Applied Intelligence (2019) 49:3947–3964
https://doi.org/10.1007/s10489-019-01488-3

Adaptive deep Q-learning model for detecting social bots and influential users in online social networks

Greeshma Lingam[1] · **Rashmi Ranjan Rout**[1] · **D. V. L. N. Somayajulu**[1,2]

Published online: 18 May 2019

Abstract

In an online social network (like *Twitter*), a botmaster (i.e., leader among a group of social bots) establishes a social relationship among legitimate participants to reduce the probability of social bot detection. Social bots generate fake tweets and spread malicious information by manipulating the public opinion. Therefore, the detection of social bots in an online social network is an important task. In this paper, we consider social attributes, such as tweet-based attributes, user profile-based attributes and social graph-based attributes for detecting the social bots among legitimate participants. We design a *deep Q-network* architecture by incorporating a *Deep Q-Learning (DQL)* model using the social attributes in the *Twitter* network for detection of social bots based on updating Q-value function (i.e., state-action value function). We consider each social attribute of a user as a state and the learning agent's movement from one state to another state is considered as an action. For Q-value function, we consider all the state-action pairs in order to construct the state transition probability values between the state-action pairs. In the proposed *DQL* algorithm, the learning agent chooses a specific learning action with an optimal Q-value in each state for social bot detection. Further, we also propose an approach that identifies the most influential users (which are influenced by the social bots) based on tweets and the users' interactions. The experimentation using the datasets collected from *Twitter* network illustrates the efficacy of proposed model.

Keywords Social bot · Deep Q-learning · Twitter · Online social networks

1 Introduction

Online social networking websites are rapidly affected by botnets, which are automated software programs that control social network user accounts with malicious activities [15]. A social botnet is a group of bots which are controlled by a botmaster. The botmaster establishes a social relationship among the social bots and the normal online social networking participants (users) in order to reduce the probability of identification [35]. Like traditional bots (in *Internet* chat), social bots are more common in *Twitter* [22]. A social bot is created with the support of open *APIs* (like *Twitter API*) [33]. Social bots have several applications with normal or malicious intention. Especially, *Twitter* online social network is a suitable platform for the generation of social bots with an intention to create sybil attack, spam distribution and posting of useful information (such as news or blog updates) [43]. Social bots are used to influence other *Twitter* users by posting tweets to degrade the services and manipulate the public opinion [14]. Moreover, the social bots add normal users as their friends and expect a few normal users to follow. The tweets posted by the social bots are displayed on the normal users' home page. However, a few normal users may be attracted by the tweets posted by the social bots (which contain malicious text or malicious *URLs*) [28]. If the normal users establish the relationships with social bots, then the entire social networking community will be affected with untrustworthy information [30]. Most of the existing techniques identify *Twitter* bots based on the *follower ratio*, *URLs* and *incomplete user profile*. For example, a *Twitter* bot can retweet other user tweets and can also follow other users. Moreover, the follower ratio is one of the important attributes to identify a social bot [17].

The traditional botnet detection approaches mainly focus on peer-to-peer networks and botnet-based command

✉ Greeshma Lingam
greeshma243@gmail.com

Rashmi Ranjan Rout
rashmi.iitkgp@gmail.com

D. V. L. N. Somayajulu
somadvlns@gmail.com

[1] Computer Science and Engineering, National Institute of Technology, Warangal, 506004, India

[2] Information Technology Design and Manufacturing (IIITDM), Kurnool, Andhra Pradesh, 518002, India

-and-control protocols [11]. Moreover, these types of approaches fail to detect social bots. Social bots are more common in *Twitter* in order to obtain command-and-control information, such as follower ratio, tweets and *URLs*. Traditional *Twitter* bots can easily be detected as they view the profile page of a user frequently in order to obtain command-and-control information [27]. However, some future bots may avoid their weaknesses by considering the *private message passing feature* provided by online social networks in order to establish the relationship among social bots (which are controlled by the botmaster) [45]. The major limitation that exists with the traditional approaches is the detection of social bots by considering only user profile-based features [12, 41]. Moreover, these types of approaches fail to distinguish legitimate users from the new kinds of social bots. The following are the challenges related to social bot detection in online social networks: (i) due to the continuous growth of online social networking participants (users) and content of information posted every day in social networks, the complexity of online social networks is rapidly increasing for analyzing the behavior of user, (ii) in an online social network, the behavior of user rapidly changes over time. Thus, it is important to extract the information that is needed to evaluate a user based on identifying the malicious behavior of user, (iii) a few malicious users may use tools to create fake accounts or manipulate their influence value and (iv) the user may be influenced by various factors, such as content of information posted in the tweet and behavior of other users. Thus, it is important to distinguish the social bots from legitimate users in online social networks.

