Final version of the accepted paper. Cite as: *Mohd Fazil, Amit Kumar Sah, and Muhammad Abulaish, DeepSBD: A Deep Neural Network Model With Attention Mechanism for SocialBot Detection, IEEE Transactions on Information Forensics and Security, Vol. 16, pp. 4211-4223, 2021.*

DeepSBD: A Deep Neural Network Model with Attention Mechanism for SocialBot Detection

Mohd Fazil, Amit Kumar Sah and Muhammad Abulaish, *SMIEEE*

Abstract—Online Social Networks (OSNs) are witnessing sophisticated cyber threats, that are generally conducted using fake or compromised profiles. Automated agents (*aka* socialbots), a category of sophisticated and modern threat entities, are the native of the social media platforms and responsible for various modern *weaponized information-related* attacks, such as astroturfing, misinformation diffusion, and spamming. Detecting socialbots is a challenging and vital task due to their deceiving character of imitating human behavior. To this end, this paper presents an attention-aware deep neural network model, `DeepSBD`, for detecting socialbots on OSNs. The `DeepSBD` models users' behavior using *profile*, *temporal*, *activity*, and *content* information. It jointly models OSN users' behavior using Bidirectional Long Short Term Memory (BiLSTM) and Convolutional Neural Network (CNN) architectures. It models *profile*, *temporal*, and *activity* information as sequences, which are fed to a two-layers stacked BiLSTM, whereas *content* information is fed to a deep CNN. We have evaluated `DeepSBD` over five real-world benchmark datasets and found that it performs significantly better in comparison to the state-of-the-arts and baseline methods. We have also analyzed the efficacy of `DeepSBD` at different ratios of socialbots and benign users and found that an imbalanced dataset moderately affects the classification accuracy. Finally, we have analyzed the discrimination power of different behavioral components, and it is found that both profile characteristics and content behavior are most impactful, whereas diurnal temporal behavior is the least effective for detecting socialbots on OSNs.

Index Terms—Social network analysis, Socialbot detection, Deep learning, CNN, BiLSTM, Data-driven cybersecurity.

1 Introduction

Web2.0-based Online Social Networks (OSNs) are one of the high impacting human innovations of the 21^{st} century that facilitate their users to express views and thoughts on current affairs and personal life, connect with friends and celebrities, and get updated with the breaking news. OSNs have revolutionized the scope and experience of human communication in the form of real-time information broadcasting. A large fraction of the world population is using one or the other OSN[1]. Though hundreds of OSNs exist, `Twitter` and `Facebook` are among the most popular ones across the globe.

1.1 OSN and Socialbots

OSNs facilitate their users in terms of connectivity, information sharing, knowledge acquisition, and entertainment, but these are not without any repercussions. The real-time message broadcasting, large user-base, open nature, and anonymity have exposed OSNs as a suitable platform for different malicious activities like *trolling*, *astroturfing*, *spamming*, and *fake news*. The malicious and anti-social elements generally perform such activities using fake profiles in the form of bots, human-assisted cyborgs, Sybil, and compromised accounts. Recently, OSN platforms have witnessed emerging threats, having serious repercussions that are much more sophisticated in comparison to the classical cyber threats like *spamming*, *DDoS attack*, and *identity theft*. Among OSN-specific threats, automated profiles (*aka* socialbots) are one of the major enablers of advanced illicit activities like *political astroturfing* [1]. In a seminal work, Varol et al. [2] estimated that approximately $9-15\%$ of the Twitter accounts are bots. Though adversaries can program socialbots for both benign and malicious objectives, recent incidents like interference in the 2016 USA presidential election, Brexit referendum [1], Arab spring [3] have branded them as devils. Socialbots are very deceptive; they mimic human behavior to gain trust in an OSN and then exploit it for illicit activities [4]. As a result, researchers are analyzing different malicious aspects of socialbots [5]. To tackle this growing threat landscape, OSNs are framing account regulation policies and developing in-house socialbot detection methods. At regular intervals, OSN platforms carry out cleanness drive to suspend the malicious accounts. In such a drive, following the exposure of Russian interference in the 2016 USA presidential election, `Twitter` suspended approximately 70 million accounts[2]. Therefore, all such incidents and reports suggest the unprecedented illicit impact of socialbots on the structure and discourse of OSN platforms.

1.2 Background and Motivation

Apart from OSNs, academia and industry researchers are also striving to decode the working environment of socialbots to develop efficient detection methods. However,

- *M. Fazil is currently working as a Research Associate at the Department of Computer Science, South Asian University, New Delhi, India. E-mail: mohdfazil.jmi@gmail.com*
- *Amit K. Sah is currently a Research Scholar at the Department of Computer Science, South Asian University, New Delhi, India. E-mail: amitcsrs@students.sau.ac.in*
- *M. Abulaish (corresponding author) is currently working as a Sr. Associate Professor at the Department of Computer Science, South Asian University, New Delhi, India. E-mail: abulaish@sau.ac.in*

1. https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

2. https://gadgets.ndtv.com/social-networking/news/twitter-said-to-have-suspended-58-million-accounts-in-q4-2017-1885193

as the approaches mature, botherder tunes the socialbots behavior to bypass the underlying detection methods. In the existing literature, supervised machine learning-based methods are the most popular, wherein authors design a number of hand-crafted features from different categories of information like *profile*, *content*, *network*, and *metadata*. Further, machine learning models are trained that classify bots and benign users [6], [7], [8]. Among the existing set of features, most are relatively easy to escape through human-assisted creation of profile attributes and scheduling the socialbot activities using sophisticated randomization algorithms. The bot accounts are generally human-assisted to mix behavioral patterns with benign accounts to bypass the complex features. The existing state-of-the-art feature engineering-based approaches suffer from three major limitations – (i) feature engineering process is a manual and time-consuming task, (ii) it is not generic because a feature engineering-based system can detect only a specific category of socialbots modeled by the defined features, and finally (iii) it includes human biases and limitations. Moreover, the efficacy of a feature-based classification system depends on the set of defined features. Therefore, if features are of low quality, the performance of the classification system will be automatically low.

The second category includes graph partitioning-based approaches for detecting the groups of socialbots operating in a coordinated manner. The graph partitioning-based approaches model the user interaction/connection formation behavior as a network, and partition it into cohesive subgraphs using graph mining techniques. These approaches generally consider only the connection forming behavior of socialbots exploiting network information, but miss the automation cues embedded in textual, temporal, and profile information. Therefore, these approaches fail to detect socialbots who somehow successfully create sufficient attack edges (links) with normal users. Though graph partitioning-based methods overcome the limitations of the feature engineering-based approaches, they do not integrate the cues from different categories of information.

Further, existing literature also has behavior and temporal modeling-based approaches for socialbot detection [9], [10], [11], [12]. Though these approaches are independent of the limitations of the feature engineering-based methods, they model users using only one behavior, either temporal [10], [11], [12], posting type [9], or some other information. As a result, a particular behavior-modeling approach can detect only a specific category of socialbots. Therefore, behavior modeling-based methods are not generalized for detecting different types of socialbots; instead, they detect a particular category, depending on the type of information used to model the users.

Neural networks are showing applicability in diverse applications where other categories of approaches were prevalent. Similarly, to overcome the limitations of the existing categories of approaches, researchers are exploiting the advancements in deep neural networks for socialbots detection. However, it is largely unexplored except few approaches like the ones that are reported in [13], [14], [15]. The deep learning-based models solved the limitations of feature engineering-based approaches because they do not employ manual and hand-crafted feature engineering, and thereby they do not include human biases. Further, if we provide different types of information to deep neural networks, it will extract patterns from each category of information. Like behavior modeling and graph-partitioning approaches, the existing deep learning methods also do not employ all the aspects of user behavior to model the automation. For example, [14] used only profile information; hence, botherders can easily circumvent it through manual creation of profiles. On the other hand, [13], [15] used *temporal* and *content* information, ignoring *profile*, *activity-type*, and *inter-activities* information. Therefore, these approaches can detect only specific classes of socialbots. Further, none of the existing deep learning methods uses the strength of a hierarchical attention mechanism while learning user representations. This study attempts to fill these research gaps and models OSN users through integrating the cues from a comprehensive set of *profile*, *temporal*, *activity*, and *content* information extracted by employing a BiLSTM, CNN, and hierarchical attention-based model.

1.3 Our Contributions

This paper presents a BiLSTM, CNN, and attention-based deep neural network model, `DeepSBD`, to profile users for detecting socialbots on OSNs. To the best of our knowledge, this is the first deep learning-based approach that jointly models a comprehensive set of *profile*, *temporal*, *activity*, and *content* information for user behavior representation. The `DeepSBD` has three levels of novelty – (i) it avoids the tedious task of human-assisted feature engineering, and extracts features using BiLSTM and CNN from the vector-based representation of *profile*, *temporal*, and *activity* information and embedding-based representation of *content* information, respectively at two different levels of granularity, (ii) unlike existing behavior modeling approaches that use only one behavior, `DeepSBD` models user behavior by integrating the cues from a comprehensive set of *profile*, *temporal*, *activity*, and *content* information, and (iii) `DeepSBD` learns user representations by using a novel hierarchical attention-based `CNN-BiLSTM` architecture to identify regularities in the users' behavior.

