Expert Systems With Applications 238 (2024) 122217

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Supervised learning models for social bot detection: Literature review and benchmark

Hoang-Dung Nguyen[a,b], Duc Q. Nguyen[a,b], Cong-Duy Nguyen[a,b], Phong T. To[a,b], Danh H. Nguyen[a,b], Huy Nguyen-Gia[a,b], Long H. Tran[a,b], Anh Q. Tran[a,b], An Dang-Hieu[a,b], Anh Nguyen-Duc[c], Tho Quan[a,b,*]

[a] *Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Viet Nam*
[b] *Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc City, Ho Chi Minh City, Viet Nam*
[c] *Department of Business and IT, University of South Eastern, Norway, P.O. Box 235, 3603 Kongsberg, Norway*

ARTICLE INFO

Keywords:
Social bot detection
Machine learning
Benchmark

ABSTRACT

Over the past decades, online social networks such as Twitter and Facebook have become a significant part of people's daily lives, particularly amid the ongoing global calamity — the COVID-19 pandemic. This gives room for social bot attacks that are designed to automatically replicate the behavior of real accounts. Most of these bots are employed for nefarious purposes such as disseminating false information, artificially amplifying the popularity of a person or movement, or spreading spam. Many studies have been conducted in an attempt to discover new strategies for identifying social bot accounts. To deal with large-scale attacks from social bots, Machine Learning has emerged as a noticeable path of bot detection problem that can handle massive amounts of data. However, the heterogeneity between studies in terms of problem statements, proposed processes, datasets, and evaluation metrics makes it difficult to assess and compare the efficiency of proposed methods. In this paper, we propose a systematic view of supervised learning methodologies for tweet-based social bot detection, ranging from shallow learning to specific deep learning models. In addition, we introduce a framework that measures various performance aspects and summarizes the in-depth analysis of the results, which were obtained using two datasets comprising 26224 labeled Twitter accounts. The results of this framework, we believe, will be beneficial as a practical guideline for other bot detection research or applications that require the use of machine learning techniques.

1. Introduction

Online social networks (OSNs) such as Facebook, Twitter, and Linkedin have become the most popular channels to share news and opinions, promote products and communicate among people within the same communities. Twitter, for example, has seen significant growth from 54 million active users in 2010 to 305 million in 2015 and 330 million as of 2019.[1] Another analytical source shows that there are more than 4.62 billion OSN users around the world in January 2022, equating to 58.4 percent of the total global population.[2] With such popularity, OSNs have also been exposed to misuse and vulnerabilities as Internet websites. Their large user base, easy-to-use functionality, and open nature further attract anti-social, malicious activities, where social bots are included.

Social bots are OSN accounts that imitate to be those of a real human but are fully operated by computer software (Shafahi et al., 2016-12). It is commonly reported the misuse of social bots to exploit and manipulate social media discourse with rumors, spam, malware,

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Viet Nam.

E-mail addresses: dungnguyen132wrk@gmail.com (H.-D. Nguyen), nqduc@hcmut.edu.vn (D.Q. Nguyen), duyngtr16061999@gmail.com (C.-D. Nguyen), phong.to19102001@hcmut.edu.vn (P.T. To), danh.nguyen.kaitokid@hcmut.edu.vn (D.H. Nguyen), nghuy1290@gmail.com (H. Nguyen-Gia), tranhoanglong2210333@gmail.com (L.H. Tran), anh.tq.2120@gmail.com (A.Q. Tran), dhan.sdh212@hcmut.edu.vn (A. Dang-Hieu), angu@usn.no (A. Nguyen-Duc), qttho@hcmut.edu.vn (T. Quan).

[1] https://backlinko.com/twitter-users
[2] https://datareportal.com/social-media-users

https://doi.org/10.1016/j.eswa.2023.122217
Received 24 October 2022; Received in revised form 15 October 2023; Accepted 16 October 2023
Available online 18 October 2023
0957-4174/© 2023 Published by Elsevier Ltd.

misinformation, slander, or noise (Ferrara et al., 2016). These bots are usually linked with the fake news generation and pose great threats to democracy, journalism, and freedom of expression. Social bots can be used during elections or moments of distinct, and country-specific, political conversation or crisis to demobilize an opposing party's followers, promote ideas, and attack targets on social media (Woolley, 2016). It is also evident that Twitter bots were used in U.S. presidential campaign in 2016, influencing discussion and opinions spared in Twitter meaningful political discussion (Trifiro et al., 2021; Woolley & Guilbeault, 2017)

During the COVID-19 pandemic, social bot attacks have had a significant impact on OSN users (Ferrara et al., 2020; Yang, Torres-Lugo, & Menczer, 2020). The outbreak of the pandemic has resulted in a significant surge in demand for information, leading to the proliferation of unregulated dissemination of untrustworthy, false, deceptive, and unconfirmed information (Ferrara et al., 2020). A public report estimates that nearly half of the Twitter accounts spreading messages on the OSN about the coronavirus pandemic are likely bots.[3]

Given the significance of social bots, social bot detection has been an emerging and active research topic for the last ten years (Cresci, 2020). Researchers have been working on different methods to detect and block social bots. Classical approaches to social bot detection include *graph-based* (or structured-based) and *crowdsourcing*. In graph-based approaches, social network relationships are employed with typical methods such as trust propagation-based, graph clustering, or making use of graph metrics and properties. Crowdsourcing refers to manual labeling, where human detection is required to distinguish between social bot accounts and human accounts. These approaches critically suffer from scalability issues and produce biases (graph-based approaches rely on the assumption that it is easier to traverse from a social bot account to another social bot account, crowdsourcing outputs are strictly followed the labeling schema from human's prior knowledge) (Alothali et al., 2018). In contrast, *Machine Learning* (ML) approach has been strongly discussed due to the potential of solving the problem on large amounts of data with numerous variables. Utilizing machine learning techniques can facilitate the process of identifying behavioral patterns based on the characteristics of user accounts and determine whether those accounts are likely occupied by bots or humans. The approach is believed to acquire faster and better performance compared to both graph-based and crowdsourcing.

Because of the high number of proposed machine learning detectors and their effectiveness, we focus solely on machine learning-based approaches for social bot detection in this study. Specifically, we will discuss supervised ML approaches where the labeled data is provided as input. Given a group of accounts to analyze, supervised detectors were applied to every account, to which they assigned a binary label (either bot or human). Our contributions are stated as below.

(i) First, we review state-of-the-art supervised learning models for social bot detection in a detailed perspective.
(ii) Secondly, we introduce a new framework for comparison among supervised ML models for social bot detection in the last decade. Our proposed framework is integrated with numerous datasets and recent methods, and has been implemented in a way that is convenient for future methods to be adapted into it. We made this framework publicly available at https://github.com/dfighter1312/social-bot-bnm-framework.
(iii) Finally, we conducted numerous experiments to evaluate methods and models from different perspectives, including multiple metrics, multiple scenarios, and datasets.

[3] https://www.npr.org/sections/coronavirus-live-updates/2020/05/20/859814085/researchers-nearly-half-of-accounts-tweeting-about-coronavirus-are-likely-bots

The remainder of the paper is organized as follows. Section 2 defines the social bot detection task. Section 3 reviews related methods using a supervised learning approach for social bot detection, which is summarized systematically following our proposed taxonomy. Section 4 describes the implementation of our benchmarking framework to compare the detectors. Section 5 lists out available datasets that we used for evaluating the detectors, along with the experimental results and outcome analysis. Finally, we conclude our key findings in Section 7.

2. Problem statement

We use the problem definition stated by Feng, Wan, Wang, Li, and Luo (2021) as a cornerstone, along with some additions and alternatives from our perspective. We limited the scope of social bots to Twitter, as this is the most popular platform where social bots are detected.

Let U be a public Twitter account with three major categories of information: list of uploaded tweets T, list of properties P and list of neighbors N. Each tweet includes semantic information of users T_s (i.e., user-generated natural language texts) and the metadata T_m such as the number of favorites, retweets, replies, URLs count, hashtags count and tweet timestamp. Property information is divided into two sub-categories: initialization P_i and activity P_a. Initialization consists of all the fields that can be initialized during the creation of the accounts (e.g., username, date of birth, gender, profile picture), while activity refers to the properties with immediate updates after every account U's behavior and/or other accounts affect on account U (follow, like, add friend, etc.). All fields in activity are numeric and initialized with zeros. Neighborhood information of an account is the list of its followers and followings, which can form a relationship graph. Note that some all-time network statistics, such as friend count and follower count are not classified into set N but set P_a. Moreover, to get all the information of an account U, the account status must be set to public. Otherwise, it is infeasible to perform the detection task since most features are hidden, leaving null fields in our dataset. Fig. 1 demonstrates how we divide Twitter user information into different groups.

Why do we need to divide into smaller categories? Existing works on tweet-based bot detection in the early stages, either using machine learning, crowd-sourcing, or a structured-based approach, focused on analyzing traditional bot accounts with bot-like patterns on a feature set, similar to answering questions using a bot detection algorithm. For example, from P_i, was the account randomly created to infer it as a bot? Or if we find a large proportion of bots is under an account's neighborhood N, can it be more likely to be a bot? However, social bot evolution has required more generalized methodologies for detecting bots. While traditional bots or spam bots cost cheaper but perform more predictable behaviors, sophisticated bots or even AI-powered social bots that now get a higher level of human imitation (Cresci, 2020). Dividing available features of a user account enables us to explain the growth in social bot detection using supervised learning, as well as support our summary to be more readable and systematic.

In supervised learning, social bot detection is defined as a binary classification problem, where the ground truth of an account can only be human ($y = 0$) or bot ($y = 1$). Therefore, the social bot detection task is defined as follows: Given an account U and its information T, P and N, learn a bot detection function $f : f(U(T, P, N)) \longrightarrow \hat{y}$, such that $\hat{y}$ approximates ground truth y to maximize prediction accuracy.

3. Supervised learning approaches for social bot detection

3.1. General pipeline

Early social bot detectors that employed Machine Learning algorithms were built based on a supervised learning approach. According to Cresci (2020), those detectors have been developed and published since 2010. The number of publications throughout the decade is still increasing and dominating other approaches in the machine learning

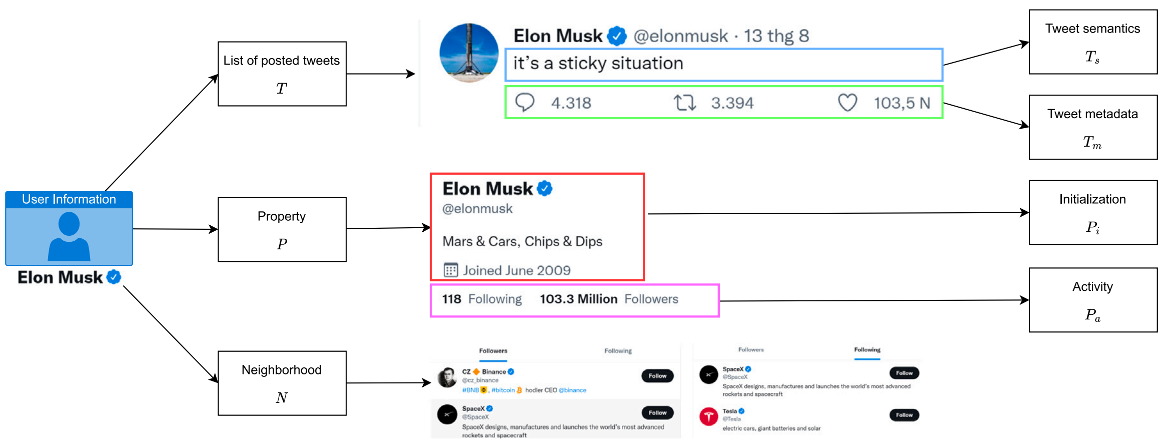

Fig. 1. Feature sets under user information.

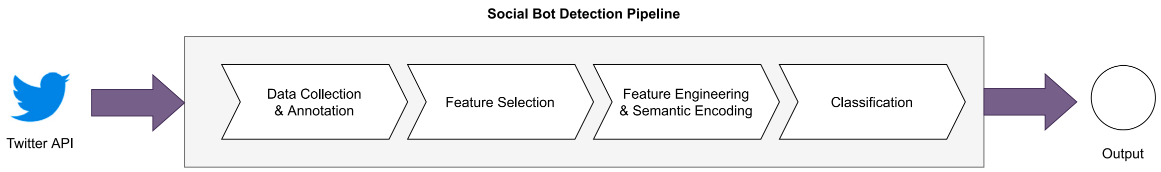
Social Bot Detection Pipeline
Twitter API
Data Collection & Annotation
Feature Selection
Feature Engineering & Semantic Encoding
Classification
Output

Fig. 2. The pipeline of supervised learning for social bot detection.

field, i.e., unsupervised, semi-supervised, adversarial, etc. A typical pipeline of model development for a supervised learning detector is described in Fig. 2:

- *Data Collection & Annotation*: a collection of Twitter accounts retrieved from Twitter API in which each object represents a user. Labeling tasks must be done by a particular labeling schema. Apart from crawling and manually annotating the data, using available datasets can reduce the time spent in this phase.
- *Feature Selection*: to pinpoint key features for concentrated analysis from the dataset. Selected features at this stage are primarily driven by the specific focus or assumptions retrieved from related studies.
- *Feature Engineering*: Besides transforming text and timestamp into numeric types so that machine learning algorithms can handle them, and narrowing down the list of features obtained from the previous stage after analysis, in *Feature Engineering*, some new features can be extracted from our knowledge. Shallow learning performance heavily depends on this part.
- *Semantic Encoding*: focuses on converting a user's tweets, or T_s, to a scalar or vector using vectorization or word embedding techniques.
- *Classification*: to tune the most suitable architecture that maps the inputs obtained from the last phase into desired outputs.