To address the above challenges, *Reinforcement Learning (RL)* has been adopted for social botnet detection problem because in an online social network the behavior of a user rapidly changes over time. To detect the social bots, the learning agent has to learn from past experiences to reach a goal state through several episodes. The main objective of *RL* is to obtain an optimal policy (i.e., process of selecting a specific learning action) at each state [31]. Moreover, two different learning strategies are adopted to determine the optimal policies quickly. In the first strategy, each learning agent (i.e., user) learns individually by considering past experiences of another learning agent from a random environment. In the second strategy, each learning agent learns by (frequently) establishing interactions with other learning agents (i.e., by commenting and retweeting on other users' tweets). Several existing reinforcement learning approaches have been proposed [25, 29, 39] to obtain the optimal policy. Q-learning is one of the *RL* techniques. In *Q*-learning, choosing an optimal policy from large number of training samples is a difficult task. Moreover, *Q*-learning needs less number of states in order to converge quickly [42]. To overcome this problem, we consider deep *Q*-learning model by using *Q*-value function (i.e., state-action value function) through a deep *Q*-network, which is an efficient method of learning from high dimensionality data. Hence, in comparison to *Q*-learning, deep *Q*-network has more ability to handle large number of states and can converge quickly when compared to *Q*-learning [18].

In this paper, we consider a set of social attributes, such as tweet-based attributes (i.e., from the content of each user tweet), user profile-based attributes (i.e., from a series of each user weekly tweets) and social graph based attributes (i.e., the users' interaction with their friends and followers) to identify the suspicious behavior of social bots. The objective of this paper is to classify the *Twitter* accounts into two categories, namely legitimate users and social bots. In this paper, we consider each social attribute of a user as a state. The learning agent's movement from one state to another state is considered as an action. For the *Q*-value function, we consider all the state-action pairs in order to construct the state transition probability values between the state-action pairs. In the proposed algorithm, the learning agent chooses a specific learning action with an optimal *Q*-value in each state for social bot detection. We have summarized our contribution as follows:

- A user behavioral analysis model is proposed by (extracting and) ranking social attributes (such as tweet-based attributes, user profile-based attributes and social graph based attributes) in *Twitter* network based on the tweet propagation and user interactions.
- We propose a deep *Q*-network based architecture by integrating deep *Q*-learning model with social attributes for social bot detection based on the *Q*-value function (i.e., state-action value function). Further, an algorithm has been proposed to identify the most influential users (which are influenced by the social bots) based on the tweets and the users' interactions.
- The experiments are conducted on three datasets collected from the *Twitter* network, such as *The Fake Project* dataset [13], *Social Honeypot* dataset [26] and *User Popularity Band* dataset [19].

The rest of the paper is organized as follows: The related work is discussed in Section 2. In Section 3, a *Deep Q-Learning* algorithm has been proposed for detecting the social bots and identifying the most influential users in an online social network. The experimental results are discussed in Section 4. Finally, the conclusion of this paper is presented in Section 5.

2 Related work

In this section, the prior works on presenting vulnerabilities of several social networks, social bot detection and identifying

top-k influential users in online social networks have been discussed.

The social bots in online social networks (like *Facebook* and *Twitter*) have gained more attention recently. Chu et al. [12] have categorized *Twitter* users into three different groups (i.e., human, bot and cyborg) based on their tweeting behavior, account based features and tweet content. Further, the authors have proposed a classification model which includes three major components, (i) an entropy based component used to measure regular tweeting behavior of user, (ii) a spam detection component used to verify whether tweet contains any spam content or not and (iii) account based features to classify the users. Yan [45] has discovered three different types of social botnets (such as appendix botnet, standalone botnet and crossover botnet) which are hidden in *Twitter* network based on dividing graph into small connected components which help to effectively monitor social botnet activities. In their work, the authors have analyzed *Twitter* network by constructing a social network graph in which node represents a *Twitter* user and edge represents the information flow between two connected users. Further, the authors have investigated the size of weakly and strongly connected components for identifying suspicious activities of social bots in *Twitter* network. Zhang et al. [47] have analyzed over thousands of *Twitter* accounts and discussed two new types of social botnet attacks, such as manipulation of user's influence value and spam distribution on *Twitter*. The authors have identified that botmaster constructs a retweeting tree, where the root bot is regarded as spam originator and remaining all other bots only retweet spam content from the parent bot. In botnet-based manipulation of user influence, the authors have found that a few malicious user can manipulate their influence value to attract legitimate users. Further, the authors have presented two countermeasures to protect against the social botnet attacks based on maintaining spam score of each user and identifying the credible users among social bots. In [40], a stegbot detection method in multimedia social network has been proposed to detect stegbots. The authors have analyzed that stegbots can affect the legitimate users by performing malicious activities such as stealing sensitive information (like credit card details and password), phishing and spreading spam content. The authors have also extracted the social attributes (such as image based features, user profile based features and network based features) to distinguish between legitimate users and malicious users (stegbots). Kudugunta et al. [24] have proposed a deep neural network model based on long short term memory (LSTM) architecture. In this architecture, content based features (such as retweet count, number of hashtags and number of mentions) and user metadata based features (such as status count, follower count and default profile) are given as input to *LSTM* for social botnet detection. The authors have also analyzed that considering only tweet based features may not effectively detect the social bots in online social networks.