The existing deep learning methods for socialbot detection solve the limitations of the features engineering-based approaches. However, they do not model all different behavior of a user, disregarding the fact that adversaries can design socialbots for varied purposes. Further, none of the existing deep learning methods uses BiLSTM and attention mechanism that are the state-of-the-art techniques for learning representations through incorporating bidirectional contextual information. The main idea behind `DeepSBD` is to learn contextual representation of each behavioral aspect of the users' personality. To this end, `DeepSBD` models the bio of the users using their basic profile attributes which makes it capable of monitoring and detecting the bots created through the automation tools. `DeepSBD` models the temporal behavior of the users to capture the *diurnal* and *periodic* patterns in their tweet times. The temporal modeling can detect the scheduling algorithm-based operating bots. On `Twitter`, a user can post plain tweet, retweet, quoted tweet, and reply. We encode the tweet posting behavior of the users as a sequence of these four types of tweeting

for detecting socialbots that are programmed to perform a particular pattern of activities. Moreover, tweet content is encoded using embedding vectors to track coherence in user-generated contents. Finally, all behavior vectors are fused to a novel and unified hierarchical attention-based CNN-BiLSTM model to learn different user behavior representations.

In short, the main contributions of this paper can be summarized as follows:

- Developing a novel CNN and BiLSTM-based deep neural network model with attention mechanism, DeepSBD, which integrates the strength of feature engineering, behavior modeling, and deep neural network in a unified manner for socialbot detection.
- User representation learning through joint modeling of *profile*, *temporal*, *activity*, and *content* information using the attention-aware DeepSBD model.
- Performing a detailed empirical evaluation of DeepSBD with seven state-of-the-art socialbot detection methods of different categories.
- Exploring the most commonly used data balancing method, SMOTE (synthetic minority oversampling technique), to judge the efficacy of DeepSBD over imbalanced datasets.

2 Related Work

Boshmaf et al. [4] classified the socialbot detection approaches into two categories – machine learning and graph partitioning-based approaches. Ferrara et al. [16] presented a detailed description of socialbots and discussed their role as a facilitating entity in various malicious activities on different OSNs. The authors further classified the existing socialbot detection methods into four categories – (i) graph-based, (ii) crowd sourcing-based, (iii) feature engineering-based, and (iv) hybrid approaches. Since the crowd sourcing-based approaches are manual, unscalable, and infeasible, they did not receive enough attention. Based on the underlying methodologies, this study groups socialbot detection approaches into four categories – (i) feature engineering, (ii) graph partitioning, (iii) behavior modeling, and (iv) deep learning-based approaches that are briefly reviewed in the following sub-sections.

2.1 Feature Engineering-Based Socialbot Detection

In this category of methods, researchers devise hand-crafted features from user profile, connection network, and textual content, and machine learning models are trained over labeled training datasets. In the early approaches for bot detection [17], researchers trained machine learning models using different categories of simple and straightforward features. Authors in [8], [18] extracted profile-, content- and interaction-based novel features to train machine learning models to segregate spambots from benign users. The authors also performed *feature ablation analysis*. Yang et al. [19] analyzed the socialbots' evasive tactics to circumvent the existing detection methods. Based on the analytical observations, they devised robust features that are difficult to evade. OSN service providers also developed their adversarial detection approaches; e.g., Facebook has Facebook Immune System, which monitors every read and write actions on the database and labels them as malicious or benign action.

For the first time, Dickerson et al. [20] utilized sentiment-driven features along with content, network, and linguistic features to train socialbot classification systems. In one of the most efficient and popular approaches, Devis et al. [6] designed more than 1000 features, including simple to complex ones to train a random forest classifier for labeling OSN users as socialbot or benign. Most of the existing feature engineering-based methods characterize users based on their attributes, but ignore their connecting users and associated communities. Fazil and Abulaish [7], [21] characterized users using four categories of features, based on their attributes, followers' attributes, and communities in the followers' network. As detection approaches mature, socialbots also evolve to bypass the detection systems. Socialbots can easily circumvent the feature engineering-based classification systems by manipulating their behavior as per the defining features of the detection systems. Hand-crafting of features is also a time-consuming and tedious task.

2.2 Graph Partitioning-Based Socialbot Detection

In graph partitioning-based approaches, connection network of the OSN users is modeled as a graph $G(V, E)$, where nodes in V represent the OSN users and edges in $E \subseteq V \times V$ represent their relationships. In this category of approaches, a graph is partitioned into *fake* and *real* regions segregating the benign and malicious users. The links between the fake and benign users are called infiltration or attack edges. The graph partitioning-based approaches assume that sybils or socialbots cannot create sufficient connections with benign users, and researchers exploit this intuition to model and partition the connection network. Viswanath et al. [22] exploited state-of-the-art community detection algorithms to partition the cohesive groups of malicious and benign users. Wang et al. [23] presented the *click stream* model to cluster sybil and benign regions. The authors first analyzed the click transition probability among sybil and benign users, and then created the activity sequence of the users based on their *click stream* data. Next, a similarity matrix was created based on the similarity between activity sequences of every pair of users. Finally, they applied a clustering technique on the similarity matrix to find sybil and non-sybil regions. In [24], the authors first presented an analysis of a socialbot injection experiment. They modeled the OSN users as a graph and applied the Markov clustering to identify malicious groups.

2.3 Behavior Modeling-Based Socialbot Detection

Researchers have also performed behavior modeling using *sequence* and *statistical* analysis of user activities for detecting socialbots. Zhang et al. [12] performed statistical analysis on tweet-times and used Pearson's chi-squared test for detecting automated accounts. They used visualization approach based on *seconds-of-the-minutes* and *minutes-of-the-hours* distribution of tweet-times to evaluate the automation of user accounts. The efficacy of tweet-time for socialbot detection lies in the fact that socialbots are generally activated using some scheduling algorithms. Pan et al. [11] proposed another temporal approach to group socialbots and normal

users. They proposed a *burstiness* parameter to observe user activity behavior within a short period. Chavoshi et al. [10] modeled the sequence of user activities as an activity time-series data and applied *dynamic time warping* for detecting a synchronized set of users considering their activities. The authors first constructed the activity time-series for each user and hashed them into multiple buckets using a lag-sensitive hashing technique. Cresci et al. [9] presented a biological DNA-inspired approach to encode *tweet*, *retweet*, and *reply* activities of a user as a digital-DNA. On the constructed digital-DNA, they applied *longest common subsequence* analysis to detect synchronous behavior among a set of users.

2.4 Deep Learning-Based Socialbot Detection

In the first approach of this kind, Cai et al. [13] jointly modeled the *temporal* and *textual* information of users for detecting socialbots. Ping et al. [15] presented another deep learning-based model for detecting socialbots using joint modeling of users based on their *temporal*, *content* and other information like *hashtags*, *URLs*, and *mentions*. However, these two approaches do not incorporate any information regarding the type of activities, such as *tweet*, *retweet*, or *reply* performed by the users. Further, they neither used profile information nor synchronization in the sequence of time intervals between consecutive tweets. In another approach, the authors of [14] used only profile information with SMOTE for detecting socialbots. Recently, Wu et al. [25] presented an active learning and deep learning-based approach for socialbots detection. They expanded labeled dataset using active learning, and proposed a set of 30 hand-crafted features for user representation. The features are passed through an attention-aware Residual and BiGRU network for classification. However, due to initial user representation using hand-crafted features, [25] suffers from the limitations of the feature engineering-based approaches.

3 Preliminaries

3.1 Long Short Term Memory

Recurrent Neural Networks (RNNs) are a type of artificial neural network to process sequential data for modeling temporal dynamic behavior. RNN consists of recurrent hidden states whose value depends on previous states. Long Short Term Memory (LSTM) is a type of RNN which contains memory block instead of self-connected hidden units. Consequently, LSTM resolves the *vanishing gradient* problem. In LSTM, memory blocks consist of memory cells, that make the neural network intelligent to decide what to remember and what to forget, enabling it to learn long range contextual information. Earlier LSTM cells were made up of three components, namely *input gate*, *output gate*, and *cell state*. Later, Gers et al. [26] added another component, called *forget gate*. Now, a standard LSTM cell is composed of four components. The *input gate* at time t, i_t, controls the flow of value in a cell through a pointwise addition to update the cell state to a new value. The updated information is written to the cell state using equation 1. The output of *forget gate* at time t, f_t, determines the amount of information to be erased, as defined in equation 2. The candidate cell value, $\tilde{C}_t$, based on the current input is determined using equation 3. current cell state C_t using $\tilde{C}_t$, i_t, f_t, and previous cell state C_{t-1} is updated using equation 4. The final output h_t of the LSTM cell is computed using equation 6, where o_t represents the output of the *output gate*, as defined in equation 5. All equations from 1–6 are defined and taken from [27].