While *Feature Engineering* and *Semantic Encoding* can be employed or ignored in some works, others are included in the processes of developing most detectors.

3.2. Technique taxonomy

As shown in Fig. 3, *Social Bot Detection Techniques* can be divided into two groups: *Shallow Learning Techniques* and *Deep Learning Techniques*. *Shallow Learning Techniques* refer to typical Machine Learning algorithms such as Random Forest, Logistic Regression, Naive Bayes, Support Vector Machine, etc. (Pranckevicius & Marcinkevičius, 2017). Some approaches also combine with *Semantic Encoding*, which enhances semantic information of the tweets into a machine-readable format. Meanwhile, *Deep Learning Techniques* approaches rely on Deep Neural

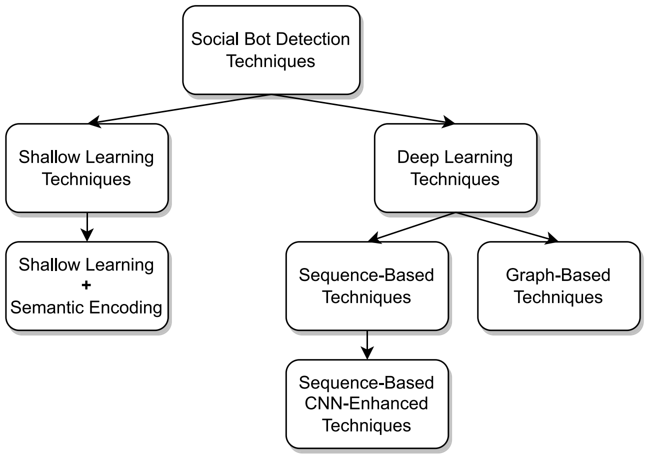
Social Bot Detection Techniques
Shallow Learning Techniques
Deep Learning Techniques
Shallow Learning + Semantic Encoding
Sequence-Based Techniques
Graph-Based Techniques
Sequence-Based CNN-Enhanced Techniques

Fig. 3. The taxonomy of social bot detection techniques.

Networks to address social bot detection challenges. Based on the nature of the data, these approaches encompass two primary directions: *Sequence-Based Techniques* and *Graph-Based Techniques*. Generally, *Sequence-Based Techniques* treat tweets as sequence data; therefore, they often utilize some Recurrent Neural Networks (RNN) based models like Long Short Term Memory (LSTM) to process data. This is because they are specifically intended for sequence data; some additional methods also use Convolutional Neural Network (CNN) model to extract more latent features, thereby enhancing the reliability of the outputs. On the other hand, *Graph-Based Techniques* represent the relationship between tweets and users as a graph and use Graph Neural Network (GNN) to extract respective features, which can be considered the latest approach in this field. In the sections that follows, we will discuss two main branches of current approaches: Shallow Learning and Deep Learning.

3.3. Shallow learning techniques

3.3.1. Feature extraction and processing

In Shallow Learning, extracting highly-predictive features may reveal interesting patterns that provide an understanding of the difference between bots and humans (Ferrara et al., 2016). Consequently, besides using original features that are available in Twitter API as well as in the datasets, some works derived more features that may be useful to distinguish between malicious and benign accounts.

Davis et al. (2016) created a web platform, *BotorNot*, to return the probability that a Twitter account is controlled by a human or machine. The system was first released in 2014, and it was the first social bot detection application for Twitter that raised awareness about the existence of social bots. From user data, the system employed over 1000 features, including network, user, friend, temporal, content, and sentiment features, with Random Forest as the classifier. It has been continuously improved, where later versions introduced additional training data and features to capture more sophisticated behavior and comply with changes from the Twitter API (Sayyadiharikandeh et al., 2020; Varol et al., 2017; Yang et al., 2022, 2019), Other research showed that with fewer features than *BotorNot*, the performance can still be retained (Ferrara et al., 2016) but with less memory and training time.

Detectors in Shallow Learning have primarily focused on user property P. In particular, user activity features P_a were analyzed and leveraged by 73% of our surveyed papers because such features like the number of followers, number of friends, protected indicator, etc., require a small effort of data transformation but clearly showed distinguishable patterns between social bots and human activity on social media. New features derived from P were commonly obtained by (i) counting the length of text generated by the account, taking screen name, username, and description as examples (ii) extracting the growth of the accounts in the number of followers, favorites, statuses, etc. by dividing all-time values by the account's lifetime, (iii) apply entropy, which is used to measure the randomness in a data signal in information theory, with the assumption that social bots' creations and their behaviors were more predictable than human.

3.3.2. Processing tweets with semantic encoder

There are limited studies processing tweet semantics that adopt Shallow Learning techniques due to two reasons. Firstly, Shallow Learning detectors were proposed in the early stages of social bot detection, when the targets had slightly small functionalities provided. Suspicious actions from these bots can be nimbly spotted via user property features. For instance, a group of social bots that are generated to follow one or several accounts that the botmaster selected will most likely have an abnormal account reputation (i.e., following/follower ratio). Similarly, social bots have a higher number of tweets, but fewer replies than human accounts (Ferrara et al., 2016) since these rule-based bots may not often be included in a conversation. Therefore, user property features were stated to be the most predictive and interpretable feature set (Ferrara et al., 2016). High accuracy was reported in Shallow Learning detectors that only leveraged this type of feature set.

Secondly, it requires less preprocessing effort because most of them are numerical or categorical properties, and no composite feature exists. Second, collecting and storing tweets are challenging for such research since APIs provided by OSNs do have limitations on the number of tweets per hour to be retrieved, and one user can make thousands of tweets, retweets, and replies. Not many available datasets were released before, including tweets for usage (Feng, Wan, Wang, Li, & Luo, 2021).

A simple transformation from tweets to numerical properties is taking the length of the tweets or counting the number of special tokens available on the corresponding OSN (e.g., URLs, hashtags and user mentions on Twitter). Entropy can be applied across these features to extract a feature that represents the randomness of the aforementioned counts, which was considered by Alarifi et al. (2016) and Kantepe and Ganiz (2017). However, this does not mean that semantic features were extracted. To obtain parts of its meaning, tweets, which are textual data, must be encoded using a semantic encoder. It is expected to look through every existing token within each tweet. For the Shallow Learning technique, semantic encoders transform tweets into vectors. These vectors can be directly concatenated with vectors obtained by extracting user property features P and tweet metadata T_m, or they can be aggregated into one or many scalars with more expressive meaning. In cases where the algorithm takes a hybrid approach such as SentiMetrix (Benamara et al., 2007; Subrahmanian & Reforgiato, 2008) or contrast pattern-based analysis (Loyola-González et al., 2019), it will not follow the above schema but directly get numerical values representing interpretable features (agreement, disagreement, irony, etc.) from every input tweet.

Some of the semantic encoders are described in Appendix A.

3.3.3. Data manipulation

Besides feature engineering, several data manipulation strategies were added which results in better performance along with feature engineering and semantic encoding. The intuition of using this method is due to data imbalance in most available datasets. For instance, Kudugunta and Ferrara (2018) with the aim of balancing the dataset, applied synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002) as an oversampling technique and Edited Nearest Neighbors (ENN) (Wilson, 1972) as a data enhancement. Addressing the problem of imbalanced datasets with too few samples of the minor class (i.e., the class with a smaller number of samples than the others) for a model to learn, SMOTE was introduced to perform data augmentation on the minor class by selecting two samples of that class, one sample is selected at random in feature space and the other one is selected among k nearest candidates to the first sample. Then, connect those samples with a line segment and get a new sample at a point along that line. Meanwhile, ENN is an undersampling technique for finding and removing ambiguous and noisy samples. For every sample x, the ENN algorithm searches for $k = 3$ nearest neighbors and performs a simple classification on x neighbors' majority voting. If it is misclassified, the sample x is deleted. It was claimed that the combination of SMOTE oversampling technique with an undersampling technique is better than undersampling itself (Chawla et al., 2002).

Echeverría et al. (2018) gathered two datasets, namely botnet and real-user datasets. For the botnet dataset, the authors utilized a wide range of bot datasets from earlier studies for its botnet dataset. To be more precise, they aggregated twenty botnet types, each with various traits and fingerprints. As a result, a dataset comprising more than one million bots from different sources with their available tweets is created. LOBO, which proposed their work, also offers novel-form testing, in which they train all bot classes aside from the target class in order to assess the predicted generalization of bot classification from the seen bot to the target class.

Rodríguez-Ruiz et al. (2020) set up the training set containing only legitimate Twitter accounts, and construct a one-class classifier to learn the behavior of such accounts. One-class classification is also considered as supervised classification and applies to general problems such as outlier detection and anomaly detection. The classifier is then expected to detect bots that deviate from human accounts regardless of the bot type, which gives the capability of detecting new types of bots. Their experimental results showed that one-class classifiers were better ranked than binary classifiers in most cases. Moreover, among five one-class classifiers, Bagging-TPMiner (Medina-Pérez et al., 2017) was the dominator.

Table 1
Summary of Shallow Learning methods. In case a work did an experiment on multiple classifiers, only the one with the best performance is reported.

Reference	Feature sets	Semantic encoder	Best classifier
Chu et al. (2012)	P, T	Orthogonal Sparse Bigram	Random Forest
Dickerson et al. (2014)	P, T_s, N	SentiMetrix (Benamara et al., 2007; Subrahmanian & Reforgiato, 2008)	Gradient Boosting
Davis et al. (2016)	P, T, N	Not mentioned	Random Forest
Morstatter et al. (2016)	P, T_s	BOW representation	BoostOR
Alarifi et al. (2016)	P_a, T_m	–	Random Forest
Kantepe and Ganiz (2017)	P, T	TF-IDF	Gradient Boosted Trees
Erşahin et al. (2017)	P, T_m	–	Naive Bayes
Gilani et al. (2017)	P_a, T_m	–	Random Forest
Khaled et al. (2018)	P	–	SVM-NN
Kudugunta and Ferrara (2018)	P_a	–	Random Forest
Echeverría et al. (2018)	P, T_m	–	Gradient Boosted Trees
Fazil and Abulaish (2018)	T_m, N	–	Random Forest
Beskow and Carley (2019)	P_i	–	Logistic Regression
Pakaya et al. (2019)	T_s	Binary Classification: TF-IDF Multi-level Classification: Word2Vec	XGBoost
Pasricha and Hayes (2019)	T_m	–	Logistic Regression
Loyola-González et al. (2019)	T	–	Random Forest & AdaBoost
Rodríguez-Ruiz et al. (2020)	P	–	Bagging-TPMiner (Medina-Pérez et al., 2017)
Yang, Varol, Hui, and Menczer (2020)	P	–	K-Nearest Neighbors
Karpov and Glazkova (2021)	N	–	Logistic Regression

3.3.4. Discussions

Several classifiers were utilized in Shallow Learning: Logistic Regression, Support Vector Machines for Classification, Random Forests, AdaBoost, Gradient Boosting, XGBoost, etc. Ensemble methods such as Random Forests or AdaBoost tend to get the best performance in most cases.

Shallow learning models are efficient on small training datasets, which can be concluded from our experiments described in Section 5. They can also be stated as reliable since the input features were analyzed to have different distributions between social bot accounts and legitimate accounts within the collected datasets.

However, the disadvantages of Shallow Learning are listed as follows.

- Most works and datasets were built on the assumption that the accounts were independent. In other words, human labeling and machine learning classifiers were examined in individual accounts to determine whether an account was a social bot or not. With better imitation of the new generation of social bots, malicious accounts become more likely to evade detection (Cresci, 2020).
- Shallow learning performance heavily relies on the quality of feature engineering which is the most challenging and time-consuming task and requires a considerable amount of data mining skill and domain expertise. Therefore, the detectors have slow improvement when new social bot attacks can be launched where the bots are designed with different characteristics.

Table 1 summarizes related works using Shallow Learning.

3.4. Deep learning techniques

The previous discussion shows that Shallow Learning methods achieve considerably good performance when handling a certain amount of data. However, it is known that when the volume of data becomes huge and complex, the accuracy of those approaches suffers from *saturation*, i.e., the accuracy cannot be furtherly improved even with more training data. The recent emerging technique, Deep Learning, can potentially solve this problem and has been attracting much attention in the research community. Thus, various approaches using Deep Learning models for social bot detection have also been reported with remarkable performance as subsequently discussed.

3.4.1. Sequence-based techniques

LSTM-Based Model. Fig. 4 illustrates the workflow of the system based on the sequence-based technique. For instance, consider a scenario in which *User 1* creates a tweet: “Next, I’m buying HCMUT to make assignments easier for students”. This tweet is then retweeted as “Check out this tweet from Elon Musk” by *User 2*. The content of this message can be categorized into two types of information. Firstly, tweet metadata T_m captures descriptive properties of the tweet (e.g., retweet count, reply count). Secondly, the tweet’s content is represented as tweet semantics T_s. In the sequence-based pipeline, T_s undergoes the following processing steps.

Word Embedding. *Word Embedding* transforms each word in T_s into its corresponding vector V_w. In consequence, a sequence of word vectors $S = V_{w_1}, V_{w_2}, \ldots, V_{w_n}$ is acquired. *Word2Vec*, which is introduced in Appendix A, is one of the most commonly-used Word Embedding techniques.

