Social Bot Detector using Graph Neural Networks

Hoang-Dung Nguyen [1,2], Duc Q. Nguyen [1,2], Hao Luong Pham [1,2], Quan Thanh Tho [1,2*]
[1] *Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT)*
[2] *Vietnam National University Ho Chi Minh City*
[*] Corresponding author: qttho@hcmut.edu.vn

***Abstract*—The importance of online social networks (OSNs) has been fueled by the human need for digital communication and broadcasting, as well as the improved state of internet connections and electronic devices. Meanwhile, social bots have been designed to automatically replicate the behavior of legitimate users in order to manipulate these OSNs. As a result, social bot detectors have been conducted concurrently, mostly on Twitter, in an attempt to discover new strategies for countering social bot attacks. In this paper, we propose SOBOG, a deep learning architecture that takes tweet relations, tweet semantics, and user properties into account to perform account-level and tweet-level detection. SOBOG also achieves outstanding performance on both tasks.**

***Index Terms*—social bot, identity deception, graph neural networks, information systems.**

I. Introduction

It was claimed in June 2016 that social bots were strategically controlling Britain's decision to leave the EU [1], which was voted on by the majority of British residents, by consistently pushing attitudes on leaving via social media posts. Later that year, in November 2016, Donald Trump was elected as the 58th President of the United States [2], igniting widespread outrage across the country. According to an Oxford University study, pro-Trump bots outnumbered pro-Clinton messages, with a bot posting every third pro-Trump tweet. It is clear that public awareness of social bots' negative influence is required, as they have become secret voting agents in such important events that could determine the fate of many countries.

Due to the negative consequences of social bots, many attempts have been made to prevent them against manipulation. One strategy is for OSN administrators to manually distinguish between bot accounts based on information provided by an account profile, which is labor-intensive. Other later approaches took advantage of machine power. The structure-based (or graph-based) approach uses a graph to represent the relationship between the accounts. However, graphs are the only consideration in this approach. By using machine learning or deep learning methodologies, the feature-based approach enables the detection of behavioral patterns based on the features of users' accounts to determine the likelihood of the accounts being bots.

Proposed feature-based methods with remarkable results typically make use of three types of data: *User properties* that includes general information such as a username, screen name, total number of followers, and statuses, *tweet semantics* captured from one or more of the contents uploaded by the account, and *user neighborhoods*, which are lists of users followed by the accounts and those who follow the account. These are employed in detectors following either *account-centric approach*, where the model labels every user as bot or human based on its information, or *tweet-centric approach*, where social bot classification performed on every tweet. Social bot detection problem raises two questions, which are is it possible to use an algorithm that can detect if both the account and the tweet are from a bot and is tweet relation effective for detecting social bots. In this paper, we attempt to find the right answers for the above questions.

To answer these questions, we introduced SOBOG (**SO**cial **BO**t Detector using **G**raph Neural Networks) which takes inputs as user properties, tweet semantics, tweet relations and returns the probabilities that the selected account is likely a bot and its tweets are bot-like. The motivation of creating SOBOG is illustrated in Figure 1. We summarize our contributions as follows: (i) We proposed a deep learning model to identify which of both accounts and tweets generated by social bots. (ii) We use a graph neural network-based model to capture the relationships between tweets, which improves confidence of claiming an account is a social bot or human. (iii) We conducted experiments on both account-level and tweet-level bot detection. Experiments on the MIB dataset show that our approach can achieve competitive performance compared to previous works [3]–[8].

The remainder of the paper is organized as follows. Section II goes over related works in social bot detection. Section III includes the description of our proposed model. Experimental results are summarized in Section IV, which compare the performance of our bot detector with that of state-of-the-art approaches, from which we analyze and give remarks in Section V. Section VI concludes the paper and outlines our future work.