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \otimes \tanh(C_t) \tag{6}$$

In these equations, W and b represent the weight and bias vector, respectively. The $\sigma()$ and $\tanh$ represent the sigma and hyperbolic tangent functions, respectively, and $\otimes$ represents an element-wise multiplication. The classical `LSTM` only incorporates historical information from a sequence, completely ignoring the information ahead in the sequence. Bidirectional LSTM (BiLSTM) resolves this issue by incorporating both backward and forward contextual information of a sequence. Moreover, deep RNN learns better low-level feature representation and greater model complexity. Therefore, in this paper, we have used a stacked BiLSTM to incorporate both backward and forward contextual information on sequential data, and to have better low-level feature representation.

3.2 Convolutional Neural Network

Convolution Neural Network (CNN) is a class of artificial neural networks to process data organized in grid format like image and text organized as a matrix. A classical CNN usually consists of two layers – *convolution* and *pooling*. Convolution layer uses a linear mathematical operation, called *convolution*, to extract high-level *feature map* from input tabular data, whereas pooling layer applies *pooling* operation to extract important features from the *feature map*. Although CNN was originally developed to deal with images, it is now extensively used in various natural language processing and text classification tasks due to its robust feature extraction power.

3.3 Attention Mechanism

Attention means concentrating on one or more components while ignoring others. Focusing on a particular unit makes the machine learning models more intelligent. In general, neural network-based classification systems model data as a numeric vector comprising of low-level features, wherein all features are assigned same weight irrespective of their potential of conceptualizing the data. The problem of equal relevance assignment to all features is resolved using an attention mechanism, which assigns variable weights to different features based on their importance. Attention mechanism can be applied at different levels of granularity like *word*, *phrase*, and *sentence* [28]. Further, the attention mechanism enables the neural model to rank features based

on their relevance. The main idea behind this mechanism is to compute a weight distribution on the input features to assign higher values to those features that have higher ranks. The attention layer consists of an alignment layer, attention weights, and context vector. First, the alignment layer computes alignment scores between encoded vector $h = \{h_1, h_2, \ldots, h_n\}$ and a vertex vector v. Thereafter, *softmax* is applied to calculate a probability distribution α_i by normalizing over all the n elements of h, where $i = 1, 2, \ldots, n$, as given in equation 7. A large α_i means that h_i contributes important information to v. The output O of the attention mechanism is a weighted sum of all elements in the encoded vector h, as given in equation 8.

$$\alpha_i = \frac{\exp(h_i' v)}{\sum_{j=1}^{n} \exp(h_j' v))} \tag{7}$$

$$O = \sum_{i=1}^{n} \alpha_i h_i \tag{8}$$

3.4 Word Embedding

The distributional representation of words is the latest and popular method for generating numeric vector representations that incorporate contextual information. In the existing literature, such numeric vectors have shown encouraging results in many natural language processing tasks like rumor detection [29]. The numeric vector representations of textual content can be generated at different levels of granularity, such as *word*, *phrase*, *sentence*, or *paragraph*. Moreover, vector operations like addition, subtraction, and concatenation can be over the vectors of the smaller constituents (word, phrase, or sentence) to get the vector representation of the larger constituents (sentence, paragraph, or document). In an early approach, Bengio et al. [30] presented a feed-forward neural model for joint learning of both the word representation and statistical language model. In a seminal work, Mikolov et al. [31] presented a computationally efficient approach, `word2vec`, to learn the word representation from an unlabelled corpus using two different representation models – (i) continuous bag of words, and (ii) skip-gram. In another widely accepted method, Pennington et al. [32] presented `GloVe` (Global Vector) for word representation which is based on a weighted least square model fitted on global words co-occurrence count. In the distributed representation, each dimension of a word embedding represents a latent concept based on the word's co-occurrence with other words in the corpus. In this paper, we have used the freely available pre-trained 200-dimensional `GloVe` word vectors, trained on a corpus of 2 billion tweets [32].

4 Proposed Approach

This section presents an architecture of the proposed `DeepSBD` model, which consists of four layers, namely (i) input layer, (ii) low-level feature representation layer, (iii) user representation layer, and (iv) output layer, as shown in figure 1. A detailed description of these layers is presented in the following sub-sections.

4.1 Input Layer

Since a user is generally characterized in the real-world based on identity and behavior, `DeepSBD` models OSN users using both profile and behavioral characteristics. To this end, the input layer of `DeepSBD` is used to read profiles, tweets, and related metadata information to learn user representations, as described in the following sub-section.

4.2 Low-level Feature Representation Layer

In the real-world, individuals/communities differ from each other in terms of physical characteristics and behavior. This difference in identity and behavior also persists on OSNs. The users within a community are generally cohesive in terms of physical and behavioral characteristics. Likewise, on OSN, socialbot and benign users differ in terms of profile information, status posting behavior, and topical inclination. The low-level feature representation layer extracts fine-grained features from *profile*, *temporal*, *activity*, and *content* information using neural network techniques that represent users' behavior. A detailed description of learning different behavior representations of users is presented in the following sub-sections.

4.2.1 Profile Representation

Users have different profile information on OSN depending on real-world identity. However, socialbots do not have any real-world identity and use bogus profile information like user name, handle, bio description, created either manually or automatically. To evade the detection systems, botherder generally creates profiles manually to make them look real. On the other hand, automatically created socialbots show regularity in their profile information [14]. The existing literature has used profile information in both feature engineering and deep learning-based approaches for detecting socialbots [14], [17]. Therefore, in order to detect socialbots with regularity in their profile information, `DeepSBD` models users' identity-related information using 12 basic features – *status count*, *followers count*, *friends count*, *favorites count*, *listed count*, *account age (in days)*, *follower rate*, *friend rate*, *status rate*, *followers to friends ratio*, *default profile status*, and *geo enabled or not*. The *follower rate* represents the number of followers gained by a user per day. Other average-related features are also computed on per day basis. We have used simple features that can be directly extracted from the user profiles (or with little computation) to avoid manual complex feature crafting. The profile feature vector P_u is passed to an attention-based two-layers stacked BiLSTM to learn the low-level profile-based representation $\mathcal{P}_u$.

4.2.2 Temporal Behavior Representation

While designing socialbots, the activity times are generally determined using some scheduling algorithms. The activity times are not completely random, rather they follow a certain distribution. Therefore, automated accounts possess a certain level of regularity in the activity timings [12], [10]. In contrast, activity times of human beings are generally random and unpredictable. In order to observe the