Sequence Modeling. Sequence models are leveraged to deal with sequence S. As a matter of course, the development of Deep Learning leads to numerous emergence of sequence models. In particular, several models are expressly developed for processing time-series data, named *Sequence Model*. They are principally constructed based on

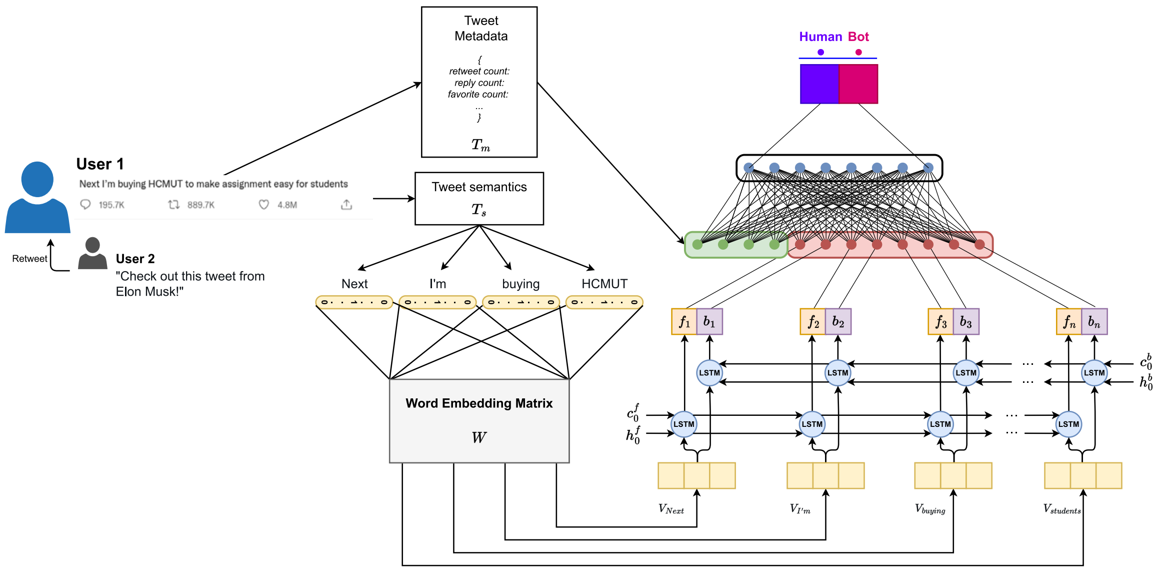

Fig. 4. Sequence-based architecture for solving social bot detection.

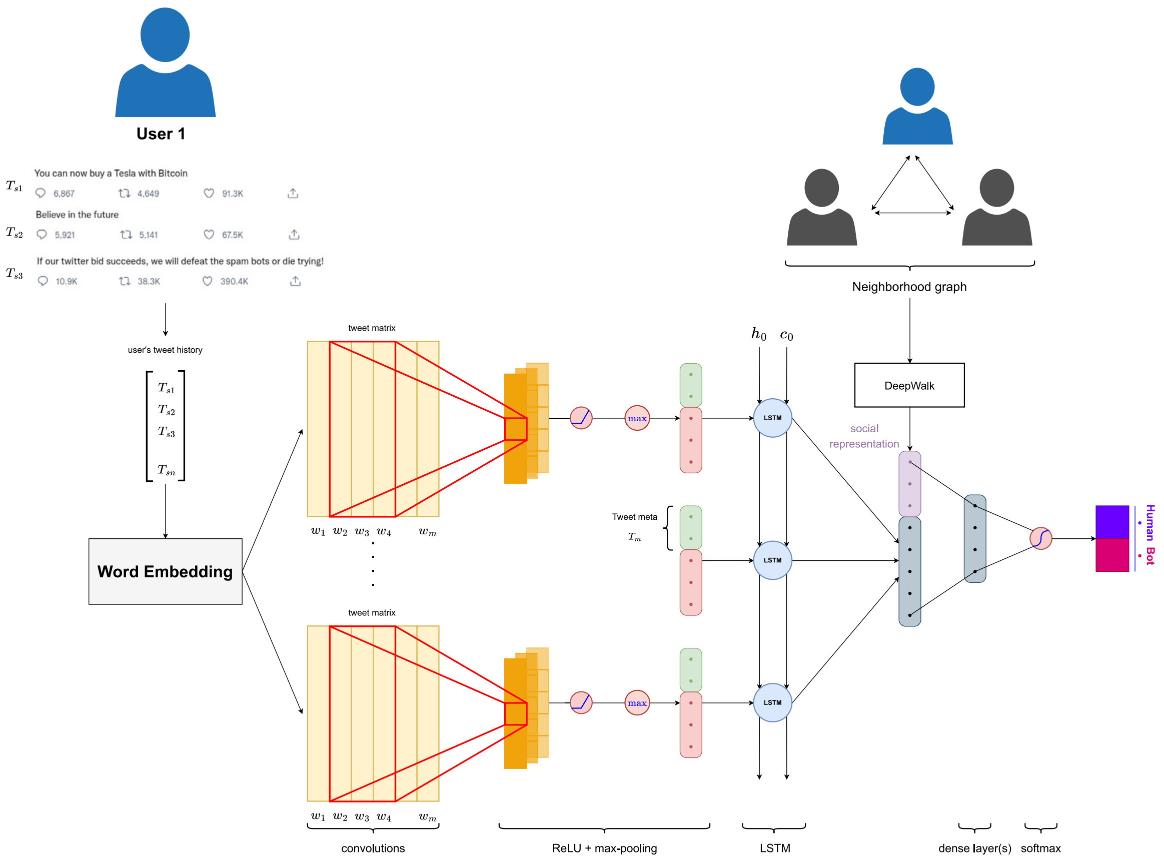

Fig. 5. Sequence-based enhanced CNN architecture proposed by Cai et al. (2017).

RNN (Tealab, 2018), the widely used *Long Short-Term Memory* (LSTM). In the problems of bot detection on social media, the most sophisticated model is *Bidirectional LSTM* (Bi-LSTM) consisting of two LSTM sub-models in opposite directions.

Comprehensive insights into the functioning of LSTM are expounded upon in Appendix B.

CNN-Enhanced Model. The architecture mentioned in Fig. 4 can be ameliorated by combining with CNN to extract latent features in multi-tweets as the architecture described in Fig. 5.

As discussed earlier in , from words $w_1, w_2, \ldots, w_m$ in T_s, a sequence S of word vectors $V_{w_1}, V_{w_2}, \ldots, V_{w_m}$ is generated. These word vectors are arranged together to generate a *tweet matrix* M_T, which has n rows and m columns, in there, n is the number of dimensions of word vector V_{w_i} and m is the number of word vectors in the sequence S.

CNN is one of the most important foundations of Deep Learning. Thanks to the weight-sharing mechanism, CNNs have a much smaller number of weights than Artificial Neural Networks (ANNs), thereby significantly saving computing time and resources.

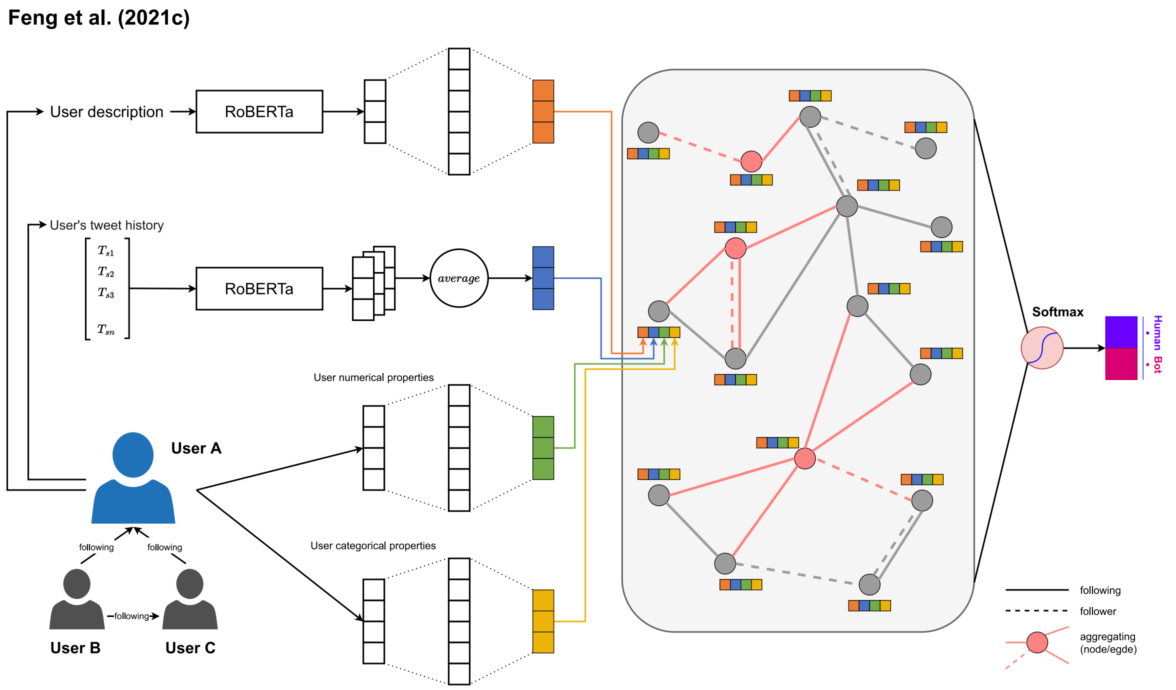

Fig. 6. Graph-based model architecture proposed by Feng, Wan, Wang, and Luo (2021) for solving social bot detection.

The intricate workings of CNNs are elaborated upon in Appendix C.

3.4.2. GNN-based techniques

In recent years, *Graph Neural Network* (GNN) has made a huge impact on various aspects in many distinct fields including Computer Science, Computational Biology, etc. Regarding the social bot detection problem, the main target is identifying whether a user is a bot or a human. By utilizing the relationship between users, Feng with his colleagues built a graph in which nodes are users and edges are following or follower relations (Feng, Wan, Wang, & Luo, 2021). A node corresponding to a user is represented by a D-dimensional vector which is created by concatenating four $(D/4)$-dimensional child vectors respectively as follows (see Fig. 6).

- User description vector (r_b): This vector is created by using RoBERTa model to encode the short biography on the user profile. Assuming the user's short biography is b, then the user-description vector is calculated by $r_b = \theta(\boldsymbol{W}_B \cdot RoBERTa(b) + b_B)$ where $\boldsymbol{W}_B$ and b_B are learnable weight matrix and bias respectively.
- User tweets vector (r_t): At first, each tweet i belonging to the user is presented by its content (semantic) T_{si} and then is encoded in the same way as user's short biography. Let r_{ti} be the encoded vector for tweet i, then $r_{ti} = \theta(\boldsymbol{W}_T \cdot RoBERTa(T_{si}) + b_T)$ where $\boldsymbol{W}_T$ and b_T are learnable weight matrix and bias respectively. After that, all the user-tweet vectors are aggregated to form a representation vector by $r_t = (1/N) \otimes \sum_{i=1}^{N} r_{ti}$ where N is the total number of tweets of that user.
- User numerical properties (r_p^{num}): This vector is obtained by performing z-score normalization on each numerical property of the user. Then, all those features are arranged into a vector and projected to a $(D/4)$-dimensional space by passing through a fully-connected layer. Considering f_i^{num} is the ith feature among total F_{num} numerical ones of the user. Then, the formal equation for the above transformations can be written as $r_p^{num} = LeakyReLU(\boldsymbol{W}_p^{num} \cdot [f_i^{num}]_{i=1}^{F_{num}} + b_p^{num})$ where $\boldsymbol{W}_p^{num}$ is learnable weight matrix, b_p^{num} is bias and σ is a non-linear activation function.
- User categorical properties (r_p^{cat}): To produce this vector, we need to encode all categorical features into one-hot vectors, then concatenate them into a F_{cat}-hot vector (F_{cat} is the total number of user's categorical properties). The fully-connected layer is used again to transform that vector into the $(D/4)$-dimensional space. Let f^{cat} be the F_{cat}-hot vector, then the user-categorical-property vector is computed by $r_p^{cat} = ReLU(\boldsymbol{W}_p^{cat} \cdot f^{cat} + b_p^{cat})$, where $\boldsymbol{W}_p^{cat}$ and b_p^{cat} are learnable weight matrix and bias respectively.

Let r_i be the feature vector for ith user, it is constructed by using $r_i = [r_b, r_t, r_p^{num}, r_p^{cat}]_i$. The following or follower relationships between users are kept separately as distinct edges. Now, the input data is completely converted into a graph structure. The initial social bot detection problem is renovated to the node classification problem, which is a well-known problem in graph analysis. Thus, various GNN-based models can be applied to solve this problem. According to Feng's study, all node feature vectors (user feature vectors) are first transformed into a higher representation by a simple linear projection. Then, the Relational Graph Neural Networks (R-GCNs) (Schlichtkrull et al., 2018), whose main purpose is to model relation data, are used to extract hidden features for each user. Specifically, assuming h_i^0 is the origin hidden feature vector of ith node, at the lth layer in R-GCN, each node feature vector is formed by summing itself and its neighbors at $(l-1)$th layer. Eq. (1) presents the detailed computing formulation.

$$\begin{cases} h_i^0 = ReLU(\boldsymbol{W}_0 r_i + b_0) & \text{(a)} \\ h_i^{l+1} = \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{|\mathcal{N}_i^r|} \boldsymbol{W}_r^{(l)} h_j^{(l)} + \boldsymbol{W}_s^{(l)} h_i^{(l)} & \text{(b)} \end{cases} \quad (1)$$

In (1), $\boldsymbol{W}_0$ and b_0 are learnable weight matrix and bias respectively; $\boldsymbol{W}_r^{(l)}$ and $\boldsymbol{W}_s^{(l)}$ are learnable weight matrices at lth layer; $\mathcal{R}$ is the set of relations (i.e. following and follower); and $\mathcal{N}_i^r$ is the set of ith node neighbors with relation r. Please notice that there are not any activation functions applied to each node at R-GCN layer in Feng's study, which poses a difference when compared with the original R-GCN.