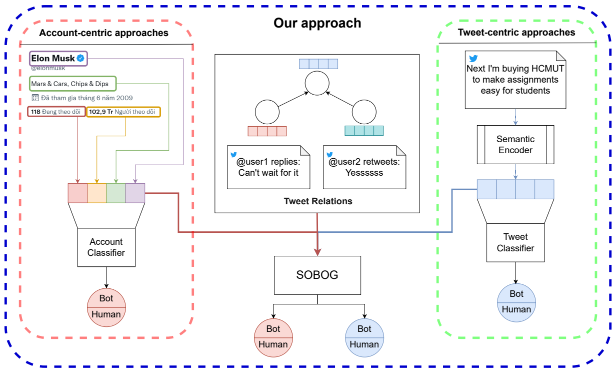
Account-centric approaches
Our approach
Tweet-centric approaches
Elon Musk
@elonmusk
Mars & Cars, Chips & Dips
Đã tham gia tháng 6 năm 2009
118 Đang theo dõi
102,9 Tr Người theo dõi
Account Classifier
Bot
Human
@user1 replies: Can't wait for it
@user2 retweets: Yessssss
Tweet Relations
SOBOG
Bot
Human
Bot
Human
Next I'm buying HCMUT to make assignments easy for students
Semantic Encoder
Tweet Classifier
Bot
Human

Fig. 1. Motivation of creating SOBOG. It collects user properties (commonly used in account-centric approach), tweet semantics (commonly used in tweet-centric approach), and tweet relations in order to perform classification on both account-level and tweet-level.

II. Related works

To address the problem of social bot detection, we explored two approaches: shallow learning and deep learning.

A. Shallow learning

Yang et al. [8] developed a scalable and generalizable social bot detection system using eight features from user properties: 6 symbolizing the increase of the previous numerical features and 6 extracted from the user's screen name, username, and description. For classification, feature vectors are fed into the classifier which employs *Random Forest* algorithm to make decisions.

Pakaya et al. [9] sought to do tweet-level detection by evaluating the difference between bot accounts and legitimate social media accounts. Every tweet was encoded into vectors, and then the classifier was tuned between Machine Learning algorithms to look for the best options. According to the results, *TF-IDF* features and the *XGBoost* algorithm produce the best accuracy on their dataset. They also used a multi-class classification to detect other variations of social bots, spam bots, and fake followers. In this case, the *Word2Vec* features and *XGBoost* performed best.

B. Deep learning

Kudugunta et Ferrara [5] introduced two distinct models for addressing account-level and tweet-level detection. On one hand, they used a mix of the *synthetic minority oversampling technique* (SMOTE) and *Edited Nearest Neighbors* (ENN) to balance the percentage of bots and legitimate accounts in the dataset, and Random Forests was the best match for account-level detection. On the other hand, *Global Vectors for Word Representation* (GLoVe) was used to embed each tweet into a sequence of vectors, and *Long Short-Term Memory* (LSTM) was used to encode those vectors into a single one. Tweet metadata was added to LSTM output before it was fed forward via fully-connected layers to get the final results. By concatenating these types of information, the model outperformed ones that did not consider tweet metadata.

Feng et al. [7] proposed *BotRGCN*, a graph convolutional neural network for bot detection. The model was given vectors of user representations that included four feature sets: user descriptions, user tweets, user numerical properties, and user categorical properties. *RoBERTa* is used to encode user descriptions and tweets in particular. The authors were interested in user networks by examining *Graph Neural Network* (GNN), which certainly influenced the choice of assessing whether an account was guilty or not. BotRGCN outperformed multiple classifiers on *TwiBot-20*, a comprehensive Twitter bot detection benchmark.

III. SOBOG model

In this chapter, we propose SOBOG, which has two main differences compared to other works:

- Our model detects social bots at both the account and tweet levels. In other words, the proposed methods are primarily concerned with determining whether the investigated account is a bot and detecting whether a tweet is "bot-featured".
- SOBOG uses tweet relations, in which two tweets build a relationship if one tweet retweets or replies to the other. We use retweets and replies as shreds of evidence since the proportion of human-human interaction was significantly higher than human-bot interaction (i.e., human is likely to retweet or reply to a retweet generated by another human rather than bots) [10].