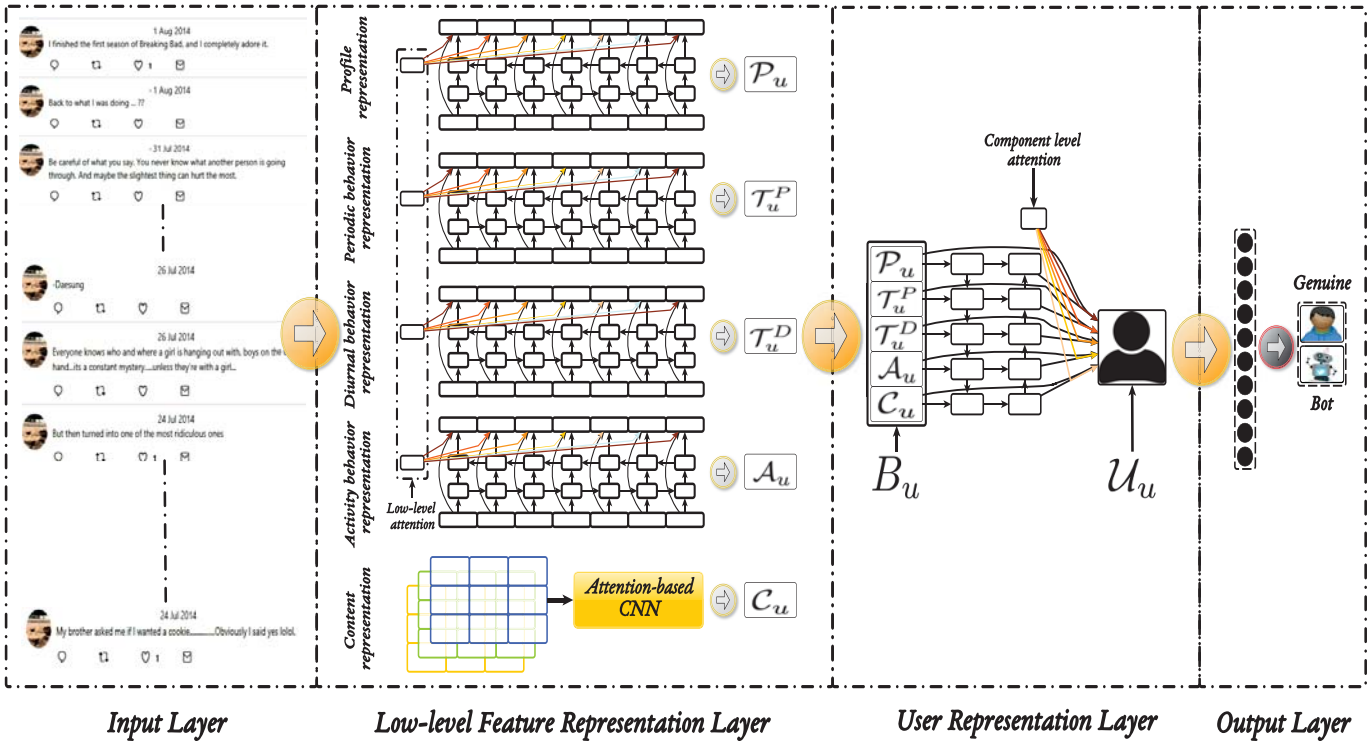
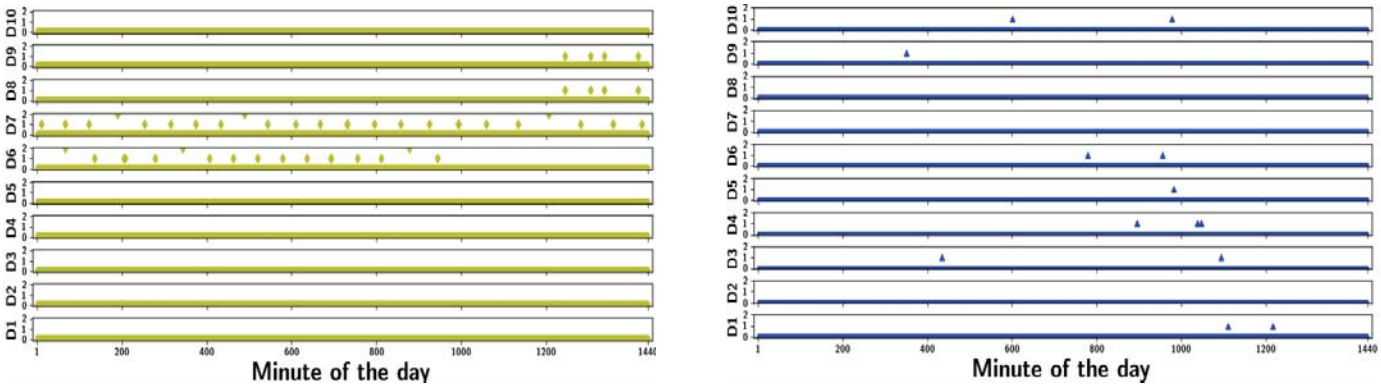


Fig. 1: Architecture of the proposed DeepSBD model



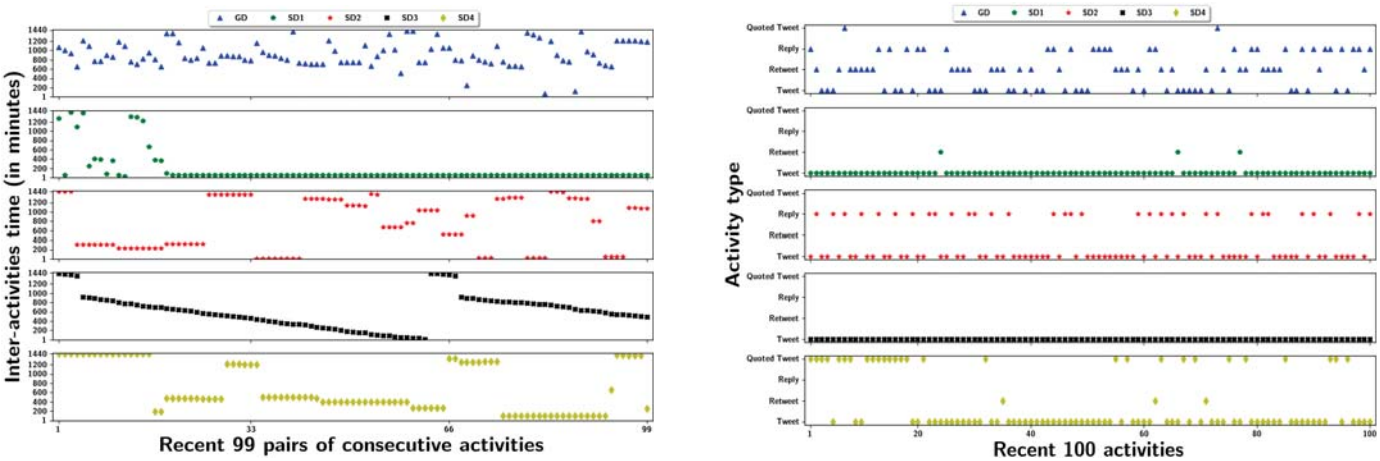
(a) Socialbot (b) Benign user

Fig. 2: Temporal distribution of tweeting activities and corresponding tweet counts over a period of 10 days. Y-axis represents the number of tweets in the respective minute of a particular day

regularity and automation of accounts, modeling of tweet-times is vital. Therefore, we perform a temporal analysis of tweet-times to observe the automation of the user accounts. Generally, bots are programmed to be active – (i) either at a specific time of the day representing the *diurnal pattern*, (ii) or after some specific time-intervals representing the *periodic pattern*. As a result, DeepSBD models two types of the temporal behavior of a user u using temporal information of either the 30 days tweets (if available) or a minimum of 100 tweets, extracted from u's timeline. The diurnal and periodic temporal modeling, described in the following paragraphs, uses the temporal information of all four types of tweeting activities – *plain tweet*, *retweet*, *quoted tweet*, and *reply*.

Diurnal Behavior Representation

As discussed earlier, bots can be programmed to be active at a specific time of the day, representing the *diurnal pattern* [12]. We model such bots using the temporal information of their tweeting activities. To this end, we divide the Twitter timeline of a user u in days, wherein each day $d \in D$ is divided into a sequence of time-intervals of 1 minute, say $\tau = \{\tau_1, \tau_2, \ldots, \tau_{1440}\}$ because 1 day=24×60 minutes, where $|D|$ represents the day count between the first and last crawled tweets of u. Further, we assign the number of *tweeting activity* by u in a particular minute of the day d to the respective minute of the day vector d_u. For example, if u performs 2 retweet activities, posts 1 tweet, and replies to a tweet within a minute between $1:30$ AM to $1:31$



(a) A distribution of *inter-activity time* of a sequence of 100 tweets

(b) A distribution of a sequence of 100 activities

Fig. 3: Inter-activity time and activity sequence behavior of a set of four socialbots (one from each of the four socialbot datasets) and a genuine user

AM, i.e., 91^{st} minute of the i^{th} day d^i, then 91^{th} index of the i^{th} day temporal vector d^i_u is assigned a value of 4. Finally, diurnal temporal behavior representation of d^i_u is $\{dv^i_1, dv^i_2, \ldots, dv^i_{1440}\}$, where dv^i_1 represents the number of activities performed by u at the first minute of d^i. Similarly, the diurnal temporal vector is constructed for each of the 30 days if the tweets exist; otherwise, a zero vector is generated. Finally, we concatenate the *diurnal temporal vector* of each day of u to create the final *diurnal temporal vector* D_u. Thereafter, D_u vector is passed to an attention-based two-layers stacked BiLSTM to extract diurnal behavior-based low-level features $\mathcal{T}^D_u$.

The diurnal temporal behavior of a randomly sampled socialbot and benign user, along with the number of tweeting activities performed in every minute of a day over 10 days, is shown in figures 2(a) and 2(b), respectively. In both figures, X-axis represents the minute of the day and Y-axis represents the number of tweeting activities performed in the particular minute of the day. In these figures, we visualize the tweet count only for a maximum of 2 tweets within a minute due to the visibility issue. It can be observed from figure 2(a) that the socialbot shows a diurnal pattern and gets activated to perform tweeting activity at some particular minutes of the day. Further, socialbot also performs 2 activities within a minute on multiple occasions, like three times on day 6 which is suspicious. On the other hand, figure 2(b) reveals that the benign user is considerably random in the temporal distribution of tweet-time and does not show any diurnal behavior.