Freitas et al. [17] have studied social bot infiltration strategies in *Twitter* network. The authors have created social bots in *Twitter* network by performing malicious activities, such as spam distribution, following other users and retweeting other users' tweets. Their work also shows that only 31% of social bot accounts have been detected by *Twitter* network after one month. Further, the authors have addressed a strategy to gain more number of followers and an automated strategy of posting tweets (through program) to avoid bot detection. In [1], a support vector machine-based optimization learning algorithm has been proposed for detecting spam bots based on content-based features, profile-based features and user behavior-based features. The authors have analyzed the most influencing features for detecting the spammers in an online social network. Subrahmanian et al. [36] have proposed to separate bots from other *Twitter* users on a specific topic. The authors have identified additional bots based on the *cosine* similarity between bot and human. Further, the authors have analyzed behavior of social bot based on the hashtag co-occurence, prediction score (higher value more likely to be social bot) and proposed program that could generate social bots by varying number of parameters for social botnet creation. Ashfaq et al. [4] have designed a framework for bot detection using *Bayesian* network classifier model. This model quantifies a belief value (which lies within a range of 0 and 1) to indicate whether a host is acting as a bot or not. Halfaker et al. [20] have summarized *Wikipedia's Immune* system to distinguish social bots from cyborgs (which integrate both human (i.e., manual characteristics) and bot (i.e., automated) behavior). The programmable *Wikipedia* social bots are capable of performing many activities (like spell checker bots) on the website. In [11], a botnet detection approach has been proposed based on the node centrality measures, such as degree centrality, betweenness centrality, eigenvector centrality and pagerank centrality. Further, the authors have adopted self organizing feature map in order to form clusters based on these features and focused on the abnormal behavior of social bots.

In large online social networks (like *Facebook*), Baltazar et al. [5] have analyzed that 20% of normal users accept friend requests with at least one common friend. Moreover, in some social networks (like *Twitter*), establishing social interaction with strangers is one of their features. Boshmaf et al. [8] have proposed a social bot network model on *Facebook* in order to infiltrate the *Facebook* users by creating programmable social bots for two months duration. Later, the authors have analyzed the behavior of users before and after infiltration. Their work shows that up to 80% of online social networks can effectively be infiltrated. Ferrara

et al. [15] have proposed botnet detection approaches based on crowdsourcing based features, graph based features and user based features. The authors have identified two limitations in the crowdsourcing based features, (i) human experts fail to detect fake accounts more accurately, and (ii) revealing the personal information to the human experts lead to privacy issue. Graph based features are taken into consideration to detect sybil accounts by analyzing the social network graph. For social botnet detection, the authors have considered user based features, such as sentimental analysis and content based features. Several existing effective approaches have been proposed for detecting social bots in online social networks [11, 12, 24, 45]. The existing approaches have considered either tweet based features or graph based features for detecting social bots in online social networks. However, by considering either of these features, we can not effectively predict the accuracy of social botnet detection. In this paper, we consider tweet based attributes, user profile based attributes and social graph based attributes to effectively distinguish social bots among legitimate users. Further, we focus on identifying influential users, which are influenced by social bots (termed as influence bots).

Zhang et al. [46] have presented a *True-Top* sybil resilient system for measuring user influence value in *Twitter* network. The authors have analyzed that in *Twitter* network, users usually interact with strangers. Ma et al. [32] have identified that detecting influential users plays a vital role in spreading spam content in online social networks. The authors have observed that centrality measures (such as betweenness, closeness and pagerank) are important to identify how quickly the information can be spread across social networks. Further, the author have proposed *Adjustable Multi-hop Spreading (AMS)* method to measure the user influence. Alshahrani et al. [3] have proposed *D*-hops model, which incorporates degree centrality with multi-hop distance measure for identifying top-k influential users in directed and undirected graph. In [2], the authors have proposed and validated a user centric approach based on four different social attributes (namely social-emotional, socio-psychological, behavior and privacy related attributes) for detecting cyber attacks in online social networks. Further, the authors have analyzed how these attributes have more impact on influencing users in social networks. Wu et al. [44] have presented topic behavior influence tree method based on five features (such as message content, hashtags, replies, mentions and retweets) for identifying influential users in *Twitter* network. In the above mentioned works, the authors have not taken into consideration the impact of social bots for identifying influential users, which are influenced by social bots (termed as influence bots) in *Twitter* network. In this paper, we propose a deep Q-network based architecture by integrating deep Q-learning model with social attributes for social botnet detection based on the Q-value function (i.e., state-action value function). Further, an algorithm has been proposed to identify the most influential users based on the tweets and the users' interactions.

2.1 Reinforcement learning

In *Reinforcement learning* (RL), the learning agent communicates with a random environment in order to increase the performance of the entire system. The random environment rewards the learning agent after executing a finite set of actions (which is termed as reinforcement signal) [31]. The learning agent considers the current reward and the cumulative reward.