A. Problem formulation

Let $\mathcal{X}$ be a set of accounts and $\mathcal{T}$ be a set of tweets created by account on Twitter. Given an account $x_i \in \mathcal{X}$ and a set of tweet uploaded by this account $\mathcal{T}_i \subset \mathcal{T}$, we need to determine that if account x_i is a bot or not and each tweet in $\mathcal{T}_i$ is generated by bot or not.

B. Input preprocessing

a) Feature extraction: A user object obtained from the Twitter API contains a wealth of information, many of which are attributes with personal significance. Although deep neural networks are said to extract latent features and ignore those that have no effect on the loss, having too many redundant features may consume more memory and can lead to a decrease in accuracy. From user properties, we choose 9 features and extracted 11 features as in Table I. These features have been used in a variety of related works [5], [8], and we discuss their significance in detail in the *Experiments* section. Similarly, we only consider the text and a few features that allow us to construct the Tweet Closure Graph in each tweet object.

TABLE I
DESIGNATED FEATURES FOR SOBOG

Selected features from user properties	statuses_count, followers_count, friends_count, favourites_count, listed_count, default_image, default_profile_image, protected, verified
Extracted features from user properties	tweet_frequency, followers_growth_rate, friends_growth_rate, favourites_growth_rate, listed_growth_rate, followers_friends_ratio, screen_name_length, num_digits_in_screen_name, name_length, num_digits_in_name, description_length

Selected and extracted user property features will be standardized due to the fact that each feature has a different scale. We also preprocess the tweets by lowercasing and replacing all hashtags, user mentions, and URLs into three tokens "<hashtag>", "<user>" and "<url>", respectively. Then, all the tweets are split into tokens. TF-IDF value for a token i of a tweet j is computed as:

$$w_{i,j} = tf_{i,j} * log(\frac{N}{df_i + 1}) \tag{1}$$

where $w_{i,j}$ is the TF-IDF value for token i of tweet j, $tf_{i,j}$ is the number of occurrences of token i in tweet j, df_i is the number of tweets (within all training tweets) that contain token i, and N is the number of training tweets.

After conversion, every tweet can be represented as a vector with dimension equal to the number of existing tokens in all training tweets. To reduce the dimensionality of the vector, we select only m tokens that have the highest term frequency across the corpus.

b) Tweet Closure Graph: SOBOG treats tweets as nodes. In Twitter, a tweet can be categorized into three types: *tweet*, *reply* and *retweet*. A normal tweet is usually affected by its retweets and replies. Base on that, we define a tweet closure graph of an original tweet t_i as a graph whose nodes are tweet t_i, its retweets or replies. Note that any tweet t_i is distinguished by its unique identifier. Even if a user posts a tweet and retweets that tweet, the retweet is considered a different tweet. Formally, let $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{|T| \times |T|}$ be the adjacency matrix of tweet set $\mathcal{T}$ where $a_{ij} = 1$ if tweet i is replied or retweeted by tweet j, otherwise, $a_{ij} = 0$. The definition of a tweet closure graph C_i from original tweet t_i is described as (2). Figure 2 illustrates an example of tweet closure graphs.

$$\begin{cases} C_i = \{V_{C_i}, E_{C_i}\} \\ V_{C_i} = \{t_i\} \cup \{t_j | a_{ji} > 0\} \\ E_{C_i} = \{a_{ij} | \{t_i, t_j\} \subset V_{C_i}\} \end{cases} \tag{2}$$