Periodic Behavior Representation

The bots can be programmed to be active at a fixed interval of time with some irregularities, representing a periodic pattern in inter-tweets time [10]. In this section, we model such behavior of users to segregate socialbots from benign users. To this end, we compute *inter-activity time* between every pair of consecutive tweets. In this behavior modeling, we consider the time-stamp sequence of all the four types of tweets without grouping. Given a user u, if T_1 and T_2 represent the posting times of tweets t_1 and t_2, respectively, then $\Delta T_{(1,2)} = T_2 - T_1$ represents the *inter-activity time* between t_1 and t_2. Similarly, we compute ΔT between every pair of consecutive tweets. The set of all *inter-activity time* over a set of N tweets of u is represented using ΔT_u. Thereafter, ΔT_u is given to an attention-based two-layers stacked BiLSTM to extract the periodic behavior-based low-level features $\mathcal{T}^P_u$. We investigate the *inter-activity time* distribution of socialbots and benign users and observe very contrasting behavior. Figure 3(a) shows the *inter-activity time* distribution of a set of four randomly selected socialbots, one from each of the four socialbot datasets, and one benign user. In this figure, the lower four sub-plots represent the socialbots' behavior and the uppermost sub-plot represents the benign user's behavior. The X-axis and Y-axis represent the pair of activities and inter-activity time, respectively. The analysis of the lower four sub-plots reveals that the four socialbots show certain periodic patterns depending on the automation strategy designed by their masters. It can be observed from the second sub-plot at the bottom of figure 3(a) that the second socialbot shows a strong periodic pattern with its inter-activity time distribution, showing a decaying trend. This figure also shows that the third socialbot uses some complex scheduling algorithm to imitate random behavior, but its *inter-activity time* distribution still shows a moderate pattern though not strong. In contrast, the uppermost sub-plot of this figure reveals that the benign user does not show any periodic pattern in posting. All these analyses reveal the use of scheduling algorithms by the socialbots.

4.2.3 *Activity Sequence Behavior Representation*

In OSNs, different socialbot perform different activities as per the necessity of their masters (botherders). There are bots who are injected to retweet, whereas some reply to queries (generally in the case of benign bots). Though socialbots can perform every possible activity like *tweet*, *retweet*, and *reply*, they follow certain distribution regarding the type of tweeting activity to perform next. Therefore, sequence of

activities is a good indicator for socialbot detection [9]. The DeepSBD models the activity sequence behavior of a user using all four activities – *tweet*, *retweet*, *reply*, and *quoted tweet*. We select 100 tweets for each user to encode their activity sequence behavior. An equal number of activities are selected for each user to ensure that the length of encoded behavior representation for all users is equal. In the activity sequence representation, each of the four activities is encoded using a unique number – *plain tweet* by 0, *retweet* by 1, *reply* by 2, and *quoted tweet* by 3. For example, given the recent 100 activities from the timeline of a user u, its activity-based behavior can be encoded as $\{0_1, 1_2, 1_3, 0_4, \ldots, 2_{100}\}$, where 1_3 represents that the 3^{rd} activity by u is a *retweet*. We also perform a visual analysis of activity sequences to identify the differences in the behavior of socialbots and benign users. Figure 3(b) represents the example activity sequence of a set of four randomly selected socialbots (one from each socialbot dataset) and one benign user. In this figure, the lower four sub-plots represent socialbots' behavior, whereas the uppermost sub-plot represents the benign user's behavior. This figure shows that the activity sequences of socialbots follow a certain pattern, wherein the second and fourth socialbots only post tweets and do not perform other activities. An interesting observation from the third sub-plot of the bottom is that the corresponding socialbot periodically posts a few tweets followed by a reply. In contrast, the activity sequence of the benign user shows very random behavior. Further, the encoded activity sequence of user u is given as an input to an attention-based two-layers stacked BiLSTM to learn a low-level activity sequence-based feature representation $\mathcal{A}_u$.

4.2.4 Content Behavior Representation

The existing literature has used the regularities in tweet content for socialbot detection [7]. The DeepSBD extracts pattern from intra- and inter-tweets content by encoding tweets in a matrix form using GloVe word-embedding. Given that a user u has posted N tweets, say, $t_1, t_2, \ldots, t_N$, first, tweets are sorted in chronological order. Thereafter, each tweet t_i is tokenized using blank space and each token (word) $w \in W$ is replaced with the respective distributional representation of word of dimension d using GloVe vectors. As a result, i^{th}tweet t_i of u is represented as a matrix $m_{ui} \in \mathbb{R}^{|W| \times d}$. Similarly, each of the N tweets of u is modeled as a 2D word-embedding matrix. Finally, all the matrices are concatenated and represented as a 3D matrix $M_u \in \mathbb{R}^{N \times |W| \times d}$, which is passed through a deep CNN containing 2 max pooling layers, placed after the 2^{nd} and 4^{th} convolutional layers. Each convolutional layer contains k filters $F = f_1, f_2, \ldots, f_k$, each of size $s \times s$. In this work, we have organized tweets as a 3D matrix to learn both intra- and inter-tweets regularities. The output of CNN is passed through an *attention layer* to learn content-based low-level contextual feature representation $\mathcal{C}_u$.

4.2.5 Low-level Attention

The previous sub-sections have presented how different low-level behavior representations (fine-grained features) can be learned. However, not every feature of a particular behavior representation is equally important. Therefore, we use the attention mechanism to assign a variable weight to all low-level features of a particular behavior representation depending on their contextual importance. For example, profile vector P_u of u; if hidden state representation of a feature $f \in P_u$ is h_f, then it is passed to a dense-layer to learn its hidden representation h'_f, as given in equation 9, where w and b represent the weight and bias, respectively. Thereafter, similarity is calculated between h'_f and a vertex vector v_f which represents the importance of $f \in P_u$. We also compute the normalized importance score of f using equation 10. The feature-level context vector v_f is randomly initialized and jointly learned during the training process [28]. Finally, attention-aware representation of a particular behavior, say profile, is learned and represented as $\mathcal{P}_u$. It is computed as a weighted sum of the hidden representation of each feature $f \in P$, as given in equation 11. Similarly, attention-aware representations, $\mathcal{T}_u^D$, $\mathcal{T}_u^P$, $\mathcal{A}_u$, and $\mathcal{C}_u$ for the remaining four behavioral aspects of u are computed.

$$h'_f = \tanh(wh_f + b) \tag{9}$$

$$\alpha_f = \frac{exp(h'_f v_f)}{\sum_f exp(h'_f v_f)} \tag{10}$$

$$\mathcal{P}_u = \sum_f (\alpha_f h_f) \tag{11}$$

4.3 User Representation Layer

This section first combines the low-level feature representations learned in the previous section such that $B_u = \{\mathcal{P}_u, \mathcal{T}_u^D, \mathcal{T}_u^P, \mathcal{A}_u, \mathcal{C}_u\}$. To assign a weight to each of the five behavior components $B_u^c \in B_u$, we also apply a component-level attention using equations 12 and 13, where v_c represents the component-level context vector. The final representation $\mathcal{U}_u$ of u is the weighted sum of the five components, as given in equation 14.

$$B_u^{c\prime} = \tanh(wB_u^c + b) \tag{12}$$

$$\alpha_c = \frac{exp(B_u^{c\prime T} v_c)}{\sum_\beta exp(B_u^{c\prime T} v_c)} \tag{13}$$

$$\mathcal{U}_u = \sum_c \alpha_c B_u^c \tag{14}$$

4.4 Output Layer (Bot Classification Layer)

The $\mathcal{U}_u$ is the higher-level user representation learned from the low-level representations. At the end, DeepSBD passes $\mathcal{U}_u$ through a dense layer, followed by the *softmax* function to classify u as either socialbot or benign user.

TABLE 1: Dataset statistics

Dataset	Number of User Accounts	
	Original	Final
SD_1	991	991
SD_2	3457	2107
SD_3	464	461
SD_4	248	248
SD_5	5203	3807
GD	3474	1973

TABLE 2: Parameter Details

Parameter	Value
Number of layers in stacked BiLSTM	2
BiLSTM Memory cells	256
Number of CNN layers	6
Number of CNN filters	256
Filter size	3×3 (first three layers) and 2×2 (last three layers)
Max pooling operation	3×3 (after three layers) and 2×2 (after last layer)
Number of neurons in dense layer	1024
Dropout value	0.5
Optimizer	Adam
Number of neurons in softmax layer	2

5 Experimental Setup and Results

5.1 Dataset Preprocessing

We evaluate DeepSBD over five datasets constructed from two main datasets out of which one is a benchmark dataset and second is generated by us. The benchmark dataset is provided by [33], which contains 4 datasets, namely *social spambots #1* (SD_1), *social spambots #2* (SD_2), *social spambots #3* (SD_3), and *genuine accounts* (*GD*). The DeepSBD is evaluated over the three socialbot datasets separately to observe its efficacy over different types of socialbots. We also constructed a dataset through a socialbots injection experiment on Twitter between January 6, 2016 to February 2, 2016. A detailed description of the injection process and corresponding statistical results can be found in one of our earlier works [34]. In the experiment, we injected 98 socialbots who were followed by 2907 users consisting of two categories of the followers. The first category includes users who followed back the socialbots after being followed by them, whereas the second category includes users who initiated the connection with the socialbots (i.e., they followed the socialbots first). Out of these 2907 users, 1248 and 1659 are related to the first and second categories of the users, respectively. In this paper, we are concerned only with the second category because established annotation methods have considered only the second category of users as probable bots [35]. Finally, we chose a set of 248 users who got suspended by Twitter as the fourth dataset SD_4. The existing literature has three categories of approaches to create the ground truth of bots. In the first line of approaches, turkers are hired to manually annotate a profile as a bot or human using certain guidelines based on some behavioral characteristics, as used by the authors who provided the benchmark dataset. The second category of approaches creates honeypot profiles to attract users and further label them as bots [35]. Finally, the third category of methods labels a profile as a bot, if suspended by the underlying OSN [36]. Based on the second and third methods [35], [36], we labeled 248 users of SD_4 as bots because these users first created connection with the injected socialbots (second method) and thereafter Twitter suspended them (third method). Moreover, to evaluate the robustness of DeepSBD against different types of socialbots, we aggregate all the four socialbot datasets to create the fifth evaluation dataset SD_5. We have done it because most of the existing bots detection approaches show good performance in detecting a particular category of socialbots, but fail on other categories of socialbots. In each of the five evaluation datasets, we selected the negative class users (benign) from genuine dataset *GD*. The final version of evaluation datasets includes only those users who have more than or equal to 100 tweets. In the experimental evaluation, DeepSBD uses 100 tweets from each user. Table 1 presents a brief statistic of all five socialbot datasets and one genuine dataset, where the *original* column represents the number of users in the provided version of the dataset and *final* column represents the number of selected users for evaluation who have more than 100 tweets.