Let $S = \{s_1, s_2, ..., s_n\}$ be a set of states and $L^A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ be a finite set of learning actions. The learning agent chooses an action $\alpha_j \in L^A$ and obtains a reward based on the present state $s_i \in S$. Further, a next state $s_{i+1} \in S$ depends on the previous state s_i and the learning action α_j. The process of selecting a specific learning action is defined as a policy. Moreover, the learning agent has to determine an optimal policy at each state s in order to maximize the cumulative reward.

Q-learning is one of the *RL* techniques [18]. The Q function is implemented by using a function approximator (i.e., a neural network or a deep neural network) to estimate the value of the function. The major limitation of Q-learning is that choosing an optimal policy from a large number of training samples is a difficult task. To overcome this problem, Q-learning model is integrated with deep neural networks (which is defined as deep Q-learning model). In this work, a deep Q-learning algorithm has been proposed (which uses a deep Q-network) to extract high dimensional user profile based features for detecting the social bots in online social networks.

2.2 Deep Q-learning

In Q-learning, the learning agent adopts a neural network (defined as a Q-network) with a Q-function, which is represented as $Q(s, \alpha; w)$. The Q-function is determined by considering the parameters, such as state s, learning action α and current reward r. The learning agent chooses a specific action α and obtains a corresponding reward. The parameter w represents the weights associated with each layer in the Q-network at time t. Further, the next state $s' \in S$ depends on the previous state s and the learning action α [42]. Therefore, the Q-function (i.e., state-action function) is given by

$$Q(s,\alpha) = (1-\epsilon)Q(s,\alpha)+\epsilon(r+\gamma[max_{\alpha' \in \alpha(s')}Q(s',\alpha')-Q(s,\alpha)]) \quad (1)$$

where ϵ represents the learning rate (i.e., $0 < \epsilon < 1$) and $\alpha(s')$ represents a finite set of learning actions in next state s'.

Deep Q-network adopts a deep neural network rather than using a Q-network. Three techniques are required to convert Q-network into deep Q-network. Firstly, neural networks are replaced with deep convolutional networks, which help to extract high dimensional features from the input dataset. Secondly, an experience relay is introduced to allow the learning agent to recall and reuse the social interactions from the past experiences. Moreover, the learning agent stores the past experiences (as an experience tuple at a time *'t'* i.e., <state, action, reward, next_state>) into a replay memory. Lastly, a deep neural network predicts the target Q-values, which are used to determine the loss function $Loss(w)$ for each learning action. If only one network is used for estimating the Q-value and the target Q-value, then the result may lead to a feedback loop [18]. Therefore, in deep Q-learning, the target weights (i.e., w) are to be periodically updated. The deep Q-learning model is trained to reduce the loss function $Loss(w)$ from the replay memory which is given by

$$Loss(w) = E\left[(y - Q(s, \alpha; w))^2\right] \tag{2}$$

$$y = r + \gamma\, Q(s^*, argmax_{\alpha^* \in \alpha(s^*)} Q(s^*, \alpha^*; w) w^*) \tag{3}$$

where y represents the target value and w^* represents the updated weight for each iteration in deep Q-network.

3 Deep Q-learning model for detecting social bots and influential users

In this section, we define a set of social attributes for distinguishing the social bots among legitimate users. In addition, we design a deep Q-network architecture, which incorporates the proposed *Deep Q-Learning* algorithm and social attributes from the *Twitter* network for social bot detection. Further, in this section, an algorithm has been proposed to identify the most influential users (which are influenced by the social bots) based on the tweets and the users' interactions.

3.1 Deep Q-network architecture

Figure 1 shows the proposed deep Q-network architecture for detecting the social bots in *Twitter* network. Three different types of social attributes (such as, tweet-based attributes, user profile-based attributes and social graph based attributes) are given as input to the deep Q-network. Firstly, the tweet-based attributes such as syntax, semantic and temporal behavior attributes are extracted from each user tweet (as shown in Fig. 1 as rectangular boxes). Secondly, user-profile based attributes are extracted from a series of each user tweets (with weekly sampling time period) based on the tweeting behavior and the user interactions. Lastly, social graph-based attributes are extracted from the tweets based on the social relationship among participants (users). Therefore, a server (i.e., an interface between an online social network data and deep Q-learning model) is responsible for collecting each participant's (user's) social attributes. For each user, the server (as shown in Fig. 1) stores the collected data in the form of state vector S (which is discussed in Section 3.3), which contains a set of states (i.e., social attributes). Next, for each user the server sends S to deep Q-network. For every user, the social attributes values are collected in each