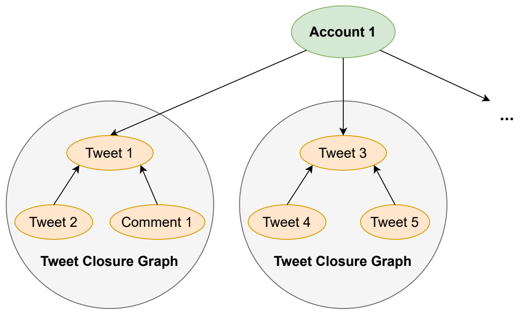

Fig. 2. An illustration for Tweet Closure Graph

c) Model input: Let $\boldsymbol{X} = \{x_i \in \mathbb{R}^n\}, i = \overline{1, |\mathcal{X}|}$ be the matrix of user representations where n is the number of user features which extracted in *Feature extraction*. Then, for each user corresponding to x_i, $\boldsymbol{T}_i = \{t_k \in \mathbb{R}^m | t_k \in C_j \wedge t_j \in \mathcal{T}_i\}$ is defined as matrix of tweet representations in all tweet closure graphs whose original tweets created by x_i; m is the dimension of tweet representation; t_k is the tweet representation vector; C_j is the Tweet Closure Graph of original tweet t_j; $\mathcal{T}_i$ is the set of tweet uploaded by account x_i. The final input is the adjacency matrix $\boldsymbol{A}^{(\boldsymbol{T}_i)}$ derived from $\boldsymbol{A}$ which indicates the relations between tweets in $\boldsymbol{T}_i$.

C. Network architecture

a) Gated Graph Attention Layers: To extract hidden information from tweet closure graph, ideally, the GNN-based layers should be considered. In our situation, beside getting to know latent features, removing irrelevant information between tweets is also taken into account. Therefore, the Gated Graph Attention layers is the most suitable among other various powerful GNN-based ones. Taking inputs as $\boldsymbol{T}_i$ and $\boldsymbol{A}^{(\boldsymbol{T}_i)}$, firstly, the tweet representation t_i is projected into embedding space as (3).

$$t_i^h = \boldsymbol{W}_h t_i, i = \overline{1, |\boldsymbol{T}_i|}, \tag{3}$$

where $\boldsymbol{W}_h \in \mathbb{R}^{d \times f}$ is a learnable weight matrix and f is the dimension of t_i. Next, the attention coefficients g_{ij} reflecting the importance of tweet t_j to t_i is computed following (4).

$$z_{ij} = \phi\left((t_i^h)^T \boldsymbol{W}_z t_j^h + (t_j^h)^T \boldsymbol{W}_z t_i^h\right); i, j = \overline{1, |\boldsymbol{T}_i|}, \tag{4a}$$

$$g_{ij} = \frac{\exp(z_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(z_{ik})} a_{ij}; i, j = \overline{1, |\boldsymbol{T}_i|}, \tag{4b}$$

where $\boldsymbol{W}_z \in \mathbb{R}^{d \times d}$ is a trainable parameter, ϕ is a non-linear activation function, $\mathcal{N}_i$ is the set of replies and retweets of t_i and $a_{ij} \in \boldsymbol{A}^{(\boldsymbol{T}_i)}$.

After that, toward each tweet t_i, all its neighbors (including itself) are weighted summed to create a feature vector t_i^{temp} (Equation 5a). Instead of consider new feature vectors as output vectors, our model is designed to control how much new information is taken to learn. To do that, the gating coefficient o_i is calculated by both old and new feature vectors, then, the output vector $\hat{t}_i$ is an element-wise sum of those feature vectors weighted by o_i. Equation 5b and 5c show the formulas for computing o_i and $\hat{t}_i$.

$$t_i^{temp} = \varphi(\sum_{i\in\mathcal{N}_i} g_{ij} t_j^h); i = \overline{1, |\boldsymbol{T}_i|} \tag{5a}$$

$$o_i = \sigma(\boldsymbol{W}_o(t_i \| t_i^{temp}) + b), i = \overline{1, |\boldsymbol{T}_i|} \tag{5b}$$

$$\hat{t}_i = (1 - o_i)t_i + o_i t_i^{temp}, i = \overline{1, |\boldsymbol{T}_i|} \tag{5c}$$