5.2 Training Details

For experimental evaluation, we used 80% data for training and remaining 20% data for the validation purpose. We used a two-layers stacked BiLSTM with 256 memory cells at each layer to learn *profile*, *temporal*, and *activity*-based behavior representations. Further, we used a six-layers CNN with 256 filters of size 3×3 at the first three layers, followed by a max pooling operation of size 3×3. At the last three layers, another 256 filters of size 2×2 are used, followed by a max pooling operation of size 2×2. In CNN, we perform 2D operations like images because tweet contents of a user are mapped to a 3D matrix similar to image representation. The 3D representation of content captures better hidden features from both *intra-tweet* and *inter-tweets* representations. The *user representation* layer is connected to a dense layer having 1024 neurons. A dropout operation of 0.5 is performed on the dense layer to reduce the over-fitting effect. The dense layer is followed by a *softmax* layer having 2 neurons for classification, and *adam* is used as an optimizer. All the experiments presented in this paper are conducted on a Linux machine with Intel Xeon processor and 64 GB RAM. Table 2 lists the parameters used in the proposed model. Further, DeepSBD is implemented using Python Keras.

5.3 Performance Evaluation Results

We performed the evaluation of DeepSBD over five socialbot datasets using four standard metrics, namely *detection rate* (*DR*), *precision* (*Pr*), *F-Score* (*F1*), and *accuracy* (*Acc*). The first row of table 3 presents the performance evaluation results of DeepSBD over all five datasets. It can be observed from this table that DeepSBD consistently shows good performance over all datasets, except SD_4. Although DeepSBD does not show significantly good performance over SD_4, it is nevertheless competitive to detect very sophisticated socialbots like the ones present in SD_4. It can also be observed from this table that DeepSBD shows the perfect performance over SD_1 in terms of *Pr*. Overall, DeepSBD shows best performance over SD_1 in terms of all the four evaluation metrics, as shown in the first row of table 3.

5.4 Comparative Performance Evaluation

5.4.1 Comparative Evaluation with Deep Learning-Based Approaches

This section presents the performance comparison of DeepSBD with three state-of-the-art deep learning-based methods [13], [14], [15] and three baseline methods. We implemented the three deep learning-based methods using the parameter settings specified in the respective papers.

TABLE 3: Comparative performance evaluation of DeepSBD with three deep learning and three baseline approaches over five datasets

Approach	SD_1				SD_2				SD_3				SD_4				SD_5			
	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc
DeepSBD	**99.64**	**100.00**	**99.81**	**99.83**	**99.43**	**99.45**	**99.42**	**99.51**	**98.40**	**99.45**	**98.86**	**99.38**	75.86	81.88	**75.34**	**94.60**	**97.10**	98.85	**97.94**	97.40
DNNBD	73.72	77.66	75.63	78.20	79.50	79.62	79.56	79.72	43.38	80.32	56.33	66.48	40.32	**88.49**	55.40	66.17	62.21	67.82	64.89	65.68
DBDM	97.83	100.00	98.82	99.32	98.66	98.00	98.25	98.80	94.27	98.26	95.89	98.54	52.90	61.40	54.10	88.98	94.31	**98.95**	96.49	95.41
DeBD	97.49	97.73	97.59	97.74	98.39	99.10	98.76	99.12	95.54	96.01	95.81	97.92	52.34	48.98	46.17	88.62	94.49	97.11	95.66	94.97
Baseline-1	98.89	97.91	98.93	98.37	97.80	99.12	98.81	98.89	95.00	96.42	95.53	98.76	47.29	69.61	52.22	90.56	95.76	97.84	96.71	95.84
Baseline-2	99.19	98.39	98.79	98.78	98.46	99.30	98.87	99.01	96.41	99.26	97.61	98.76	53.95	65.08	55.79	90.78	94.06	98.90	96.34	95.58
Baseline-3	98.38	96.91	97.64	97.61	98.10	99.06	98.65	98.83	98.34	96.83	97.92	98.97	**78.35**	75.31	74.51	94.05	95.34	98.56	96.84	**99.17**

In [13], Cai et al. presented DBDM to jointly model *temporal* and *textual* information of users for socialbot detection. Similarly, Ping et al. [15] presented another deep learning-based model, DeBD, for socialbot detection by joint modeling of users based on their *temporal*, *textual*, and other information like *mention*, *hashtag*, and *URL*. These two approaches do not incorporate any information regarding the activity sequence behavior of the users. Moreover, they do not consider the profile and inter-tweets temporal behavior to model the automation of users or socialbot detection. In [14], Kudugunta et al. presented another deep neural network-based model, DNNBD, which uses only profile information along with SMOTE for socialbot detection. Additionally, we created three baselines using the DeepSBD model, based on the exclusion of different neural network components to observe their impacts on model performance. It is like an *ablation analysis* to observe the impact of different components over performance. In an *ablation analysis*, a feature/feature vector/component is excluded from the classification model to observe its impact on performance evaluation results. In the first baseline (Baseline-1), we used LSTM in place of BiLSTM without attention mechanism. Second baseline (Baseline-2) used LSTM with the attention mechanism. Finally, third baseline (Baseline-3) excluded the attention mechanism from DeepSBD to observe its impact on its performance.

Table 3 presents the comparative results of DeepSBD with three deep learning and three baseline approaches. In this table, the best performance for each of the five datasets is shown in bold typeface. It can be observed from this table that DeepSBD outperforms the comparison approaches over all five datasets in terms of all four evaluation metrics, except for four instances. Over SD_5, DeepSBD shows poor performance in comparison to DBDM and baseline-3 in terms of *Pr* and *Acc*, respectively. DeepSBD also shows poor performance in comparison to *baseline-3* and DNNBD over SD_4 in terms of *DR* and *Pr*, respectively. During the experiment, we found that the DNNBD model is highly inconsistent in terms of performance when the same experiment is repeated multiple times. Therefore, the average evaluation results for this approach is significantly low in comparison to DeepSBD and other comparative models. Another interesting observation is that over a large dataset like SD_2, all the approaches show comparative performance. Among the baselines approaches, it can be observed that the more a baseline is close to DeepSBD in terms of model configuration (e.g., *Baseline-2* and *Baseline-3*) closer is its result to the DeepSBD.

On investigating the baseline results, it is observed that exclusion of *attention mechanism* has the highest impact on the results over the SD_1, SD_2, and SD_5 datasets. On contrast, exclusion of *attention mechanism* along with the replacement of BiLSTM with LSTM shows the highest impact on the performance over the SD_3 and SD_4 datasets. Following a critical analysis of all the evaluation results, we conclude that the performance of existing methods is dataset dependent and specific to a socialbot category.

