.; Paxson, V. (2011): Detecting and analyzing automated activity on twitter. *Proceedings of the 12th International Conference on Passive and Active Measurement*, pp. 102-111.

Zhu, C., Zhao, W., Li, Q., Li, P., Da, Q. (2019). Network embedding-based anomalous density searching for multi-group collaborative fraudsters detection in social media. *Computers, Materials & Continua*, vol. 60, no. 1, pp. 317-333.