In (5), $\boldsymbol{W}_o \in \mathbb{R}^{1\times 2d}$ is learnable weight matrix, φ and σ are non-linear activation functions and $\|$ indicates the concatenation operation. For further convenition, we consider that GAT$(\cdot)$ representing for a gated graph attention layer.

b) Network flow: The architecture of the SOBOG model is depicted in Figure 3, in which information about tweets and users are fed to the model and processed separately before being combined to determine which accounts and tweets belong to bots. Firstly, both user representation vector x_i and corresponding tweet representation matrix $\boldsymbol{T}_i$ are fed into the embedding layer to get embedding features. Equation 6 shows the formulas inside the embedding layer.

$$\begin{cases} \hat{x}_i = \boldsymbol{W}_x x_i \\ \hat{\boldsymbol{T}}_i = \boldsymbol{W}_t \boldsymbol{T}_i \end{cases} \tag{6}$$

where $\boldsymbol{W}_x \in \mathbb{R}^{d\times m}$ and $\boldsymbol{W}_t \in \mathbb{R}^{f\times n}$ are trainable weight matrices. Next, we put tweet embedding features into M gated graph attention layers to extract information of tweet closure graphs by following (7).

$$\boldsymbol{T}_i^{(gated)} = \text{GAT}^{(M)}(\hat{\boldsymbol{T}}_i) \tag{7}$$

At next step, the tweet representations from gated layers are filtered to keep only original tweets and then fed into two branches. On one hand, they are concatenated with user embedding vector after passing through the max pooling layer. The combined vector is then used to classify whether this account is bot or not and is computed following (8).

$$\begin{cases} \tau_i^{(pooled,original)} = \text{Pooling}\left(\boldsymbol{T}_i^{(gated,original)}\right), \\ y_{x_i} = \sigma\left(\boldsymbol{W}_{y_x}\left(\hat{x}_i \| \tau_i^{(pooled,original)}\right)\right), \end{cases} \tag{8}$$

where y_{x_i} is the probability output, σ is sigmoid activation function, $\boldsymbol{W}_{y_x} \in \mathbb{R}^{1\times 2d}$ is a weight matrix. In the other hand, the original tweet representations from GAT layers are passed into some linear layers to get final representations as (9). We use them to detect whether tweets are belong to bot or not.

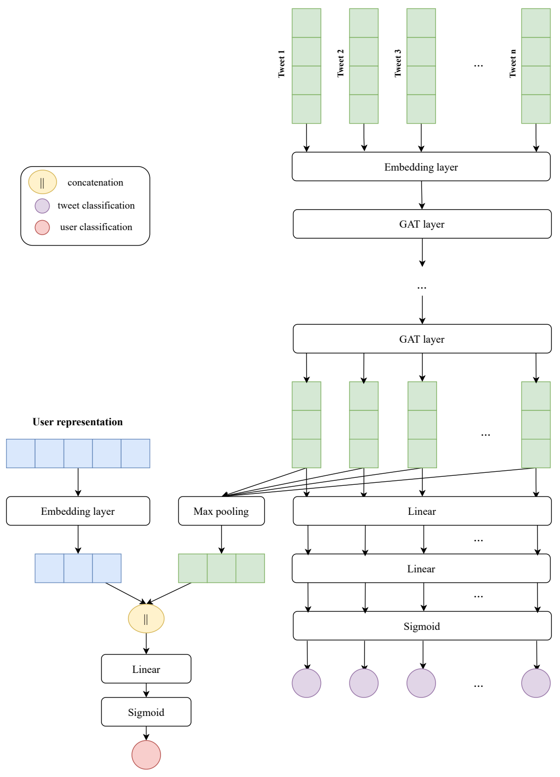

Fig. 3. SOBOG architecture

$$\Gamma_i^{(original)} = \Phi\left(\boldsymbol{T}_i^{(gated,original)}\right) \tag{9a}$$