Next, each vector at the final Lth R-GCN layer is transformed by a ReLU-activated fully-connected layer to obtain the final representation for each user. Finally, each user-representation vector is passed through a fully-connected layer with *softmax* activation function to classify

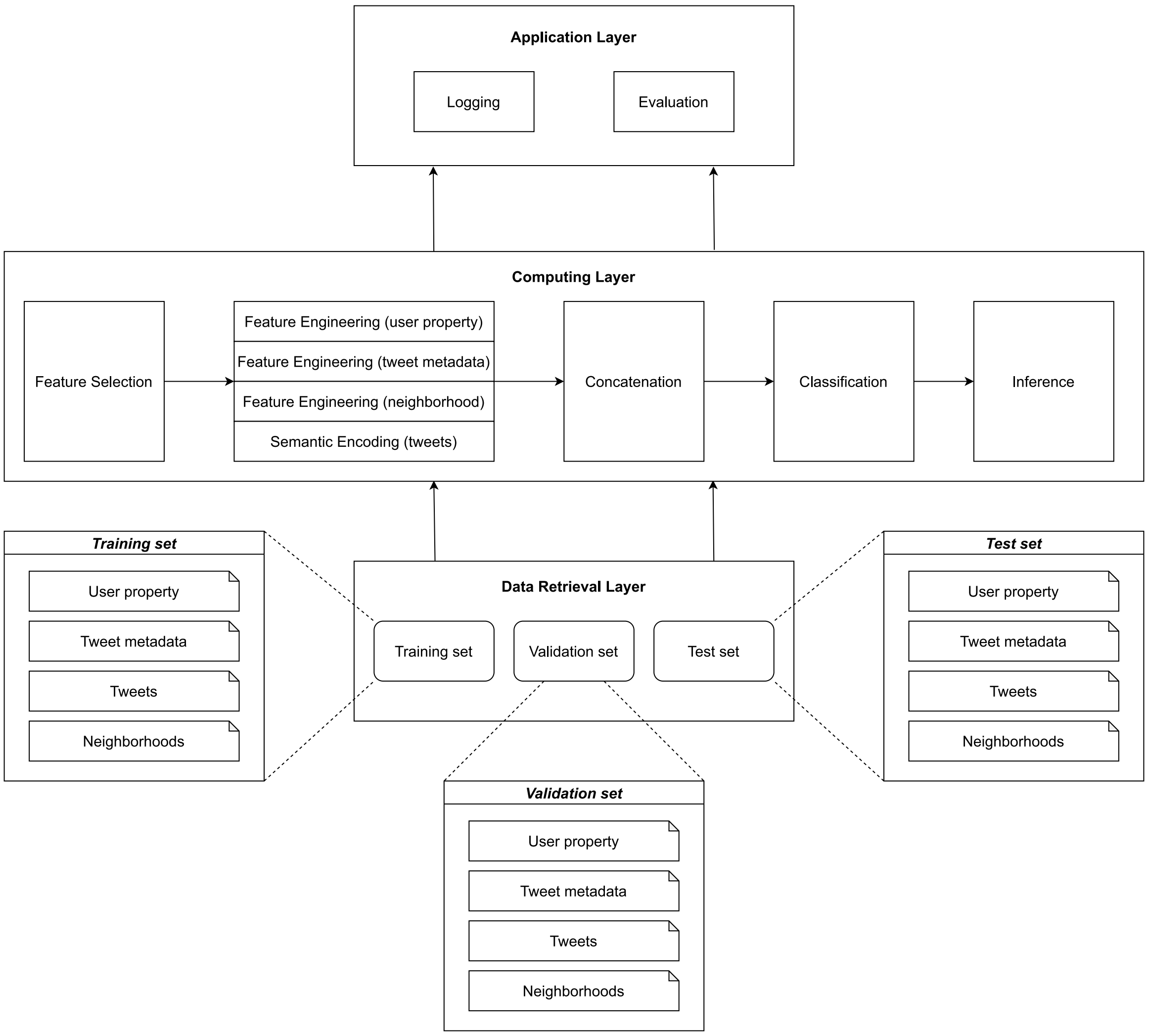

Fig. 7. The architecture of benchmarking framework.

whether it presents for a bot or a human. Eq. (2) demonstrates the computing process for predicting in a formal form.

$$\begin{cases} h_i^{repr} = ReLU(\boldsymbol{W}_1 h_i^L + b_1) & \text{(a)} \\ y_i = Softmax(\boldsymbol{W}_o h_i^{repr} + b_o) & \text{(b)} \end{cases} \tag{2}$$

In (2), $\boldsymbol{W}_1$ and $\boldsymbol{W}_o$ are learnable matrices; b_1 and b_o are biases; h_i^{repr} is the final user-representation vector and y_i is the output probability for predicting whether the ith user is a social bot or not.

3.4.3. Discussions

In general, Deep Learning approaches outperform Shallow Learning ones when handling a large dataset of text-based tweets. This is because these sophisticated models can effectively analyze and extract latent features from sequential data. However, these approaches have not yet leveraged the relationships between multiple tweets, as well as the inter-user relationships on Twitter. These relationships naturally introduce a graph-based representation, leading to the consideration of *Graph Neural Networks* (GNNs) for social bot detection.

The main drawback of these Deep Learning models is the exhaustive training time when using tweet semantic and user neighborhood features. In particular, more sophisticated architectures cause a substantial increase in training and inference time, requiring greater infrastructure to be deployed. Additionally, although the GNN architecture enables parallelization, taking all relationships existing in the Twitter-sphere into account may lead to performance overhead. Therefore, this approach tends to have better generalization but worse scalability in contrast to traditional machine learning algorithms. This claim can be clearly inferred from the experimental results, which will be discussed in the next section.

4. Benchmarking framework

4.1. Framework setup

In order to retrieve a non-discriminatory comparison between social bot detectors applying supervised learning approaches, we present the details for our benchmarking framework. The purpose of establishing the framework is to provide a flexible and powerful tool to support the implementation of methods as well as the observation of several evaluation metrics. The framework has been built on top of Python — a well-known and largely-used programming language specifically in data science, along with two powerful tools: *Pandas* and *Scikit-learn*. Pandas is an open-source data analysis and manipulation tool. To work with data in Pandas, every two-dimensional structured data composed of rows and columns will be converted into a *dataframe*. Scikit-learn is a tool specialized for applying machine learning techniques with lots of preprocessing extensions.

Fig. 7 depicts the architecture of our benchmarking framework. Our framework has a similar process to our proposed taxonomy and is divided into three major layers: *data retrieval layer*, *computing layer* and *application layer*. The *data retrieval layer* reads the data in *Comma-separated values* (CSV) format, transforms the data into dataframes, splits them into several sets, and passes them to the succeeding layers. The output of this layer consists of at most 12 dataframes in three groups: training data, validation data, and test data. Each group contains at most four dataframes: user property dataframe, neighborhood dataframe, tweet dataframe, and tweet metadata dataframe. The *computing layer* simulates a pipeline with four processes. Firstly, based on user-defined configurations, it removes unselected features. Then, a user-implemented preprocessing function is applied for each type of dataframe. We provide three optional preprocessing functions for the user property dataframe, neighborhood dataframe, and tweet metadata dataframe, along with a semantic encoding function. Before continuing, a type checker is employed to remove non-numeric features in those dataframes. Next, the processed dataframes in each group are combined into one big dataframe. Finally, the dataframes from the three groups are passed through a user-specified classification function for training and evaluation. The *application layer* shows the progress of processing the data as well as prints out the summarized information, facilitating test analysis.

The framework is made publicly available.[4]

4.2. Evaluation measures

There exists a large amount of research evaluating Machine Learning model and performance metrics that have been "de facto" standards in software engineering research. Having positive class (class 1) refer to social bot accounts, while negative class (class 0) designate human user accounts, we reported the following metrics:

- **Accuracy** quantifies the number of correct predictions over the test set.
- **Precision** is the ratio of correctly predicted positive classes to all items predicted to be positive.
- **Recall** is the ratio of correctly predicted positive classes to all items that are actually positive.
- **Matthews correlation coefficient (MCC)** measures the quality of binary classifications. If the predictions return good rates on true positives, true negatives, false positives and false negatives, the MCC score will be high and close to 1 (Matthews, 1975).
- **Training time** shows the time taken to process and fit the data to the classifier.
- **Inference time** shows the average time taken to predict the label for a user in the test set.

To ensure fairness between account-level detectors and tweet-level detectors, we only support account-level evaluation. After the inference stage, tweet-level predictions will be converted into account-level predictions by averaging the predicted scores of all tweets posted by each account.

5. Experimental results

In this section, we provide general information on social bot detection datasets that were leveraged in our benchmarking framework, describe the implementation process with several precautions and present our outlook on the results.

[4] https://github.com/dfighter1312/social-bot-bnm-framework

Table 2
Summary of applied datasets, from left to right, number of accounts, tweets and available feature sets: property P, tweet semantic T_s, tweet metadata T_m and neighborhood N.

Dataset	Accounts	Tweets	P	T_s	T_m	N
Cresci et al. (2017)	14,398	18,179,186	✓	✓	✓	
Feng, Wan, Wang, Li, and Luo (2021)	11,826	1,999,868	✓	✓		✓

5.1. Datasets

There exists around 20 small to medium datasets on social bot detection,[5] most of which were employed on shallow learning models. In this part, we introduce *MIB* – the most frequently used dataset and *Twibot-20* – the latest dataset proposed for social bot detection (see Table 2).

5.1.1. MIB

MIB (Cresci et al., 2017) contains a random sample of genuine human user accounts on Twitter along with seven other sub-datasets of bot accounts. The data was collected from various sources over a period of a few months in 2014 and contains profile information of 14,398 accounts with more than 18 million posts (tweets, retweets, and replies) in total. The bot accounts in this dataset are broadly categorized into three categories: social spambots, traditional spambots, and fake followers. The profiles of these social spambot accounts display detailed information such as profile pictures, biography, location, etc., although most of them were found to be either fake or stolen from other accounts.

5.1.2. Twibot-20

TwiBot-20 (Feng, Wan, Wang, Li, & Luo, 2021) is representative of the current generation of Twitter bots and genuine users. It is divided into four domains: politics, business, entertainment and sports and each user has semantics, property and neighborhood information. It is obtained by a controlled breadth-first search expanded from different seed users. TwiBot-20 establishes the largest benchmark of Twitter bot detection to date. The dataset was divided in a random partition of 7:2:1 on the labeled users to generate the training, validation and test set. In order to preserve the dense graph structure that follows relationship forms, they also provide unsupervised users as the support set of TwiBot-20.

5.2. Implementation

To compare the performance between methodologies, we made our best effort to fairly re-implement several works on our benchmarking framework. Re-implementations were done using Scikit-learn[6] for machine learning and Tensorflow[7] and Pytorch[8] for deep learning as programming libraries. In addition, the following rules were considered during re-implementation before obtaining final results:

- If the paper did not mention the model configurations, we will use the default configurations for the classifier applied in programming libraries.
- For the implementations of deep learning models, the training phase was done on three epochs and in the same device in order to fairly compare the training time which is a concern in the deep learning approach.

[5] https://botometer.osome.iu.edu/bot-repository/datasets.html
[6] https://scikit-learn.org/stable/
[7] https://www.tensorflow.org/
[8] https://www.pytorch.org/

Table 3
Experimental results on MIB dataset using our benchmarking framework (SL: Shallow Learning approach, DL: Deep Learning approach, TT (s): Training time (in seconds), IT (s): Inference time (in seconds)).

Reference	Category	MIB					
		Accuracy	Precision	Recall	MCC	TT (s)	IT (s)
Baseline model	SL	0.9897	0.9937	0.9898	0.9781	0.42	0.08
Yang, Varol, Hui, and Menczer (2020)	SL	0.9865	0.9915	0.9870	0.9713	2.39	0.49
Kudugunta and Ferrara (2018)	SL	0.9823	0.9908	0.9807	0.9623	0.39	0.05
Pasricha and Hayes (2019)	SL	0.9170	0.8952	0.9836	0.8232	15.61	8.10
Gilani et al. (2017)	SL	0.9869	0.9887	0.9904	0.9719	3.53	3.14
Alarifi et al. (2016)	SL	0.9850	0.9886	0.9875	0.9667	38.58	20.97
Kudugunta and Ferrara (2018)	DL	0.9796	0.9919	0.9756	0.9567	23 324.58	959.39
Wei and Nguyen (2019)	DL	0.9610	0.9627	0.9694	0.9351	121 188.57	10 275.44
Feng, Wan, Wang, and Luo (2021)	DL	0.8124	0.5847	0.9794	0.6491	250 512.63	23 471.98

- If the paper employed tweet metadata features, in the Twibot-20 dataset, they will be excluded. If that is the only feature set that the paper used, the result cannot be obtained.
- If the paper employed network features, in the MIB dataset, the corresponding model will use the network data collected by us, since it is not available in this dataset. We are aware that our retrieved network feature data is not identical to those at the time the original dataset was ingested, which could affect model learning.

To effectively evaluate models for real-life usage, we conducted three scenarios for benchmarking the selected models. Besides the traditional transductive scenarios, we also set up two additional scenarios for deeper evaluation including inductive scenarios and mixed-dataset scenarios. These three scenarios are described as follows.