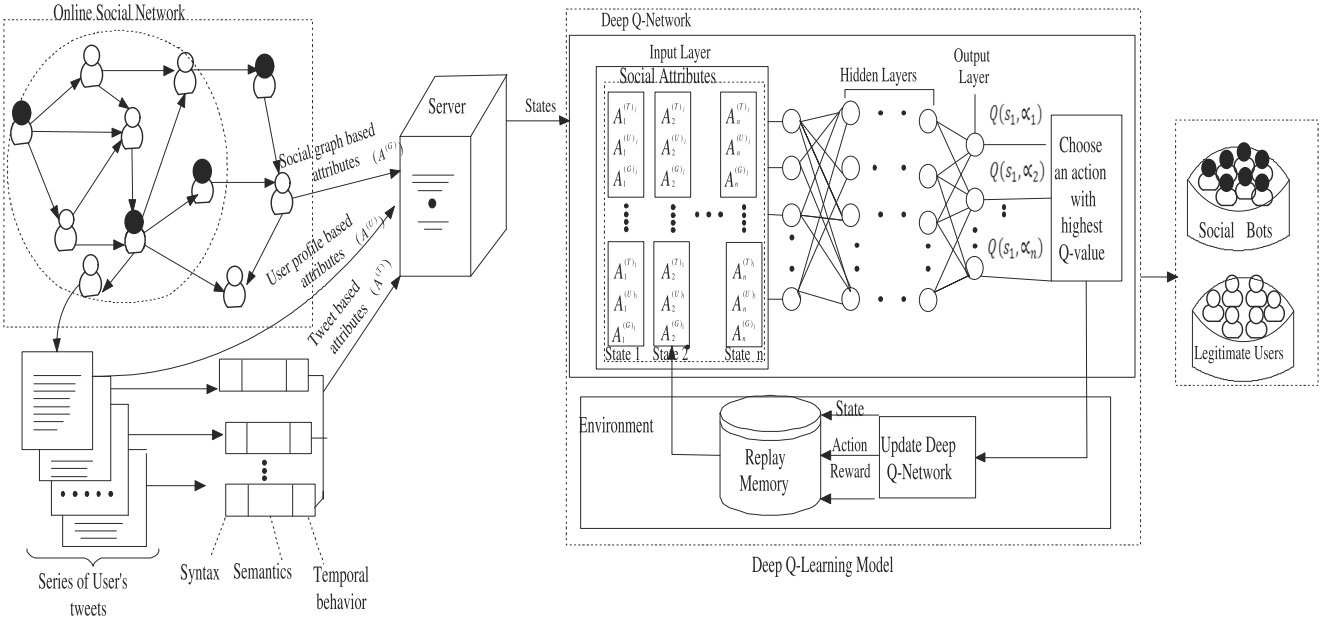


Fig. 1 Deep Q-learning architecture for social botnet detection

Table 1 Overview of Tweet-based Attributes from [1, 15, 36]

Category	Length	Description
Syntax (Sy)	1	Whether the tweet contain any emoticons (e.g.,☹)
	5	Total number of URLs, replies, retweets, mentions and hashtags in tweets
	1	Whether the tweet is socially geo-enabled
Semantics (Sm)	2	Number of positive and negative sentimental words
	1	Identifying the most frequent words or topics tweeted about the user
	1	Computing the user's sentimental score
Temporal behavior (Tm)	1	Timestamp of each user' tweet

time slot (where the total time 't' is divided into l_s time slots and $\sum_{i=1}^{l_s} \tau_i = t$). Later, deep Q-network determines an optimal action for each state. After executing the action, the server decides whether the corresponding user is acting as a social bot or a legitimate user. Finally, after detecting the corresponding user as a social bot, then the server isolates the social bot from the *Twitter* network. Therefore, the system (i.e., deep Q-learning model) is transferred to the next state after a specific action is executed and obtains a reward which is computed by using the reward function (which is discussed in Section 3.3). Moreover, at each time *'t'*, the deep Q-network stores the experience tuple (i.e.,<state, action, reward, next_state>) into a replay memory. Therefore, the proposed deep Q-learning model is used in the proposed architecture that helps to differentiate social bots among legitimate users and identifies the most influential users in the *Twitter* network.

3.2 Classification of social attributes

To address the challenging issue of social bot detection, the social attributes are classified into three categories, such as tweet-based attributes [1, 15, 36], user profile-based attributes [12, 40, 41] and social graph based attributes [11, 40, 45].

3.2.1 Tweet-based attributes

We extract tweet-based attributes from the content of each user's tweet. Hence, tweet-based attributes describe about tweet syntax, tweet-semantics and temporal behavioral features, which are listed in Table 1.

We adopt a lexical normalization technique on *Twitter* data to obtain the individual words (or tokens) [23]. Further, we classify the individual words into either positive or negative emotions based on the user's tweets. Moreover, we need to extract punctuation symbols ('#','?', '!', '....', '.'), special characters ('@', '$', '%') and emoticons (i.e., ☹) in the tweet. Geo-tagged user's tweets gives the information about location of the posted tweets [10]. *Latent dirichlet allocation* is used to identify the most frequent words or topics posted by the *Twitter* users [7]. A sentimental analysis framework and an opinion analysis system are adopted in order to compute a user's sentimental score based on the tweet [9, 37].