$$y_{\boldsymbol{T}_i}^{(original)} = \sigma\left(\boldsymbol{W}_{y_t}\Gamma_i^{(original)}\right) \tag{9b}$$

where $y_{\boldsymbol{T}_i}^{(original)}$ are the probability outputs, σ is sigmoid activation function, $\boldsymbol{W}_{y_t} \in \mathbb{R}^{1\times d}$ is a trainable weight matrix and Φ is the combination of multiple linear transformations for outputting vectors in $\mathbb{R}^d$.

With supervised annotation, the cross entropy loss is chosen in our design, it can be described as (10).

$$\begin{cases} \text{Loss}_{x_i} = \text{BCE}(y_{x_i}, \hat{y}_{x_i}) \\ \text{Loss}_{\boldsymbol{T}_i} = \sum_{t_i\in\boldsymbol{T}_i} \text{BCE}(y_{t_i}^{(original)}, \hat{y}_{t_i}^{(original)}) \\ \text{Loss} = \sum_{x_i\in\boldsymbol{X}} (\alpha\text{Loss}_{x_i} + \beta\text{Loss}_{\boldsymbol{T}_i}) \end{cases} \tag{10}$$

In (10), BCE$(\cdot)$ is the *Binary Cross Entropy* loss function and $\alpha, \beta \in (0, 1)$ are hyper-parameters. The final loss allows the model to learn both tasks concurrently. Loss$_{\boldsymbol{T}_i}$ helps the model learn to represent tweets better. As a result, the model can detect bots more accurately.

As the optimization progress, the additional computational cost of recognizing the bot's tweets will enhance the model in extracting a better tweet representation.

TABLE II
STATISTICS ABOUT THE MIB-1 DATASETS [11]

Dataset	Accounts	Tweets
genuine accounts	3,474	8,377,522
social spambots #1	991	1,610,176
social spambots #2	3,457	428,542
social spambots #3	464	1,418,626
traditional spambots #1	1,000	145,094

TABLE III
STATISTICS ABOUT THE MIB-2 DATASETS [12]

Dataset	Accounts	Tweets
TFP (@TheFakeProject)	469	563,693
E13 (#elezioni2013)	1,481	2,068,037
FSF (fastfollowerz)	1,169	22,910
INT (intertwitter)	1,337	58,925
TWT (twittertechnology)	845	114,192

IV. EXPERIMENTS

A. Dataset

The majority of Twitter social bot detection studies employ the MIB dataset, which is made up of two distinct datasets we shall call to as MIB-1 [11] and MIB-2 [12]. We utilized them for our testing. Details of the datasets are described in Table II and Table III.

B. Model configuration

The hyperparameters are set as follows: number of TF-IDF features m is 20000, embedding dimension of tweet vectors and user vectors are 128 and 20, respectively, number of Gated Graph Attention layers M is 3, number of ReLU units is 64. Loss control arguments α and β were chosen from tuning values on the same architecture. $\alpha = 0.3$ and $\beta = 0.7$ gave the best result.

SOBOG implementation can be acquired from https://github.com/hcmut-epfl/SOBOG

We have set up our training strategy with the parameters as follows: the model is trained over 3 epochs, each epoch has a batch size of 4, adaptive optimization algorithm *Adam* is used as an optimizer and the base learning rate is 0.001.

C. Baseline models

We benchmark SOBOG with several proposed methods. To achieve a fair comparison, we have re-implemented the works below:

- Alarifi et al. [3] conducted automatic feature selection methods to finally use 8 features for a Random Forests classifier.
- Gilani et al. [4] introduced newly extracted features for checking their importance in identifying social bot accounts (activity source type, source count, CDN content size).
- Kudugunta et Ferrara [5] employed *Contextual LSTM* which took tweet semantics and tweet metadata into account.