- *Transductive scenario:* We performed splitting one dataset into two subsets of training and testing. The training set is used to train the model or tuning parameters and the testing set is used to benchmark the performance of trained models. The results of this scenario are illustrated in Table 3 (MIB dataset) and Table 4 (Twibot-20 dataset).
- *Inductive scenario:* We used MIB dataset for training and Twibot-20 for evaluation to observe how recent social bots' characteristics changed, which can fool the classifiers trained on the bygone dataset. Table 5 shows the comparisons between the performance of different models in this scenario.
- *Mixed-dataset scenario:* We merged the training and testing set of each dataset to obtain a combined set. It is expected that a classifier with good generalizability will induce no or minor reduction in performance. The performance of models in this scenario is presented in Table 6.

The models in transductive scenario were trained on a local machine with an Intel Core i7 6700HQ, 16 GB of RAM, and a single Nvidia Geforce GTX 960 GPU with 2 GB of memory. The models in other scenarios were trained using an Intel Core i5 9400F, 32 GB of RAM, and a Nvidia Quadro RTX6000 GPU with 24 GB of memory.

We also added a baseline model in the shallow learning approach that naively selects all numerical features and ignores the others and employed Random Forest as the binary classifier.

5.3. Evaluation

In this section, we analyze the characteristics of the methods, datasets and approaches to the performance behavior.

The evolution of social bots from the datasets. On one hand, most detectors did well on the MIB dataset with an average accuracy of about 0.98. On the other hand, since MIB was released in 2017 while the TwiBot-20 dataset was published in 2020, the manipulative behaviors of social bots became more diverse and complex, which results in a lower performance of all classifiers in the TwiBot-20 dataset. TwiBot-20 has become a more challenging dataset compared to other datasets served for the social bot detection task. Feng, Wan, Wang, Li, and Luo (2021), moreover, stated that in order to achieve desirable performance, incorporating such features as semantics and neighborhood must be considered. Note that the detector created by Pasricha and Hayes (2019) employs some tweet metadata features that indicate a tweet is a reply, a retweet, or simply a tweet and it cannot be experimented with in the TwiBot-20 dataset due to the unavailability of this feature set.

The effects of feature engineering in the shallow learning approach. Alarifi et al. (2016) and Gilani et al. (2017) focused on discovering newfangled features that may be more effective in conveying the disjunction between social bot accounts and human accounts. Pasricha and Hayes (2019), similarly, employed a lossless compression algorithm on Digital DNA sequences of users as a feature extraction technique. Digital DNA was designed to encode the historical tweet types of an account as a sequence of characters. In the account-level detection, Kudugunta and Ferrara (2018) worked on a small subset of 10 original features and emphasized the use of data enhancement techniques to improve the accuracy. Interestingly, our baseline model results in an exemplary performance without processing feature extraction, proving that original features can also have high importance and should be taken into consideration when building a model. Moreover, the social bot detector developed by Yang, Varol, Hui, and Menczer (2020) is statistically the best detector in the shallow learning approach which ranked second in the MIB dataset and ranked first in the Twibot-20 dataset. It uses 8 original user property features along with 12 derived features that mostly focus on the growth of the accounts on some statistics such as statuses count, favorites count, etc. Therefore, combining original features from the dataset and extracted features in shallow learning can engender a detector that detects a large proportion of elementary-level bots.

Performance and scalability trade-off in the deep learning approach. With the ability to leverage more features and automatically extract latent features throughout the layers in neural networks, the deep learning approach potentially obtains better performance in the Twibot-20 dataset. However, as mentioned in the previous section, deep learning detectors have to deal with the scalability problem. The detector built by Wei and Nguyen (2019) uses only tweet semantics as the input and applies GloVe embeddings and Bidirectional LSTM to classify the accounts. Their aim is to strike off the need for human-crafted features and make assumptions about profiles, neighborhoods, and behaviors of social media accounts. Nevertheless, from our experiments, the model training time is extensively high and it will

Table 4
Experimental results on Twibot-20 dataset using our benchmarking framework (SL: Shallow Learning approach, DL: Deep Learning approach, Acc: Accuracy, Pre: Precision, Rec: Recall, TT (s): Training time (in seconds), IT (s): Inference time (in seconds)).

Reference	Category	TwiBot-20					
		Acc	Pre	Rec	MCC	TT (s)	IT (s)
Baseline model	SL	0.8022	0.7576	0.9328	0.6140	0.97	0.06
Yang, Varol, Hui, and Menczer (2020)	SL	0.8183	0.7633	0.9625	0.6536	5.07	0.35
Kudugunta and Ferrara (2018)	SL	0.6374	0.8601	0.3938	0.3674	0.60	0.03
Gilani et al. (2017)	SL	0.7633	0.7466	0.8516	0.5237	2.87	0.38
Alarifi et al. (2016)	SL	0.5537	0.5780	0.6484	0.0923	3.21	0.38
Kudugunta and Ferrara (2018)	DL	0.8166	0.7474	0.9984	0.6686	4756.99	78.34
Wei and Nguyen (2019)	DL	0.5689	0.5621	0.9188	0.1172	31 204.35	540.29
Feng, Wan, Wang, and Luo (2021)	DL	0.8445	0.8239	0.9062	0.6882	38 539.17	5525.04

Table 5
Experimental results on MIB dataset for training and Twibot-20 dataset for evaluation using our benchmarking framework (SL: Shallow Learning approach, DL: Deep Learning approach, TT (s): Training time (in seconds), IT (s): Inference time (in seconds)).

Reference	Category	MIB for training – TwiBot-20 for evaluation					
		Accuracy	Precision	Recall	MCC	TT (s)	IT (s)
Baseline model	SL	0.4877	0.7044	0.1389	0.1041	0.18	0.09
Yang, Varol, Hui, and Menczer (2020)	SL	0.4727	0.6746	0.1035	0.0723	0.96	1.45
Kudugunta and Ferrara (2018)	SL	0.4781	0.7623	0.0920	0.1108	0.23	0.03
Gilani et al. (2017)	SL	0.4592	0.6600	0.0604	0.0480	0.30	0.93
Alarifi et al. (2016)	SL	0.4868	0.5628	0.3538	0.0083	1.32	0.60
Kudugunta and Ferrara (2018)	DL	0.5010	0.8034	0.1383	0.1614	1581.75	144.97
Wei and Nguyen (2019)	DL	0.4364	0.3455	0.0129	−0.0621	20 006.88	803.57
Feng, Wan, Wang, and Luo (2021)	DL	0.5298	0.6593	0.3231	0.1261	3302.84	9323.71

Table 6
Experimental results on the mixed dataset using our benchmarking framework (SL: Shallow Learning approach, DL: Deep Learning approach, TT (s): Training time (in seconds), IT (s): Inference time (in seconds)).

Reference	Category	Mixed dataset					
		Accuracy	Precision	Recall	MCC	TT (s)	IT (s)
Baseline model	SL	0.8487	0.7062	0.9198	0.6963	0.74	0.05
Yang, Varol, Hui, and Menczer (2020)	SL	0.9334	0.8592	0.9524	0.8554	3.49	0.58
Kudugunta and Ferrara (2018)	SL	0.8467	0.8699	0.6245	0.6410	0.38	0.01
Gilani et al. (2017)	SL	0.9117	0.8458	0.8927	0.8029	1.69	0.34
Alarifi et al. (2016)	SL	0.5430	0.4040	0.7969	0.2117	2.11	2.39
Kudugunta and Ferrara (2018)	DL	0.8543	0.7377	0.8698	0.6912	1300.14	515.86
Wei and Nguyen (2019)	DL	0.7866	0.9728	0.3665	0.5146	12 506.51	3252.00
Feng, Wan, Wang, and Luo (2021)	DL	0.8861	0.7855	0.8963	0.7541	8915.38	3647.26

be burdensome for hyperparameters tuning. Meanwhile, concatenating user and tweet metadata features with encoded semantic characteristics in contextual LSTM with simpler architecture (Kudugunta & Ferrara, 2018) still gives a better classification performance while scaling down the time spent for training and inference. In conclusion, deep learning detectors should be aware of model complexity so that they can be deployed and evaluated in real-world situations where instant detections are required to reduce the manipulation caused by new social bots.

The effects of collecting user relationships to the performance of Graph Neural Networks. Surprisingly, BotRGCN, which was proposed by Feng, Wan, Wang, and Luo (2021), achieves the best results in Twibot-20, but the worst in MIB dataset. There exist two possible explanations for the drastic reduction in the performance of the model on the MIB dataset. First, as stated in Section 5.2, network features in this dataset were collected by us, and they are surely dissimilar to those at the time the original dataset was ingested. Several accounts were banned and deleted as well, which makes retrieving any information nonviable. Second, each account information within the Twibot-20 dataset consists of at most 20 relationships, 10 followers, and 10 followings, using a controlled-BFS approach, whereas all acquired relationships for every user in the remaining dataset, MIB, were included. In consequence, MIB is not preferable to apply Graph Neural Networks, and the efficiency of this architecture is required to be proven in more experiments.

Performance evaluation in inductive setting. Generally, the experiment of all classifiers in this scenario induced a drastically reduction in accuracy, precision, recall and MCC scores. Therefore, a fixed model for social bot detection over a long haul is discouraged. Nevertheless, deep learning methods slightly outperform shallow learning from the result given by Table 5, with highest accuracy rate at 0.5298. This shows that modern neural networks architectures have the ability to inspect social bots whose characteristics did not exist in the training dataset.

Performance evaluation in mixed-dataset setting. Experimental results from this setting is given by Table 6. With painless feature extraction process, the model proposed by Yang, Varol, Hui, and Menczer (2020) produces a relatively higher performance compared to other competitors. Meanwhile, there is no clear statistical difference between shallow learning and deep learning models due to the previous observation in the transductive scenario that shallow learning did well in detecting classical social bots in the MIB dataset, while deep learning enables to exploit semantic and network features, which were used to detect more sophisticated bots appear in the Twibot-20 dataset. Hence, a hierarchy-based approach which employs a combination of distinct shallow and deep learning models can be considered to gain high performance and scalability. With this approach, low inference time is taken for detecting simple bots, and an understandable duration is required for correctly determine such sophisticated bots.

5.4. Ablation study

We conducted experiments on three aspects in order to investigate the most optimal components for constructing a deep learning variant that produces a solid performance on two aforementioned datasets — MIB and Twibot-20. First and foremost, taking tweet semantics as the only input, we compared two types of word embeddings: GloVe and Word2Vec for word-level embedding, and TF-IDF for document-level embedding. Secondly, one of three basic deep learning architectures was attached to output the likelihood of a tweet posted by a social bot. These architectures include Multi-layer Perceptron (MLP), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) — a variant of Recurrent Neural Networks (RNNs). In addition, we examined whether the model's size and complexity made any impact by configuring the number of layers and units within a layer. Finally, we appended tweet metadata and user property features to observe the effects on model prediction.

The effects of word embedding techniques. In the MIB dataset, TF-IDF gave superior results compared to GloVe and Word2Vec. This can be clearly explained by the nature of the dataset. While interested domains in genuine accounts that existed in the MIB dataset were varied since they were randomly chosen, bot accounts were collected from specific events (e.g., an Italian political election, a mobile app promotion, Amazon product-on-sale promotion). Therefore, tweet domains from social bot accounts were narrower than from legitimate ones, resulting in two distinct word clouds, and easier classification on the dataset but worse generalization in real word cases. On the other hand, Twibot-20 restricted accounts' interested domains in 4 categories: politics, business, entertainment, and sports. Consequently, comparisons of model performance using tweet semantics can be stated as more convincing if conducted in this dataset. The experiment results show that GloVe and Word2Vec embeddings with LSTM produce a slightly better performance than TF-IDF. A considerable side of choosing the word embedding technique is tweets' multilingualism in both datasets. The proportions of tweets created from non-English speaking countries are relatively high, giving a large vocabulary collected from all tweets. In our experiments, TF-IDF and Word2Vec had to be trained, and GloVe was employed as a pre-trained instance from a large number of tweets. With generalized datasets, training a word-level embedding like Word2Vec or a more complex embedding technique such as RoBERTa may overwhelm training time and consume tremendous amounts of memory (which can be observed by the training time result in the work proposed by Feng, Wan, Wang, and Luo (2021) in Tables 3 and 4). To conclude, deciding on a word embedding technique to be used may depend on the specificity of the dataset. In a domain-specific dataset, word-level embeddings can be adapted rather than document-level embeddings to deeply distinguish the way humans and bots compose tweets. In a dataset that contains tweets in multiple languages, selecting a pre-trained word-level embedding or a document-level embedding can acquire better usage (see Table 7).

Table 7
Study on different word embedding using our benchmarking framework.

Encoder	Layer type	MIB		Twibot-20	
		Accuracy	MCC	Accuracy	MCC
GloVe	LSTM	0.9261	0.8513	0.6788	0.3645
Word2Vec	LSTM	0.8916	0.7171	0.7185	0.4393
TF-IDF	Dense	0.9399	0.8820	0.6508	0.3506

Table 8
Study on different semantic encoder configurations using our benchmarking framework.

Layer type	Units	Layers	MIB		Twibot-20	
			Accuracy	MCC	Accuracy	MCC
Dense	32	2	0.6750	0.5078	0.6484	0.3026
Dense	32	3	0.6994	0.5371	0.6560	0.3172
Dense	64	3	0.6850	0.5198	0.6585	0.3225
CNN	32	2	0.5315	0.3346	0.6145	0.2301
CNN	32	3	0.5290	0.3315	0.6162	0.2355
CNN	64	3	0.5222	0.3241	0.6010	0.2033
LSTM	32	2	0.9216	0.8499	0.6771	0.3616
LSTM	32	3	0.9261	0.8513	0.6788	0.3645
LSTM	64	3	0.9313	0.8667	0.6788	0.3645

The effects of semantic encoder configurations. In this part, we experienced 3 different architectures: MLP, CNN and LSTM for GloVe embeddings (which returns a sequence of vectors for every processed tweet). It can be inferred from Table 8, which is an assessment of the complexity of the model, that LSTM gives the best results, followed by MLP and CNN ranked last. This can be explained as LSTM being the most basic yet efficient architecture in natural language processing tasks. Surprisingly, MLP performed better than CNN, which may come from the length of short sentences. Moreover, the larger model did not significantly improve the results in LSTM, while in MLP and CNN, the most complex model gave worse results than the small models.