Table 2 Overview of User profile-based Attributes from [12, 40, 41]

Category	Length	Description
Tweet Behavior (TB)	2	Total number of tweets and retweets posted by user's (If number of retweets is more than the number of the user's tweets, then the user is most likely to be a social bot)
	1	Posting tweets in several languages (which may be a social bot)
	1	Total number of user's tweets posted per day (If the value is too large then the user may be a social bot)
	1	User session time (If the user session is continued for a long time without any discontinuity for 5-10 minutes, then the user is most likely to be a social bot)
	1	*URL* ratio- $\frac{\|tw_{URL}\|}{\|tw\|+\|tw_{URL}\|}$ (where $\|tw\|$ is the total number of tweets posted by user's and $\|tw_{URL}\|$ is the total number of user's tweets containing URLs)
	1	Hashtag ratio- $\frac{\|tw_{\#}\|}{\|tw\|+\|tw_{\#}\|}$ (where $\|tw_{\#}\|$ is the total number of tweets posted by user's starting with $\#name$)
	1	Mention ratio- $\frac{\|tw_{@}\|}{\|tw\|+\|tw_{@}\|}$ (where $\|tw_{@}\|$ is the total number of tweets posted by user's starting with $@name$)
User Interactions (UI)	2	Number of user's friends and followers (If number of followers is more than the number of friends, then the user is most likely to be a social bot)
	1	Follower ratio- $log\frac{\|followers\|+1}{\|friends\|+1}$
	2	Number of messages and images shared
	1	Number of active days
	1	Number of retweeted tweets
	1	Total number of user's trusted neighbors (follower/friends)
	2	Total number of trusted neighbors with strong and weak ties

The tweet-based attribute vector $A_{p_i}^{(T)}$ of i^{th} participant's j^{th} tweet is represented as

$$A_{p_i}^{(T)} =<< Sy_i^j >, < Sm_i^j >, < Tm_i^j >> \quad (4)$$

3.2.2 User profile-based attributes for behavioral analysis

User profile-based attributes show the behavioral characteristics of the user's. Moreover, some malicious users send friend requests to unknown user accounts or randomly share tweets with other users. In this paper, we extract user profile based attributes from a series of user's tweets with sampling time per-week basis. Moreover, the tweet size is limited up to 140 characters [47]. Hence, we define user profile-based attributes to distinguish social bots among legitimate users based on two aspects, such as tweet behavioral attributes and user interaction attributes. An overview of user profile-based attributes are listed in Table 2.

The user profile-based attribute vector $A_{p_i}^{(U)}$ of participant p_i is represented as

$$A_{p_i}^{(U)} =<< TB_i >, < UI_i >> \quad (5)$$

3.2.3 Social graph-based attributes for tweet propagation

The social graph-based attributes mainly focus on the social relationships among the users. Hence, we define social graph-based attributes, such as clustering coefficient, closeness centrality, betweenness centrality and pagerank centrality for each user [11, 40, 45].

i) *Clustering coefficient:* A social graph $G = (P, L)$, where P represents a set of participants (users) and L represents a set of directed links. A directed link l_{ij} represents the social interaction from a participant p_i to a participant p_j. If $p_i's$ has n links with its neighbors, then the clustering coefficient $CC(P_i)$ is defined as $CC(P_i) = \frac{n}{d_i(d_i-1)}$, where d_i represents the number of neighbors of a participant p_i.

ii) *Closeness Centrality:* Closeness centrality is defined as sum of distance between a participant and all other participants in a social network. Therefore, the closeness centrality of the participant p_i (which is denoted as $C(p_i)$) is defined as $C(p_i) = \frac{1}{\sum d(p_i,p_j):p_j \in P}$.

iii) *Betweenness Centrality:* Betweenness centrality is a measure for identifying the important participants (users) within a social network. The betweenness centrality of the participant p_k (which is denoted as $B_c(p_k)$) is defined as $B_c(p_k) = \sum_{p_i \neq p_k \neq p_j} \frac{\sigma_{p_i p_j}(p_k)}{\sigma_{p_i p_j}}$, where $\sigma_{p_i p_j}(p_k)$ represents the total number of paths from participant p_i to participant p_j passing through an intermediate participant p_k and $\sigma_{p_i p_j}$ represents the total number of shortest paths from p_i to p_j

iv) *Pagerank Centrality:* Pagerank centrality of a participant is defined as the out-degree centrality of participant by establishing social relationships among participants (users) based on their interactions. Therefore, the pagerank centrality of a participant p_i (which is denoted as $PR(p_i)$) is defined as $PR(p_i) = 1 - d_f + d_f \sum_{\forall p_k \in M(p_i)} \frac{PR(p_k)}{deg_o(p_k)}$, where d_f represents the damping factor (whose value usually set to 0.85 [45]). Further, $M(p_i)$ represents the set of participants that have directed links pointing from participant p_i. The term $deg_o(p_k)$ represents the out-degree of a participant (node) p_k.

The social graph-based attribute vector $A_{p_i}^{(G)}$ is represented as

$$A_{p_i}^{(G)} =< CC(p_i), B_C(p_k), C(p_i), PR(p_i) > \quad (6)$$

For a given (unknown) participant $p_i \in P$, the social attribute vector is represented as

$$A_{p_i} =<< A_{p_i}^{(T)} >, < A_{p_i}^{(U)} >, < A_{p_i}^{(G)} >> \quad (7)$$

Therefore, the social attributes determines the social bots among legitimate users.