TABLE IV
EXPERIMENTAL RESULTS ON ACCOUNT-LEVEL CLASSIFICATION ON MIB-1

Work	Accuracy	Precision	Recall	F1	MCC
SOBOG	**0.9936**	0.9872	**0.9957**	**0.9914**	**0.9863**
SOBOD	0.9877	0.9747	0.9928	0.9837	0.9740
[3]	0.9850	0.9886	0.9875	0.9880	0.9667
[4]	0.9869	0.9887	0.9904	0.9895	0.9719
[5]	0.9796	**0.9919**	0.9756	0.9837	0.9567
[6]	0.9170	0.8952	0.9836	0.9373	0.8232
[7]	0.9610	0.9627	0.9694	0.9660	0.9351
[8]	0.9865	0.9915	0.9870	0.9892	0.9713

TABLE V
EXPERIMENTAL RESULTS ON ACCOUNT-LEVEL CLASSIFICATION ON MIB-2

Work	Accuracy	Precision	Recall	F1	MCC
SOBOG	**0.9877**	0.9936	0.9735	0.9834	**0.9738**
SOBOD	0.9820	0.9723	0.9864	0.9793	0.9730
[3]	0.9516	0.9602	0.9612	0.9607	0.8979
[4]	0.9830	0.9908	0.9819	**0.9863**	0.9640
[5]	0.9711	**0.9937**	0.9597	0.9764	0.9401
[6]	0.8775	0.8630	0.9517	0.9052	0.7403
[7]	0.7538	0.6146	**0.9994**	0.7611	0.6042
[8]	0.9805	0.9928	0.9758	0.9842	0.9590

- Pasricha et Hayes [6] applied a lossless compression algorithm on encoded tweet type history and took the compression statistics for classification.
- Wei et Nguyen [7] considered using only tweet semantics and let *Bidirectional LSTM* perform the classification task.
- Yang et al. [8] extracted a total of 14 features and processed them using a Random Forests classifier.

The evaluation of tweet-level classifiers can also be converted into account-level. An account is evaluated as bot when the mean bot score (i.e. probability of a given tweet being created by a social bot account) across the tweet is greater or equal to 0.5. And an account with zero tweets will be classified as human.

D. Account-level evaluation

For account classification, a machine learning-based approach outperforms two deep learning-based approaches. The accuracy of these works exceeded 98% using only user property features. Using recurrent neural networks such as LSTM or Bidirectional LSTM to extract tweet semantics and employ it independently has also been shown to be effective in evaluating bot accounts within the dataset. However, SOBOG takes advantage of user properties, tweet semantics, and tweet relations, outperforming all baselines in four categories: accuracy, recall, F1-Score, and MCC.

E. Tweet-level evaluation

For tweet classification, Bidirectional LSTM architecture proposed by [7], Contextual LSTM by [5] and SOBOG achieve similar performance, the ranking varied

TABLE VI
EXPERIMENTAL RESULTS ON TWEET-LEVEL CLASSIFICATION ON MIB-1

Work	Accuracy	Precision	Recall	F1	MCC
SOBOG	0.9343	0.9482	0.9598	0.9540	0.8396
SOBOD	0.8805	0.9215	0.9089	0.9152	0.7134
[5]	0.9259	0.9155	0.9865	0.9497	0.8184
[7]	**0.9608**	**0.9557**	**0.9905**	**0.9728**	**0.9045**

TABLE VII
EXPERIMENTAL RESULTS ON TWEET-LEVEL CLASSIFICATION ON MIB-2

Work	Accuracy	Precision	Recall	F1	MCC
SOBOG	0.9476	**0.9787**	0.9651	0.9719	0.5905
SOBOD	0.9142	0.9506	0.9272	0.9388	**0.7964**
[5]	**0.9731**	0.9734	**0.9984**	**0.9857**	0.7708
[7]	0.9545	0.9605	0.9919	0.9759	0.5831

on two datasets. Overall, LSTM models achieve slightly better than SOBOG.