The effects of tweet metadata and user property features. Combining additional information does have an influence on decision-making, which results in an increase to almost absolute in evaluation metrics within the MIB dataset. Both feature sets showed their potential due to the better performance when combining tweet semantics with either set, and the best one acquired when all were employed. This statement has also been proved by related works with shallow-learning approaches, and by several studies using complex deep-learning methodologies. In our perspective, for a generalized detector, naively attaching numerical and categorical data from tweet metadata and user property features only gives room for detecting simple social bots with weak human imitation such as accounts that are created to follow only one designated user but forget to make a change on their display name (keeping as by default name generator). Adding tweet semantics can also help to check the similarity between tweet meanings posted by an account, which gives the ability to detect intermediate AI-powered social bots. However, leveraging only numbers (except IDs) may let advanced bots evade the detection of social bot detectors, even when Deep Learning models are the great latent feature extractors. Manually investigating and transforming such fields containing IDs and timestamps may boost the performance in the Twibot-20 dataset as well as in real-world cases. Deep Learning researchers can consider some feature extraction techniques from shallow learning approaches in order to fuse more information as inputs (see Table 9).

6. Challenges

Data Shift. The experimental results in the inductive scenario indicate that more improvements are required to effectively identify social

Table 9
Study on the feature set effects using our benchmarking framework (T_m: tweet metadata is included, P: user property features are included).

Feature sets		MIB		Twibot-20	
T_m	P	Accuracy	MCC	Accuracy	MCC
		0.9216	0.8499	0.6771	0.3616
✓		0.9324	0.8686	–	–
	✓	0.9846	0.9669	0.7320	0.4665
✓	✓	0.9864	0.9708	–	–

bots. The only plausible explanation is that data shift occurs frequently in OSNs. In Twitter, the follower growth rate every month varies from 1% to 7.92%.[9] Furthermore, accounts created by former celebrities may have exceptionally high follower growth rates and other properties. It is claimed that the fastest account to reach one million followers only took four hours to achieve this accomplishment.[10] Therefore, model retraining needs to be reviewed in social bot detection applications to ensure that it can be continuously delivered with high reliability.

Explosive Twitter-sphere. The relationships between users (i.e., following and followers) as well as tweets (i.e., reply and retweet) form a Twitter-sphere, which is extensive and poses a challenge for graph-based methods. Therefore, when creating or leveraging a dataset with graph information, choosing a strategy to determine useful relationships between accounts, tweets, or account-tweet is highly recommended to be given great attention in the future. For instance, in the Twibot-20 dataset (Feng, Wan, Wang, Li, & Luo, 2021), at most 10 followers and 10 followings per user are collected. If a graph-based model considers vague instances such as inactive accounts or tweets with a relatively small number of reaches, it would result in zero improvements but lead to a large performance overhead.

Feature Selection. So far, along with users' general information, tweets and their metadata have become a higher concern for recent social bot detectors. Specifically, Shallow Learning models commonly extract behavioral signals from tweet metadata, such as how frequently an account generates tweets, including hashtags, URLs, or user mentions. In contrast, Deep Learning approaches use tweets from the dataset and let the hidden layers extract features. However, social bots with the support of generative and large language models like GPT-3 (Brown et al., 2020) can evade detectors that heavily rely on tweet semantics. For this reason, it is possible for deep learning models to come up with a more hybrid approach. In addition to the current trend of leveraging network information, temporal information from tweet metadata could be processed, or other types of media such as images and videos could be incorporated into a multimodal architecture in future research.

7. Conclusion

This paper presents a thorough evaluation and comparison of social bot detectors that are created in the supervised learning approach. We have tried to give the most systematic report on existing works by dividing user information into subsets and introducing a taxonomy that reflects the process of building any detectors. We then provided a flexible and extensible benchmarking framework in which new classifiers or datasets can be easily plugged. From the framework, we have re-implemented several detectors, evaluated them in a fair manner, and analyzed characteristics of detectors and datasets on various classification performance metrics including accuracy, precision, recall, and MCC as well as scalability measures such as training and inference time.

Our principal findings are stated as follows:

- Supervised learning methods for social bot detection have been being a considerable space for researchers to explore. Initially, the detectors were developed from machine learning algorithms. Deep learning emerged as a new branch of detecting social bots. The number of publications in both approaches has simultaneously increased over recent years.
- Descriptions of model construction processes are diverse between studies. Still, they can be generalized in four consecutive steps: data collection & annotation, feature selection, semantic encoding & feature engineering, and classification.
- Shallow learning detectors have better scalability and can be deployed for real-time applications, but their performances are limited due to the account-independence assumption. While deep learning approaches overcome this assumption to obtain better generalization, model complexity prevents these approaches to provide widely-used products.

Our future directions consist of (i) improving the benchmarking framework in order to plug in any detectors in various types of learning such as unsupervised or semi-supervised learning, as well as building a higher level of abstraction for studies with the desire to use our framework, and (ii) developing a detector that employs informative features in a simple deep learning model to balance the performance and training time trade-off.

CRediT authorship contribution statement

Hoang-Dung Nguyen: Conceptualization, Methodology, Software, Investigation, Visualization, Writing – original draft. **Duc Q. Nguyen:** Methodology, Visualization, Resources, Validation, Writing – original draft. **Cong-Duy Nguyen:** Methodology, Software, Investigation, Writing – original draft. **Phong T. To:** Visualization, Writing – original draft. **Danh H. Nguyen:** Visualization, Writing – original draft. **Huy Nguyen-Gia:** Investigation, Writing – original draft. **Long H. Tran:** Investigation, Writing – original draft. **Anh Q. Tran:** Investigation, Writing – original draft. **An Dang-Hieu:** Investigation, Writing – review & editing. **Anh Nguyen-Duc:** Validation, Writing – review & editing. **Tho Quan:** Validation, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tho Quan reports financial support was provided by Vietnam National university Ho Chi Minh City.

Data availability

Data will be made available on request.

Acknowledgments

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number NCM2021-20-02.

Appendix A. Semantic encoders for social bot detection

In this appendix, we present a detailed analysis of the semantic encoders employed in related social bot detection frameworks.

- *Bag of Words (BOW) representation*
  Applied by Morstatter et al. (2016), BOW is a representation of textual data that describes the occurrence of words within a tweet. It outputs a vector for each tweet with a dimension equals to the vocabulary size (i.e., the number of distinct words that occurred in the whole training corpus). For example, given

[9] https://locowise.com/blog/twitter-study-follower-growth-and-engagement
[10] https://www.guinnessworldrecords.com/world-records/fastest-time-to-reach-1-million-followers-on-twitter

	I	am	a	social	bot	human	we	love	twitter
(a)	1	1	1	1	1	0	0	0	0
(b)	1	1	0	0	0	1	0	0	0
(c)	0	0	0	0	0	0	1	1	1

Fig. A.8. An example of transforming sentences to BOW representation. Sentence (a) "I am a social bot". Sentence (b) "I am human". Sentence (c) "We love Twitter".

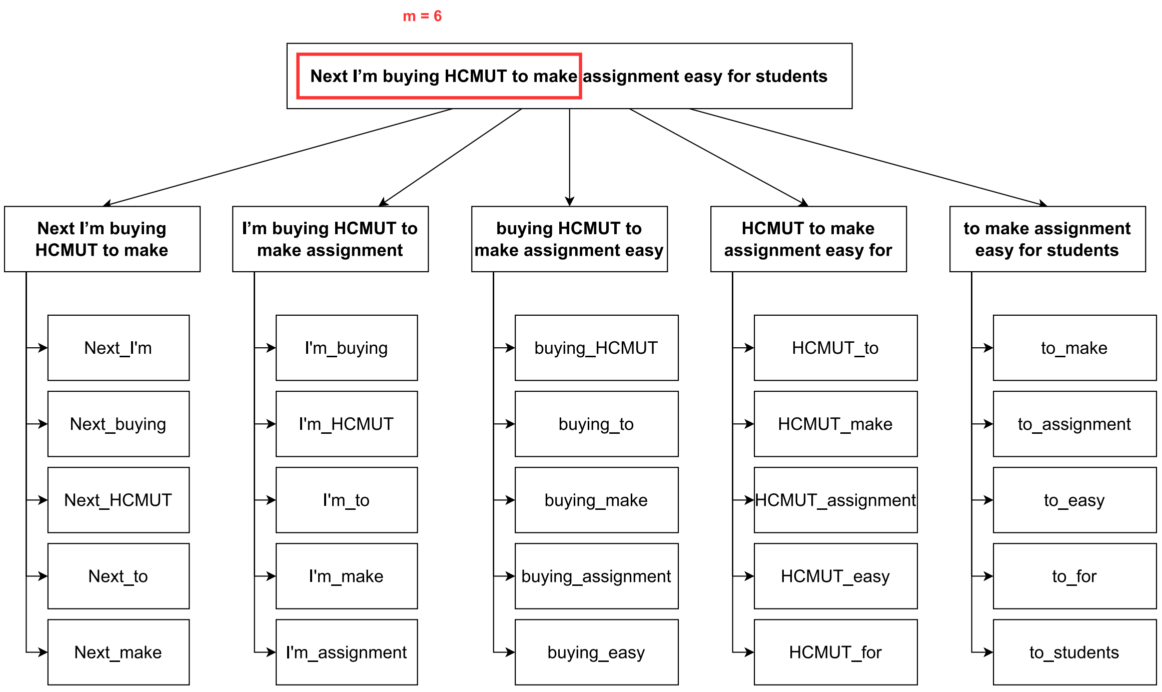

Fig. A.9. An illustration of extracting orthogonal sparse bigrams from a sentence.

a corpus with 3 sentences: (a) "I am a social bot", (b) "I am human", (c) "We love Twitter", we have the vocabulary with n words: I, am, a, social, bot, human, we, love, twitter and the corresponding BOW representations are shown as in Fig. A.8.
n-gram, or "Bag of n-gram" is a variation of BOW, whereas the vocabulary contains strings corresponding to sliding a window of n words. If we take $n = 2$ with the previous corpus, then the vocabulary is now: I_am, am_a, a_social, social_bot, am_human, we_love, love_twitter. n-gram is experimented by Pakaya et al. (2019), but it did not give the highest performance. However, there exists a variant of bigram that was proposed in a social bot detection paper, Orthogonal Sparse Bigram.

- *Orthogonal Sparse Bigram*
  The transformation was employed by Chu et al. (2012). In Orthogonal Sparse Bigram, vocabulary consists of bigrams generated by sliding the window of size m over the text and outputting every pair of words that includes the first word in the window. Take the tweet with the content: "Next I'm buying HCMUT to make assignment easy for students" as an example. If $m = 6$, then 5 six-word windows are obtained, each of which produces 5 bigrams to totally acquire the final 25 bigrams as illustrated in Fig. A.9.
- *Term frequency-Inverse document frequency* (TF-IDF)
  TF-IDF employs the frequency of terms (words) to determine how relevant and vital those words are to a given document. The *term frequency* $tf(t,d)$ is the number of occurrences of a specific term t in a document d. Term frequency indicates how important a particular term is in a document. There are multiple ways of defining frequency, but the well-known measure is described as follows.

$$tf(t,d) = \frac{n_t}{s} \tag{A.1}$$

In Eq. (A.1), n_t, s denotes the number of particular terms t in d and the number of terms in d. *Document frequency* $df(t)$ is the number of documents containing a specific term t. *Inverse document frequency* (IDF) is the weight of the term t, it aims to reduce the weight of the term if the term appears frequently throughout all the documents. IDF can be calculated as follows.

$$idf(t) = log(\frac{N}{df(t)}) \tag{A.2}$$

In Eq. (A.2), $df(t)$, N denotes TF score of term t, the number of documents. Then the TF-IDF value of the specific term t in document d is obtained by multiplying two values $tf(t,d)$ and $idf(t)$.

$$tf\_idf(t,d) = tf(t,d) \times idf(t) \tag{A.3}$$

- *Word2Vec*
  The word vectors generated from Word2Vec are contingent on context dependency between words, which is clearly useful for differentiating homonyms. This means that words in different contexts have different meanings. The inferred meaning of a word, which we call the focus word, depends on context words (or surrounding words). The word w in text is expressed to one hot vector $\widetilde{V_w}$ having a number of dimensions equal to the number of words in vocabulary denoted V. Vector $\widetilde{V_w}$ is respective to one sequence of bit 0, and only has one bit 1 respective to the index

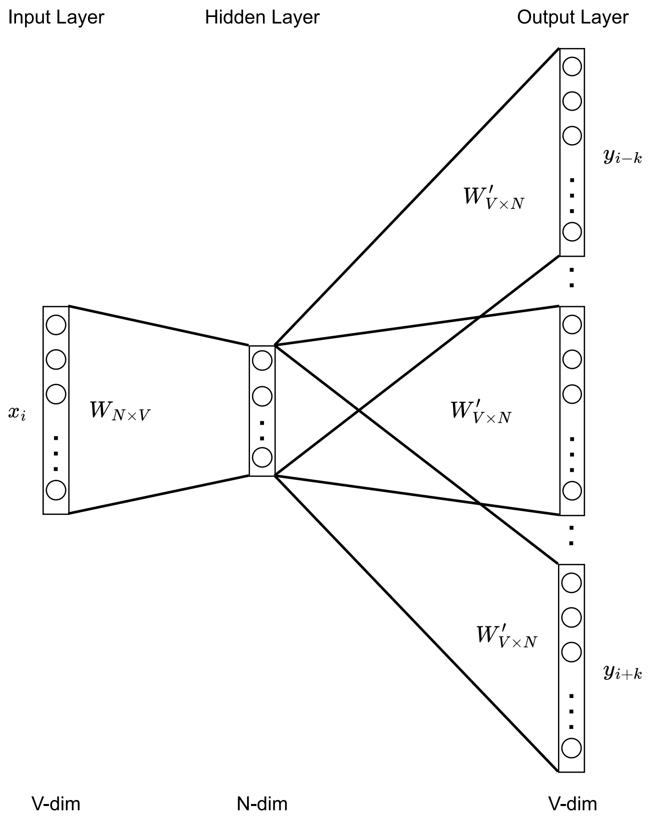

Fig. A.10. General Skip-Gram model.