Algorithm 1 Extract_Social_Attributes.

Input:
A set of all online social network participants (users) $P = \{p_1, p_2, ..., p_n\}$
Output:
S, set of states (social attributes)
Procedure:
Initialize S={}
for each participant $p_i \in P$ **do**
  $< A^{(T)} >=$ Extract_Tweet_Based_Features(p_i)
  $< A^{(U)} >=$ Extract_User_Profile_Based_Features(p_i)
  $< A^{(G)} >=$ Extract_Graph_Based_Features(p_i)
  $A_{p_i}=<< A_{p_i}^{(T)} >, < A^{(U)_{p_i}} >, < A^{(G)_{p_i}} >>$
  $\bar{A}_{p_i}$=Normalization(A)
  pv=PCA($\bar{A}_{p_i}$)
  for each attribute $a_i \in \bar{A}_{p_i}$ **do**
    Compute geometric mean $wt_{a_i} = \frac{(\prod_{j=1}^{n} a_{ij})^{1/n}}{\sum_{i=1}^{n}(\prod_{j=1}^{n} a_{ij})^{1/n}}$
    S = wt_{a_i}
  end for
end for
S=Create a list of ranked states (social attributes) with respect to pv
return S

As shown in Algorithm 1, for each participant (user) in an online social network, we extract three different types of social attributes, such as tweet-based attributes ($< A^{(T)} >$), user-profile based attributes $< A^{(U)} >$ and social graph

based attributes $< A^{(G)} >$. The tweet-based attributes are extracted from the content of each user's tweets. The user-profile based attributes are derived from a series of each user's tweets (with weekly sampling time period) based on the tweeting behavior and the user interactions. Further, the social graph-based attributes are extracted based on the social relationships among the users (Line 3-6). The social attributes have to be normalized since the range of the values are different for different social attributes. In this paper, we adopt *z-score* normalization technique, where a value of each social attribute a_i is normalized as $a_i' = \frac{a_i - \bar{a_i}}{\sigma_{a_i}}$ (σ_{a_i} and $\bar{a_i}$ are the standard deviation and mean of social attribute a_i) (Line 7). All the social attributes are not equally important for social bot detection. Further, we adopt principal component analysis (*PCA*) method by integrating with a ranking measure in order to create a priority vector which ranks the social attributes based on their relative importance. Moreover, the extracted social attributes should be weighted before determining Q-value function, since the social attributes have more impact on bot detection (Line 8-13).

3.3 Our proposed deep Q-learning model for detecting social bots

We propose a deep Q-learning algorithm by considering the following elements:

- *State:* The state vector S_t (for each user) at time slot t is defined by a set of social attribute. The state vector S for a participant (i.e., user) is represented as

$$S_t =< \{s_t^{i1}, s_t^{i2}, ...s_t^{ij}\} > \tag{8}$$

  Here, each participant (user) $p_i \in P$ is associated with a set of social attributes A (refer Section 3.2 (7)). Moreover, each value s_t^{ij} represents the j^{th} social attribute of i^{th} participant at time slot t. Further, the goal state is defined as detecting each user as a social bot or not.
- *Action:* An action is the selection of a state among *'n'* states based on the current state. Moreover, the learning agent's movement from one state to another state is defined as an action α. Further, at each state, the server has to decide whether the corresponding user is a social bot or a normal user based on the social attributes (refer Section 3.2 (7)).
- *Reward:* The reward value is determined based on the social behavior of each participant (user).Therefore, the reward function r_t at a time *'t'* is computed as follows:

$$r_t = \beta x_t^{ij} + c \tag{9}$$

where x_t^{ij} represents the j^{th} social attribute value associated with a participant P_i at time *'t'*. Let β represents a model parameter (whose value lies between 0 and 1). Further, if the state is a goal state (i.e., detected as a social bot) then it gets rewarded and c set to 1. Otherwise, if goal state is not obtained then c will be -1.

Algorithm 2 Deep Q-Learning algorithm.

Input:
A set of all online social network participants (users) $P = \{p_1, p_2, ..., p_n\}$
Output:
Set of all social bots
Procedure:

1: Initialize replay memory m, deep Q-network with associated weights w as $Q(s_t^{ij}, \alpha_t;w)$ and the deep Q-network with associated weights $w^- = w$, episode $i = 1$
2: **while** $i < n$ **do** // n represents total number of participants
3: **for** t = 1, 2.....τ **do**
4: Initializes state vector S_t and begins with a state s_t^{ij}
5: Randomly choose a learning action α_t
6: Determine $\alpha_t = argmax_\alpha[Q(s_t^{ij}, \alpha_t; w)]$ by observing the next state $\overline{s}_t^{ij}$ and the reward r_t
7: Store $e_t =< s_t^{ij}, \alpha_t, r_t, \overline{s}_t^{ij} >$ into m
8: Compute $Q(s_t^{ij}, \alpha_t; w) = (1-\epsilon)Q(s_t^{ij}, \alpha_t) + \epsilon\{r + \gamma\, Q(\overline{s}_t^{ij}, argmax_{\alpha'}[Q(\overline{s}_t^{ij}, all\_actions; w)])\}$
9: Compute the target Q-value y by using $y = r + \gamma\, Q(s_t^{ij}, argmax_{\alpha'}[Q(\overline{s}_t^{ij}, all\_actions; w)]; w^-)$
10: Deep Q-network is updated by reducing the loss function $Loss(w)$ by using (2)
11: Deep Q-network is updated with a learning rate parameter ϵ, $w^- = \epsilon w + (1-\epsilon)w^-$
12: **end for**
13: i++
14: **end while**
15: **return** Set of all social bots