TABLE 4: Performance evaluation results of BotOrNot and DeBot approaches

Dataset	Bots Statistic			Evaluation Methods			
				BotOrNot		DeBot	
	Original	Suspended or Deleted	Active	TP	FN	TP	FN
SD_1	991	696	295	254	41	6	289
SD_2	3457	138	3319	3280	39	4	3315
SD_3	464	83	381	338	43	10	371
GD	3472	1011	2461	2410	51	2453	8

5.4.2 Comparative Evaluation with State-of-the-Art Tools

In addition to deep learning-based models, existing literature has various state-of-the-art feature engineering and behavior modeling-based methods like BotOrNot [6] and DeBot [10] which present their approach in the form of API services. Both are well-known methods for socialbot detection. For an account, BotOrNot assigns a bot score between 0 to 5, instead of a binary classification. We used a bot score of 4.0 as the threshold, and a user having the bot score greater than this is labeled as bot. On contrast, DeBot performs a binary classification to label a user as either socialbot or genuine. Further, we verified the status of each user account to filter the suspended and deleted accounts from the four socialbot datasets and one genuine dataset. We did this because Twitter has already brought down the suspended and deleted accounts; therefore, exposed APIs of [6] and [10] can not verify their status. We do not evaluate the efficacy of these two methods on SD_5 because it is the aggregation of all four types of socialbots. A brief statistics of user accounts from the three socialbot datasets and one genuine dataset in terms of *original number of users*, *suspended and deleted users*, and *active users* is given in the second, third, and fourth columns of table 4, respectively. We do not present the statistics and results of SD_4 because all the users of this dataset are already suspended. Table 4 shows that approximately 70%, 4% and 18% of the socialbots from SD_1, SD_2 and SD_3, respectively are already suspended. Further, we verify the status of identified active accounts passing their *user ids* to the exposed APIs of the two methods. Table 4 presents the detection results for both the methods in terms

TABLE 5: Performance evaluation results of DeepSBD with different ratio of socialbots and benign users

Socialbots and Benign Users Ratio	Dataset														
	SD_1			SD_2			SD_3			SD_4			SD_5		
	DR	Pr	F1	DR	Pr	F1	DR	Pr	F1	DR	Pr	F1	DR	Pr	F1
1:1	100.00	100.00	100.00	98.83	99.33	99.09	100.00	100.00	100.00	87.53	97.09	92.01	97.41	97.11	97.26
1:2	99.64	100.00	99.81	97.87	97.18	97.52	99.28	100.00	99.62	78.63	86.25	82.17	97.32	96.24	96.78
1:5	94.41	97.89	96.12	95.43	95.91	95.67	97.96	95.31	96.62	75.86	81.88	78.34	93.63	95.83	94.72

of *TP* and *FN*. It can be observed from the fifth and sixth columns of this table that BotOrNot shows comparative performance and detects a significant number of socialbots as per the defined threshold. However, it is still low in comparison to the deep learning-based approaches. On contrast, the seventh and eighth columns of this table show that DeBot completely failed to detect the socialbots. However, at this point, we cannot ascribe the theoretical foundations of the poor results of DeBot. We also present the results for genuine users in the last row of table 4.

5.4.3 Comparative Evaluation with State-of-the-Art Feature Engineering-Based Approaches

The existing literature has many approaches, which use different categories of information to extract hand-crafted features for learning classification models. The authors in [8] and [18] also used *profile*, *content*, and *interaction*-based hand-crafted features for socialbots detection. The adversaries bypass feature engineering approaches like [8] and [18] by manipulating their behavior as per the designed features. Although DeepSBD uses a given set of information along with other information for modeling the automation and suspicious behavior of users, it does not employ any manual feature engineering process. The DeepSBD uses *profile*, *content*, *temporal*, and *activity* information to observe the automation cues from each aspect of a user activity to monitor different types of socialbots. We implemented both [8] and [18] and trained the Naive Bayes, Decision Tree, and Random Forest classification models. Table 6 presents the underlying evaluation results over the four datasets SD_1, SD_2, SD_3, and SD_5. We did not perform the experimental evaluation over SD_4 because values for most of the required attributes are missing in this dataset. Since Twitter has already suspended the socialbot accounts of SD_4, we could not extract the missing information. It can be observed from this table that except for one instance, DeepSBD performs better than both [8] and [18] over all datasets.

5.5 Results on Different Ratio of Socialbots and Benign Users

Table 1 reveals that the ratio of socialbots and benign users in all five datasets is not balanced. For example, the ratio of socialbots and benign users in SD_1 is approximately 1:2, whereas it is nearly 1:8 in SD_4. Further, in OSNs like Twitter and Facebook, the percentage of socialbots is generally low in comparison to the benign users. In a study, Varol et al. [2] estimated that approximately 9% to 15% of Twitter accounts are bots. Therefore, to investigate the efficacy of DeepSBD on real-network like situation, this section presents evaluation results on different ratios of socialbots and genuine users, viz 1:1, 1:2, and 1:5. We repeated the experimental evaluation over these three ratios of datasets and observed the respective evaluation results in terms of *DR*, *Pr*, and *F1*. The dataset with user ratio as 1:1 has an equal number of socialbots and benign users, whereas dataset with 1:2 ratio has two benign users for each socialbot. Table 5 presents the evaluation results. On investigation, we observe that the performance of DeepSBD is adversely impacted by the ratios of socialbots and benign users, though not significant. Table 5 shows that when we increase the ratio of socialbots and benign users (i.e., when the number of genuine users increases) for all five datasets, the performance of DeepSBD goes down, and it is significant in case of SD_4. Based on these analyses, we conclude that an unbalanced dataset adversely affects the performance of the socialbot detection approaches.

5.6 Behavior Ablation Analysis

This section evaluates the contribution of each of the five behavioral components using *behavior ablation analysis*. In order to observe the impact of a particular behavior, its underlying behavior vector is excluded from the model, and the respective result represents the impact of the behavior. For example, in order to investigate the impact of the *profile* feature, its feature vector is excluded from the model and the performance of the modified model is observed, as shown in the third row of table 7. Similar experiment was repeated for all behavior vectors to observe their impact on the model performance. The *behavior ablation analysis* results are shown in table 7. In this table, the evaluation result corresponding to the behavior component having the highest impact on the performance is shown in bold typeface. It can be observed that *profile* feature shows the highest discriminating power over SD_4, and moderate discriminating power over the remaining datasets. Similarly, *content* feature shows good discriminating power over SD_3 and SD_5, and little impact on the remaining datasets. Likewise, periodic temporal behavior also shows good and moderate performance over different datasets. Among the five behavior components, *diurnal* behavior shows the least impact on the performance of DeepSBD. An interesting observation from this analysis is that different behavioral components show disparate performance over different datasets, emphasizing the significance of using different behavioral features for socialbot detection which is a limitation of the existing behavior modeling and deep learning-based approaches. It also endorses the fact that socialbots are injected in OSNs for distinct reasons, and accordingly their operating behavior and other characteristics are designed by their botmasters. Therefore, using a comprehensive set of behavioral features is essential and effective for modeling different categories of socialbots which is one of the strengths of DeepSBD model.

TABLE 6: Comparative evaluation results of DeepSBD with [8] and [18]

Approach	SD_1			SD_2			SD_3			SD_5		
	DR	Pr	Acc	DR	Pr	Acc	DR	Pr	Acc	DR	Pr	Acc
DeepSBD	**99.64**	**100.00**	**99.83**	**99.43**	**99.45**	**99.51**	**98.40**	**99.45**	**99.38**	**98.22**	98.92	**97.80**
	Naive Bayes											
[8]	95.81	93.09	96.82	97.77	88.42	85.74	93.02	94.11	96.39	89.15	95.33	89.26
[18]	66.50	98.38	76.82	96.33	80.45	92.73	63.84	83.83	79.34	87.57	98.46	87.98
	Decision Tree											
[8]	96.60	97.61	97.07	97.28	94.75	95.67	91.95	94.11	96.06	97.18	97.77	96.82
[18]	96.84	94.92	98.04	96.61	97.27	97.57	92.78	90.90	94.75	95.06	**99.02**	96.72
	Random Forest											
[8]	98.52	97.63	96.77	97.75	98.50	96.21	94.86	91.46	97.37	98.09	97.47	96.24
[18]	99.01	98.46	98. 54	99.21	98.37	98.43	96.04	90.90	94.75	97.86	97.32	97. 59

TABLE 7: Performance evaluation results of DeepSBD for behavior ablation analysis

Model	SD_1				SD_2				SD_3				SD_4				SD_5			
	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc	DR	Pr	F1	Acc
DeepSBD	99.64	100.00	99.81	99.83	99.43	99.45	99.42	99.51	98.40	99.45	98.86	99.38	75.86	81.88	75.34	94.60	97.10	98.85	97.94	97.40
DeepSBD-$\mathcal{P}$	**96.92**	99.58	98.11	97.81	**96.01**	99.62	**97.71**	**97.67**	**91.05**	98.68	94.38	98.35	**50.67**	**47.28**	**44.90**	88.53	94.49	98.98	96.56	95.67
DeepSBD-$\mathcal{T}^D$	98.50	97.43	97.92	98.98	99.55	98.21	98.84	98.77	96.20	97.96	96.92	98.76	71.98	73.99	70.14	93.70	94.90	98.18	96.40	95.76
DeepSBD-$\mathcal{T}^P$	99.49	**97.00**	98.19	99.32	99.75	**96.61**	98.11	96.92	94.40	96.71	95.33	98.76	67.62	59.44	60.05	**86.58**	97.23	**96.79**	96.96	96.02
DeepSBD-$\mathcal{A}$	99.50	98.74	99.07	99.30	98.77	98.85	98.78	98.77	92.10	96.65	**93.83**	**97.96**	63.49	65.24	60.63	92.13	96.37	97.72	96.99	96.19
DeepSBD-$\mathcal{C}$	97.19	97.56	**97.37**	**96.78**	98.46	99.30	98.87	99.01	95.00	**96.42**	95.53	98.14	73.95	75.08	74.51	90.78	**94.06**	98.90	**96.34**	**95.58**