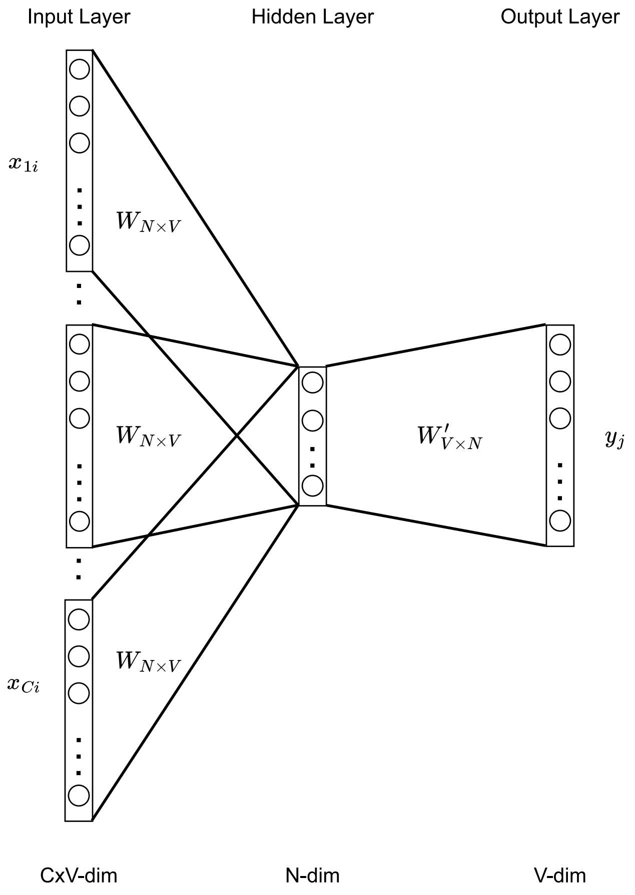

Fig. A.11. General CBOW model.

of w in vocabulary. Vector $\widetilde{V_w}$ is changed to word vector V_w by the transform matrix $\boldsymbol{W}$. Matrix $\boldsymbol{W}$ is trained such that words frequently appear in similar contexts are transformed into similar word vectors. We have two preferred ways to train matrix $\boldsymbol{W}$ is *Skip-Gram* and *Continuous Bag of Words* (CBOW) (Mikolov et al., 2013).

In Skip-Gram, the distributed representation of the focus word as the input to predict the context words as the output from the neuron network was applied for training. Therefore, each training sample consists of one focus word and one relevant context word. A scenario of deciding context words is by choosing k preceding words and k following words with respect to the focus word within a document or tweet in our problem. Mikolov et al. (2013) introduced one hyper-parameter which is the dimension of the embedded vector N. In general, there are around $2k$ training samples for one focus word fed to train the Skip-Gram neuron network.

In Fig. A.10, $x_i \in \mathbb{R}^V$ denotes the one hot vector for the focus word in the sample. $y_{i-k+m}, m = 1, 2, \ldots, 2k, m \neq k$ denote the embedded one hot vector for context words. The computing process inside the Skip-Gram model is described by the following equations.

$$\begin{cases} \upsilon = \boldsymbol{W} x_i & \text{(a)} \\ z = \boldsymbol{W}' \upsilon & \text{(b)} \\ \hat{y} = Softmax(z) & \text{(c)} \end{cases} \tag{A.4}$$

In Eq. (A.4), $\upsilon, z, \hat{y}$ represent the embedded focus word vector in hidden layer, the score vector and the probabilities turned from the score vector respectively. As the Skip-Gram models treat each context word equally, $\boldsymbol{W}$ and $\boldsymbol{W}'$ are learned by minimizing the following loss function.

$$L = \sum_{m=0, m\neq k}^{2k} H(\hat{y}, y_{i-k+m}) \tag{A.5}$$

In Eq. (A.5), $H(\hat{y}, y_{i-k+m})$ is the cross-entropy function between the probability vector $\hat{y}$ and one hot vector y_{i-k+m}. Finally, the matrix $\boldsymbol{W}$ is employed as the transfer matrix to generate the word-embedded vector.

In contrast to the Skip-Gram models, distributed representations of C context words $x_m i, m = 1, 2, \ldots, C$ are treated to generated focus word y. Generally, the CBOW model described in A.11 consists of C context words, two hyper-parameters V and N as mentioned in the Skip-Gram section.

The dimension of the hidden layer and output layer remain the same. Only the dimension of the input layer and the calculation of the hidden layer change. The computing process inside the CBOW model is described by the following equations.

$$\begin{cases} \upsilon_{mi} = \boldsymbol{W} x_{mi}, m = 1, 2, \ldots, C & \text{(a)} \\ \hat{\upsilon} = \dfrac{\upsilon_{1i} + \upsilon_{2i} + .. + \upsilon_{Ci}}{C} & \text{(b)} \\ z = \boldsymbol{W}' \hat{\upsilon} & \text{(c)} \\ \hat{y} = Softmax(z) & \text{(d)} \end{cases} \tag{A.6}$$

In Eq. (A.6), $\upsilon_m i, \hat{\upsilon}, z, \hat{y}$ indicate the embedded context word vector x_i in hidden layer, average embedded context word vector, score vector, probabilities turned from score vector respectively. As the CBOW models learn the probability distribution $\hat{y}$ from the true one y which is a one-hot vector, the cross-entropy function is used as the loss function for the models.

$$L = H(\hat{y}, y) = -\sum_{j=1}^{V} y_j log(\hat{y}_j) \tag{A.7}$$

Since y is the one hot vector, L in (A.7) is calculated as below.

$$L = H(\hat{y}, y) = -y_c log(\hat{y}_c) \tag{A.8}$$

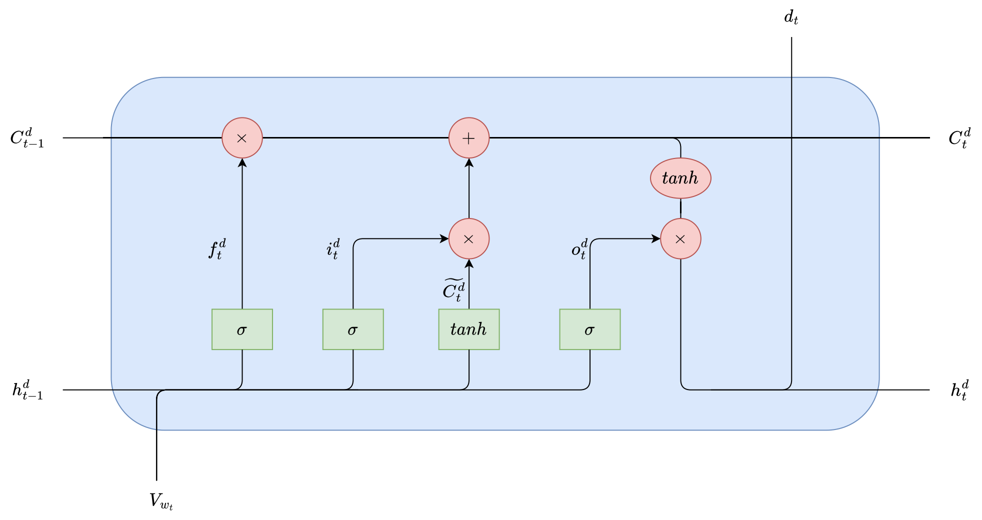

Fig. B.12. A cell architecture in LSTM model (Graves et al., 2009).

In Eq. (A.8), c is the index where the focus word's one hot vector is 1. In contrast to Skip-Gram, in CBOW models the matrix $\boldsymbol{W'}$ is leveraged to embed words in vector.

Appendix B. Long-short term memory (LSTM)

LSTM is formed by n cells (units). Each cell takes input as one-word vector V_{w_k} from the input sequence S. To be more specific, the output of $(n-1)^{th}$ cell is in tune with the $(n-1)^{th}$ input word vector and the partial input of nth cell.

LSTM is a variant of RNN, introduced by Hochreiter and Schmidhuber in 1997 (Graves et al., 2009). LSTMs use a hidden layer as a memory cell instead of a recurrent cell in order to deliver data at a higher performance (Fig. B.12). LSTM model consists of computational blocks repeated over time steps, which helps the LSTM model to have the ability to add or remove information to cell state, regulated by three gates *input gate*, *output gate*, *forget gate*. Controlling flow in LSTM is calculated by the following equations:

$$\begin{cases} i^d_t = \sigma(\boldsymbol{U}^d_i V_{w_t} + \boldsymbol{W}^d_i h^d_{t-1} + b^d_i) & \text{(a)} \\ o^d_t = \sigma(\boldsymbol{U}^d_o V_{w_t} + \boldsymbol{W}^d_o h^d_{t-1} + b^d_o) & \text{(b)} \\ f^d_t = \sigma(\boldsymbol{U}^d_s V_{w_t} + \boldsymbol{W}^d_f h^d_{t-1} + b^d_f) & \text{(c)} \\ \tilde{C}^d_t = tanh(\boldsymbol{U}^d_C V_{w_t} + \boldsymbol{W}^d_C h^d_{t-1} + b^d_C) & \text{(d)} \\ C^d_t = f^d_t \otimes C^d_{t-1} + i^d_t \otimes \tilde{C}^d_t & \text{(e)} \\ d_t = h^d_t = o^d_t \otimes tanh(C^d_{t-1}) & \text{(f)} \end{cases} \tag{B.1}$$

In Eqs. (B.1)(a)–(f), i^d_t, o^d_t, f^d_t, C^d_t, h^d_t denote input gate, output gate, forget gate, internal state, and hidden layer at cell state t (the word vector V_{w_t} as input respectively) in the LSTM cell. d denotes the direction of the LSTM model, it is replaced by f for forward direction (in forward LSTM model) or b for backward direction (in backward LSTM model). Moreover, $\boldsymbol{U}^d_i$, $\boldsymbol{U}^d_o$, $\boldsymbol{U}^d_s$, $\boldsymbol{U}^d_C$ and $\boldsymbol{W}^d_i$, $\boldsymbol{W}^d_o$, $\boldsymbol{W}^d_s$, $\boldsymbol{W}^d_C$ and b^d_i, b^d_o, b^d_f, b^d_C indicate the weight matrices and biases of three gates and a memory cell, in the order given. The $\otimes$, $+$ represent the element-wise multiplication and the addition of two matrices respectively. Fundamentally, the *sigmoid* (σ) activation function helps LSTM determine what new information is stored in the cell state due to the output being between zero and one, so when the output is zero, the information is dismissed, otherwise, it is completely kept. Furthermore, the input gate decides how much information take from the input state, which is word vector V_{w_t} and hidden layer from the previous layer. Similarly, the output gate regulates the information from the cell state revealed appropriately to the output of the hidden state.

Therefore, in order to handle one tweet semantic T_s comprising k words indicated by k vectors V_{w_i}, $i = 1, 2, \dots, k$, Bi-LSTM adds one more LSTM layer, which reverses the direction of information flow. The two layers with opposite directions are denoted as L_f and L_b. Besides, the ith word vector is processed by the ith cell of LSTM L_f and $(k-i+1)^{th}$ cell of LSTM L_b and have the two output denoted as f_i and b_i respectively.

Given two row vectors $x_1 = (a_1, a_2, \dots, a_n) \in \mathbb{R}^n$ and $x_2 = (b_1, b_2, \dots, b_m) \in \mathbb{R}^m$, we denote that notation $||$ is the concatenation operator of these two vector x_1 and x_2 obtain one single row vector, denoted as $\bar{x}$ in $\mathbb{R}^{n+m}$.

$$\bar{x} = (a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m) = x_1 \,||\, x_2 \tag{B.2}$$

The output of Bi-LSTM model is a vector, V_{BiLSTM}, formed by concatenating all k vectors obtained from k pairs of vectors (f_i, b_i).

$$V_{BiLSTM} = (f_1 \,||\, b_1) \,||\, (f_2 \,||\, b_2) \,||\, \dots \,||\, (f_k \,||\, b_k) \tag{B.3}$$

Therefore, the output of this system V_{output} is the concatenation vector from two vector V_{T_m} and V_{BiLSTM}. V_{T_m} is the vector express Tweet Metadata T_m, and V_{BiLSTM} is the output vector of Bi-LSTM.

$$V_{output} = V_{T_m} \,||\, V_{BiLSTM} \tag{B.4}$$

Finally, the vector V_{output} is fed to softmax layer to be classified.