V. Discussion

From the experiment results, it is concluded that the model acquired decent performance on both the tasks of determining whether the account is a bot and whether the tweet is generated by the bot or not. This infers that the proposed method is capable of detecting whether both accounts and tweets are from bots, without affecting performance.

When comparing the results of SOBOG and SOBOD (the model without the tweet close graph information), we can see that SOBOG performs better. SOBOG achieved better results than SOBOD in both experiments to detect whether accounts or tweets are from bots, with higher accuracy and F1. Therefore, information about the relationship between tweets plays a positive role in detecting bot accounts.

VI. Conclusion

In this paper, we developed a deep learning-based model called SOBOG for resolving both account-level and tweet-level tasks in social bot detection, surmounting drawbacks that remain in each task. We also take tweet relations into account, which, to the best of our knowledge, is the first study on this type of information. SOBOG has become the top-tier classifier with 99.36% account-level accuracy and 93.43% tweet-level accuracy on the MIB dataset. However, we were unable to compare our work with studies on user neighborhood features due to schema in compatibility between datasets. In addition, TF-IDF may show a decrease in performance when dealing with large corpus having rich vocabulary. Our future direction for social bot detection consists of conducting data-oriented tasks and comparing the importance between tweet relations and user neighborhoods. Another proposal is to extract textual features from tweets using pre-trained transformer architectures.

VII. Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number IZVSZ2.203310

References

[1] M. T. Bastos and D. Mercea, "The brexit botnet and user-generated hyperpartisan news," *Social Science Computer Review*, vol. 37, no. 1, pp. 38–54, 2019. [Online]. Available: https://doi.org/10.1177/0894439317734157

[2] P. N. Howard, S. Woolley, and R. Calo, "Algorithms, bots, and political communication in the us 2016 election: The challenge of automated political communication for election law and administration," *Journal of Information Technology & Politics*, vol. 15, no. 2, pp. 81–93, 2018. [Online]. Available: https://doi.org/10.1080/19331681.2018.1448735

[3] A. Alarifi, M. Alsaleh, and A. Al-Salman, "Twitter turing test: Identifying social machines," *Information Sciences*, vol. 372, pp. 332–346, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516306077

[4] Z. Gilani, E. Kochmar, and J. Crowcroft, "Classification of twitter accounts into automated agents and human users," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ser. ASONAM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 489–496. [Online]. Available: https://doi.org/10.1145/3110025.3110091

[5] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, pp. 312–322, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518306248

[6] N. Pasricha and C. Hayes, "Detecting bot behaviour in social media using digital DNA compression," in *Proceedings for the 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science, Galway, Ireland, December 5-6, 2019*, ser. CEUR Workshop Proceedings, E. Curry, M. T. Keane, A. Ojo, and D. Salwala, Eds., vol. 2563. CEUR-WS.org, 2019, pp. 376–387. [Online]. Available: http://ceur-ws.org/Vol-2563/aics_35.pdf

[7] F. Wei and U. T. Nguyen, "Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings," in *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2019, pp. 101–109.

[8] K.-C. Yang, O. Varol, P.-M. Hui, and F. Menczer, "Scalable and generalizable social bot detection through data selection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 1096–1103, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5460

[9] F. N. Pakaya, M. O. Ibrohim, and I. Budi, "Malicious account detection on twitter based on tweet account features using machine learning," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–5.

[10] M. Stella, E. Ferrara, and M. D. Domenico, "Bots increase exposure to negative and inflammatory content in online social systems," *Proceedings of the National Academy of Sciences*, vol. 115, no. 49, pp. 12 435–12 440, 2018. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1803470115

[11] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 963–972. [Online]. Available: https://doi.org/10.1145/3041021.3055135

[12] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923615001803