TABLE 8: Training time of DeepSBD and other deep learning approaches

Dataset	Training time (in sec)			
	DeepSBD	DBDM	DeBD	DNNBD
SD_1	311.59327	289.5386	161.6182	11.0073
SD_2	428.5452	397.6422	273.6716	12.7356
SD_3	255.7353	238.9604	107.120	9.8362
SD4	248.5316	218.3477	85.5257	8.7645
SD4	634.9055	564.2776	441.3769	19.9573

5.7 Training Time Analysis

This section investigates the training time cost/epoch (in seconds) of both DeepSBD and existing deep learning-based socialbot detection approaches. Table 8 presents the underlying evaluation results. It can be observed from this table that the training time cost of DeepSBD and DBDM is comparable. Furthermore, both are the best performing approaches in terms of F1-score and accuracy. DNNBD is computationally fast, having a smaller training time cost/epoch because it trains the model using a small set of profile attributes. However, DNNBD is unstable and shows poor evaluation results. Though DeepSBD is comparatively slow, such a small difference in computation cost is not an issue in the current era of GPU and TPU. However, in terms of accuracy, DeepSBD performs better than the SOTA deep learning, feature engineering, and behavior modeling-based approaches. Hence, considering both accuracy and computational efficiency, DeepSBD is better.

6 Conclusion and Future Work

This paper presents a deep neural network-based model, DeepSBD, for detecting socialbots on Twitter. It is an attention-aware deep neural network model, which jointly integrates the cues from four information categories using five behavior representations. Unlike existing approaches which exploit only one or two behaviors, DeepSBD is the first deep learning approach which models user representations using a comprehensive set of profile and behavior information, because socialbots are injected in OSN with different targets and thereby they show different behavioral patterns. DeepSBD jointly models users' behavior using CNN and BiLSTM architectures. It models the *profile*, *temporal* and *activity* information as sequences, which are fed to a two-layers stacked BiLSTM, whereas *content* information is given to a deep CNN. The DeepSBD is evaluated over five real-world datasets and performed better in comparison to the existing state-of-the-art deep learning, feature engineering, and behavior modeling-based methods. Extending DeepSBD for other social media platforms and developing its API for academic and research use are the potential future directions of research.

References

[1] A. Bessi and E. Ferrara, "Social bots distort the 2016 u.s. presidential election online discussion," *First Monday*, vol. 21, no. 11, 2016.

[2] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. of the ICWSM*. Montreal, Canada: AAAI Press, 2017, pp. 280–289.

[3] N. Abokhodair, daisy Yoo, and D. W. McDonald, "Dissecting a social botnet: Growth, content and influence in twitter," in *Proc. of the CSCW*. Vancouver, BC, Canada: ACM, 2015, pp. 839–851.

[4] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "Design and analysis of social botnet," *Computer Networks*, vol. 57, no. 2, pp. 556–578, 2013.

[5] M. Abulaish and M. Fazil, "Socialbots: Impacts, threat-dimensions, and defense challenges," *IEEE Technology and Society Magazine*, vol. 39, no. 3, pp. 52–61, 2020.

[6] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots," in *Proc. of the WWW*. Montreal, Canada: ACM, 2016, pp. 273–274.

[7] M. Fazil and M. Abulaish, "A hybrid approach for detecting automated spammers in twitter," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2707–2719, 2018.

[8] F. Ahmed and M. Abulaish, "A generic statistical approach for spam detection in online social networks," *Computer Communications*, vol. 36, no. 10, pp. 1120–1129, 2013.

[9] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Dna-inspired online behavioral modeling and its application to spambot detection," *IEEE Intelligent System*, vol. 31, no. 5, pp. 58–64, 2016.

[10] N. Chavoshi, H. Hamooni, and A. Mueen, "Debot: Twitter bot detection via warped correlation," in *Proc. of the ICDM*. Barcelona, Spain: IEEE, 2016, pp. 817–822.

[11] J. Pan, Y. Liu, and H. Xiang Liu, "Discriminating bot accounts based solely on temporal features of microblog behavior," *Physica A: Statistical Mechanics and its Applications*, vol. 450, no. 3, pp. 193–204, 2016.

[12] C. M. Zhang and V. Paxson, "Detecting and analyzing automated activity on twitter," in *Proc. of the PAM*. Atlanta, USA: Springer, Berlin, Heidelberg, 2011, pp. 102–111.

[13] C. Cai, L. Li, and D. Zeng, "Detecting social bots by jointly modeling deep behavior and content information," in *Proc. of the CIKM*. Singapore: ACM, 2017, pp. 1995–1998.

[14] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, no. 10, pp. 312–322, 2018.

[15] H. Ping and S. Qin, "A social bots detection model based on deep learning algorithm," in *Proc. of the ICCT*. Chongqing, China: IEEE, 2018, pp. 1435–1439.

[16] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[17] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *Proc. of the SIGIR*. Geneva, Switzerland: ACM, 2010, pp. 435–442.

[18] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *Proc. of the RAID*. Menlo Park, California, USA: Springer Berlin Heidelberg, 2011, pp. 318–337.

[19] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving twitter spammers," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, 2013.

[20] J. P. Dickerson, V. Kagan, and V. S. Subrahmanian, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?" in *Proc. of the ASONAM*. Beijing, China: IEEE, 2014, pp. 620–627.

[21] S. Y. Bhat and M. Abulaish, "Community-based features for identifying spammers in online social networks," in *Proc. of the ASONAM*. Ontario, Canada: ACM, 2013, pp. 100–107.

[22] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *Proc. of the SIGCOMM'10*. New Delhi, India: ACM, 2010, pp. 363–374.

[23] G. Wang, T. Konolige, C. Wilson, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proceedings of 22nd USENIX Security Symposium*. Washington, USA: USENIX Association, 2013, pp. 241–255.

[24] M. Fazil and M. Abulaish, "A socialbots analysis-driven graph-based approach for identifying coordinated campaigns in twitter," *Journal of Intelligent & Fuzzy Systems,*, vol. 38, no. 3, pp. 2961–2977, 2020.

[25] Y. Wu, Y. Fang, S. Shang, J. Jin, L. Wei, and H. Wang, "A novel framework for detecting social bots with deep neural networks and active learning," *Knowledge-Based Systems*, vol. 211, no. 1, pp. 1–16, 2021.

[26] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *Proc. of the ICANN*. Valencia, Spain: IEEE, 1999, pp. 850–855.

[27] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, no. 1, pp. 325–338, 2019.

[28] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. of the NAACL-HLT*. California, USA: ACL, 2016, pp. 1480–1489.

[29] M. Abulaish, N. Kumari, M. Fazil, and B. Singh, "A graph-theoretic embedding-based approach for rumor detection in twitter," in *Proc. of the WI*. Thessaloniki, Greece: ACM, 2019, pp. 466–470.

[30] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 1, pp. 1137–1155, 2003.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computing Research Repository*, vol. arXiv:1301.3781, 2013, version 3.

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of EMNLP*. Doha, Qatar: ACL, 2014, pp. 1532–1543.

[33] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the International Conference on World Wide Web*. Perth, Australia: ACM, 2017, pp. 963–972.

[34] M. Fazil and M. Abulaish, "Why a socialbot is effective in twitter? a statistical insight," in *Proceedings of the 9th International Conference on Communication Systems and Networks, Social Networking Workshop*. Bengaluru, India: IEEE, 2017, pp. 562–567.

[35] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th International Conference on Computer Security Applications*. Texas, USA: ACM, 2010, pp. 1–9.

[36] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proceedings of the International Internet Measurement Conference*. Berlin Germany: ACM, 2011, pp. 243–258.

Mohd Fazil received Master's and PhD degrees in Computer Science from Aligarh Muslim University, Aligarh, India and Jamia Millia Islamia, New Delhi, respectively. While working on this paper, he was a Research Fellow at the Department of Computer Science, South Asian University, New Delhi. His research interests include data mining, machine learning, social computing, and data-driven cyber security, and he has published over 13 research papers.

Amit Kumar Sah received Master's degree in Computer Science from South Asian University, New Delhi. He was the recipient of the SAARC-India Silver Jubilee Scholarship for Master's students. Currently, he is pursuing PhD in Computer Science from the same university, and he is the recipient of the SAARC-India Silver Jubilee Scholarship for PhD students. His research interests include text mining, machine learning, deep learning, and data-driven cyber security.

Muhammad Abulaish received PhD degree in Computer Science from Indian Institute of Technology (IIT) Delhi in 2007. He is currently a Sr. Associate Professor at the Department of Computer Science, South Asian University, New Delhi. His research interests span over the areas of data analytics and mining, machine learning, social computing, and data-driven cyber security. He is a senior member of the IEEE, ACM, and CSI. He has published over 118 research papers in reputed journals and conference proceedings, including 6 papers in ACM/IEEE transactions.