Appendix C. Convolutional neural networks (CNNs)

In CNNs, there are three main hyperparameters controlling the size of the output: the *depth*, *stride*, and *zero-padding*. The *depth* is the number of sliding windows we would like to use, each learning to look for a specific pattern in the input. The *stride* is the step size that modifies the amount of movement over the *tweet matrix* of the sliding window. The *zero-padding* allows us to control the spatial size of the output volumes (most commonly when using it to exactly preserve the spatial size of the input volume so the input and output width and height are the same). If the input volume size is $V_w \times V_h$, the receptive field size of the siding windows is $F_w \times F_h$, the stride with which they are applied is S, and the amount of zero-padding used is P on the border, spatial size $R_w \times R_h$ of the output volume is computed by the following formula.

$$R_d = \frac{V_d - F_d + 2P}{S} + 1, d \in \{w, h\} \tag{C.1}$$

Consider the ith tweet matrix in p user's tweet matrices, denoted as M_{T_i}, has the size of $n \times m_i$ as mentioned above, we use q sliding windows K_j size of $n \times k_j$, $j = 1, 2, \dots, q$ and use the stride is one and no zero-padding, which means V_w, V_h, F_w, F_h, P, S are $n, m_i, n, k_j, 0, 1$

respectively, then we acquire q result matrices, each has the size of $1\times(m_i - k_j + 1)$. Applying ReLU activation and 1-max pooling respectively on each resulting matrix, we obtain q vectors $a_j \in \mathbb{R}$, $j = 1, 2, \dots, q$. After that, we acquire feature map F_i, which is a concatenation vector from q vectors a_j, $j = 1, 2, \dots, q$.

$$F_i = CNN(M_{T_i}) \tag{C.2}$$

Next, each feature map F_i of *tweet semantics* T_{s_i} is concatenated with the corresponding *tweet metadata* T_{m_i} to generate vector V_{t_i}, where *tweet metadata* T_{m_i} contains properties of a tweet, described in Sequence-Based Techniques.

$$V_{t_i} = T_{m_i} \parallel F_i \tag{C.3}$$

Then, vector V_{t_i} is fed to an LSTM model in which each cell takes the vector V_{t_i} as the input at state ith. The ith cell of the LSTM model has the output denoted as d_i.

$$d_i = LSTM(V_{t_i}) \tag{C.4}$$

The output of the model is the concatenation vector V_{LSTM} from all p vectors $d_1, d_2, \dots, d_p$.

$$V_{LSTM} = d_1 \,||\, d_2 \,||\, \dots \,||\, d_p \tag{C.5}$$

In addition, neighborhood information N is also employed, which is represented in the form of an undirected graph where vertices are users and edges are the "following" relationships between them. DeepWalk (Perozzi et al., 2014) was used to generate *Graph Embedding* vector of this undirected graph, denoted as V_{Graph}. In addition, we obtain the concatenation vector V_{Sys} from V_{LSTM} and V_{Graph}.

$$V_{Sys} = V_{LSTM} \,||\, V_{Graph} \tag{C.6}$$

Finally, the vector V_{Sys} is fed to a fully-connected layer to be classified.

References

Alarifi, A., Alsaleh, M., & Al-Salman, A. (2016). Twitter turing test: Identifying social machines. *Information Sciences*, *372*, 332–346. http://dx.doi.org/10.1016/j.ins.2016.08.036.

Alothali, E., Zaki, N., Mohamed, E. A., & Alashwal, H. (2018). Detecting social bots on Twitter: A literature review. In *2018 international conference on innovations in information technology* (pp. 175–180). http://dx.doi.org/10.1109/INNOVATIONS.2018.8605995.

Benamara, F., Cesarano, C., Picariello, A., Recupero, D. R., & Subrahmanian, V. S. (2007). Sentiment analysis: Adjectives and adverbs are better than adjectives alone.. *ICWSM*, *7*, 203–206, URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.1338&rep=rep1&type=pdf.

Beskow, D. M., & Carley, K. M. (2019). Its all in a name: detecting and labeling bots by their name. *Computational and Mathematical Organization Theory*, *25*(1), 24–35. http://dx.doi.org/10.1007/s10588-018-09290-1.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems, vol. 33* (pp. 1877–1901). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cai, C., Li, L., & Zengi, D. (2017). Behavior enhanced deep bot detection in social media. In *2017 IEEE international conference on intelligence and security informatics* (pp. 128–130). http://dx.doi.org/10.1109/ISI.2017.8004887.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. http://dx.doi.org/10.1613/jair.953.

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 and Secure Computing*, *9*(6), 811–824. http://dx.doi.org/10.1109/TDSC.2012.75.

Cresci, S. (2020). A decade of social bot detection. *Communications of the ACM*, *63*(10), 72–83. http://dx.doi.org/10.1145/3409116.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2017). The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th international conference on world wide web companion* (pp. 963–972). International World Wide Web Conferences Steering Committee, http://dx.doi.org/10.1145/3041021.3055135.

Davis, C. A., Varol, O., Ferrara, E., Flammini, A., & Menczer, F. (2016). BotOrNot: A system to evaluate social bots. In *Proceedings of the 25th international conference companion on world wide web* (pp. 273–274). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, http://dx.doi.org/10.1145/2872518.2889302.

Dickerson, J. P., Kagan, V., & Subrahmanian, V. (2014). Using sentiment to detect bots on Twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 620–627). http://dx.doi.org/10.1109/ASONAM.2014.6921650.

Echeverría, J., De Cristofaro, E., Kourtellis, N., Leontiadis, I., Stringhini, G., & Zhou, S. (2018). LOBO: Evaluation of generalization deficiencies in Twitter bot classifiers. In *Proceedings of the 34th annual computer security applications conference* (pp. 137–146). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3274694.3274738.

Erşahin, B., Aktaş, Ö., Kılınç, D., & Akyol, C. (2017). Twitter fake account detection. In *2017 international conference on computer science and engineering* (pp. 388–392). http://dx.doi.org/10.1109/UBMK.2017.8093420.

Fazil, M., & Abulaish, M. (2018). A hybrid approach for detecting automated spammers in Twitter. *IEEE Transactions on Information Forensics and Security*, *13*(11), 2707–2719. http://dx.doi.org/10.1109/TIFS.2018.2825958.

Feng, S., Wan, H., Wang, N., Li, J., & Luo, M. (2021). TwiBot-20: A comprehensive Twitter bot detection benchmark. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 4485–4494). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3459637.3482019.

Feng, S., Wan, H., Wang, N., & Luo, M. (2021). 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* (pp. 236–239). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3487351.3488336.

Ferrara, E., Cresci, S., & Luceri, L. (2020). Misinformation, manipulation, and abuse on social media in the era of COVID-19. *Journal of Computational Social Science*, *3*(2), 271–277. http://dx.doi.org/10.1007/s42001-020-00094-5.

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, *59*(7), 96–104. http://dx.doi.org/10.1145/2818717.

Gilani, Z., Kochmar, E., & Crowcroft, J. (2017). Classification of Twitter accounts into automated agents and human users. In *2017 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 489–496). URL https://ieeexplore.ieee.org/abstract/document/9069085.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 855–868. http://dx.doi.org/10.1109/TPAMI.2008.137.

Kantepe, M., & Ganiz, M. C. (2017). Preprocessing framework for Twitter bot detection. In *2017 international conference on computer science and engineering* (pp. 630–634). http://dx.doi.org/10.1109/UBMK.2017.8093483.

Karpov, I., & Glazkova, E. (2021). Detecting automatically managed accounts in online social networks: Graph embeddings approach. In W. M. P. van der Aalst, V. Batagelj, A. Buzmakov, D. I. Ignatov, A. Kalenkova, M. Khachay, & et al. (Eds.), *Recent trends in analysis of images, social networks and texts* (pp. 11–21). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-71214-3_2.

Khaled, S., El-Tazi, N., & Mokhtar, H. M. O. (2018). Detecting fake accounts on social media. In *2018 IEEE international conference on big data* (pp. 3672–3681). http://dx.doi.org/10.1109/BigData.2018.8621913.

Kudugunta, S., & Ferrara, E. (2018). Deep neural networks for bot detection. *Information Sciences*, *467*, 312–322. http://dx.doi.org/10.1016/j.ins.2018.08.019.

Loyola-González, O., Monroy, R., Rodríguez, J., López-Cuevas, A., & Mata-Sánchez, J. I. (2019). Contrast pattern-based classification for bot detection on Twitter. *IEEE Access*, *7*, 45800–45817. http://dx.doi.org/10.1109/ACCESS.2019.2904220.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. http://dx.doi.org/10.1016/0005-2795(75)90109-9.

Medina-Pérez, M. A., Monroy, R., Camiña, J. B., & García-Borroto, M. (2017). Bagging-TPMiner: a classifier ensemble for masquerader detection based on typical objects. *Soft Computing*, *21*, 557–569. http://dx.doi.org/10.1007/s00500-016-2278-8.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. http://dx.doi.org/10.48550/ARXIV.1301.3781, arXiv URL https://arxiv.org/abs/1301.3781.

Morstatter, F., Wu, L., Nazer, T. H., Carley, K. M., & Liu, H. (2016). A new approach to bot detection: Striking the balance between precision and recall. In *2016 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 533–540). http://dx.doi.org/10.1109/ASONAM.2016.7752287.

Pakaya, F. N., Ibrohim, M. O., & Budi, I. (2019). Malicious account detection on Twitter based on tweet account features using machine learning. In *2019 fourth international conference on informatics and computing* (pp. 1–5). http://dx.doi.org/10.1109/ICIC47613.2019.8985840.

Pasricha, N., & Hayes, C. (2019). Detecting bot behaviour in social media using digital dna compression. In *27th AIAI Irish conference on artificial intelligence and cognitive science*. AICS (Artificial Intelligence and Cognitive Science) 2019, URL http://ceur-ws.org/Vol-2563/aics_35.pdf.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2623330.2623732.

Pranckevicius, T., & Marcinkevičius, V. (2017). Comparison of naive Bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, *5*, http://dx.doi.org/10.22364/bjmc.2017.5.2.05.

Rodríguez-Ruiz, J., Mata-Sánchez, J. I., Monroy, R., Loyola-González, O., & López-Cuevas, A. (2020). A one-class classification approach for bot detection on Twitter. *Computers & Security*, *91*, Article 101715. http://dx.doi.org/10.1016/j.cose.2020.101715, URL https://www.sciencedirect.com/science/article/pii/S0167404820300031.

Sayyadiharikandeh, M., Varol, O., Yang, K.-C., Flammini, A., & Menczer, F. (2020). Detection of novel social bots by ensembles of specialized classifiers. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 2725–2732). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3340531.3412698.

Schlichtkrull, M., Kipf, T. N., Bloem, P., vanden Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, & et al. (Eds.), *The semantic web* (pp. 593–607). Cham: Springer International Publishing.

Shafahi, M., Kempers, L., & Afsarmanesh, H. (2016-12). Phishing through social bots on Twitter. In *2016 IEEE international conference on big data* (pp. 3703–3712). http://dx.doi.org/10.1109/BigData.2016.7841038.

Subrahmanian, V. S., & Reforgiato, D. (2008). AVA: Adjective-verb-adverb combinations for sentiment analysis. *IEEE Intelligent Systems*, *23*(4), 43–50. http://dx.doi.org/10.1109/MIS.2008.57.

Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, *3*(2), 334–340. http://dx.doi.org/10.1016/j.fcij.2018.10.003, URL https://www.sciencedirect.com/science/article/pii/S2314728817300715.

Trifiro, B. M., Paik, S., Fang, Z., & Zhang, L. (2021). Politics and politeness: Analysis of incivility on Twitter during the 2020 democratic presidential primary. *Social Media + Society*, *7*(3), Article 20563051211036939. http://dx.doi.org/10.1177/20563051211036939.

Varol, O., Ferrara, E., Davis, C., Menczer, F., & Flammini, A. (2017). Online human-bot interactions: Detection, estimation, and characterization. *Proceedings of the International AAAI Conference on Web and Social Media*, *11*(1), 280–289. http://dx.doi.org/10.1609/icwsm.v11i1.14871, URL https://ojs.aaai.org/index.php/ICWSM/article/view/14871.

Wei, F., & Nguyen, U. T. (2019). 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* (pp. 101–109). http://dx.doi.org/10.1109/TPS-ISA48467.2019.00021.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-2*(3), 408–421. http://dx.doi.org/10.1109/TSMC.1972.4309137.

Woolley, S. C. (2016). Automating power: Social bot interference in global politics. *First Monday*, *21*(4), http://dx.doi.org/10.5210/fm.v21i4.6161.

Woolley, S. C., & Guilbeault, D. (2017). Computational propaganda in the United States of america: Manufacturing consensus online. *Computational Propaganda Worldwide*, URL http://oxis.oii.ox.ac.uk/wp-content/uploads/sites/89/2017/06/Comprop-USA.pdf.

Yang, K.-C., Ferrara, E., & Menczer, F. (2022). Botometer 101: Social bot practicum for computational social scientists. *Journal of Computational Social Science*, 1–18. http://dx.doi.org/10.1007/s42001-022-00177-5.

Yang, K.-C., Torres-Lugo, C., & Menczer, F. (2020). Prevalence of low-credibility information on twitter during the COVID-19 outbreak. http://dx.doi.org/10.36190/2020.16, arXiv preprint arXiv:2004.14484.

Yang, K.-C., Varol, O., Davis, C. A., Ferrara, E., Flammini, A., & Menczer, F. (2019). Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies*, *1*(1), 48–61. http://dx.doi.org/10.1002/hbe2.115, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbe2.115 URL https://onlinelibrary.wiley.com/doi/abs/10.1002/hbe2.115.

Yang, K.-C., Varol, O., Hui, P.-M., & Menczer, F. (2020). Scalable and generalizable social bot detection through data selection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 1096–1103. http://dx.doi.org/10.1609/aaai.v34i01.5460.