The proposed *Deep Q-Learning* algorithm (refer Algorithm 2) initializes replay memory and deep Q-network (which is denoted $Q(s_t^{ij}, \alpha_t; w)$) with associated weights w (Line 1). For each episode (i.e., user), *Deep Q-Learning* algorithm initializes state vector S_t and begins with a state s_t^{ij} (which represents the j^{th} social attribute of i^{th} participant) at time slot t. Moreover, by random selection the learning agent chooses an action α_t at time t. Later, the learning agent executes the learning action α_t by observing the next state $\overline{s}_t^{ij}$ and reward r_t at time slot t. For each action,

the learning agent (i.e., deep Q-network) stores the past interactions (as an experience tuple $e_t = < s_t^{ij}, \alpha_t, r, \overline{s}_t^{ij} >$) into replay memory. The *Deep Q-Learning* algorithm is executed as follows: (i) Update the Q-values (by using (1)), (ii) Compute the target Q-values (by using (3)), (iii) Deep Q-network is updated by reducing the loss function (by using (2)) and (iv) Deep Q-network parameter w (where w represents the weights associated with deep Q-network) is updated at time slot t. Therefore, this process will be repeated for a finite number of episodes (Line 2-14).

3.4 Influence bots in Twitter

In order to influence the user in the *Twitter* network, the social bots may post malicious information in a tweet. Moreover, social bots may influence a few legitimate users by posting attractive and fake information in the tweet. Even though the social bots are isolated from the *Twitter* network, few users may be influenced by the tweets posted by the social bot. In this paper, we have identified the most influential users, which are influenced by social bots (termed as influence bots) in *Twitter* network. The user influence value is defined as a measure of influencing more number of users by rapidly sharing a tweet among users and influencing based on their social interactions (such as retweets, replies, comments and mentions) in the *Twitter* network. Moreover, after reading a tweet, a reader may post comment about a tweet, retweeting a tweet or posting a tweet with similar opinion. Hence, this implies that a user has influenced reader's opinion. Therefore, the user influence value is determined based on the social interaction behavior of other user's after reading a tweet. If a tweet has more number of comments, likes and retweets, then the user influence value is high. As mentioned in Algorithm 3, a user influence score (UI) is based on two parameters, such as the influence of user's tweets and influence of user's interactions in the *Twitter* network and it is defined as

$$UI_{p_i} = \begin{cases} I^T(p_i) + I^I(p_i) & \text{if participant (user) } p_i \text{ follows participant } p_j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $I^T(p_i)$ is the influence of user's tweets and $I^I(p_i)$ is the influence of user's interactions.

3.4.1 Influence of user's tweets

The influence of each user's tweet is based on the number of comments, retweets and replies. Commenting on a tweet represents that a user wants to express his/her views and willing to share the opinion of tweet with his/her friends. Retweeting a tweet represents that a user is supporting about the opinion of tweet. Moreover, if a tweet is commented, retweeted, liked and replied more number of times, then it indicates that the probability of user reading a tweet is high. The influence of user's series of tweets I^T is defined as the probability of sharing a tweet from participant (user) p_i to its neighbors is defined as

$$I^T(p_i) = Co(tw) + Li(tw) + RT(tw) + RE(tw) \tag{11}$$

where Li(tw) and Co(tw) represent the number of likes and comments posted for tweet tw, respectively. Further, RT(tw) and RE(tw) represent the number of retweets and replies posted for tweet tw, respectively.

Algorithm 3 User_Influence_score.

Input:
A set of all online social network participants (users) $P = \{p_1, p_2, ..., p_n\}$

Output:
User Influence score

Procedure:

1: **for** each participant $p_i \in P$ **do**
2: **for** each tweet $tw \in$tweets **do**
3: Compute influence value of user's tweet $I^T_{p_i}$ by using (11)
4: **end for**
5: Compute influence of user's interactions $I^I_{p_i}$ by using (12)
6: Compute user influence score UI_{p_i} by using (10)
7: **end for**

3.4.2 Influence of user's interactions

The influence of user's interactions is based on the clustering coefficient, betweenness centrality and closeness centrality measures (refer Section 3.2.3). If the user's degree centrality is high, then it implies that the probability of reading a tweet will be high. If the user's clustering coefficient is high, then it implies that all its neighbors are strongly connected. If the user's betweenness centrality is high, then the user can quickly share tweet to the entire *Twitter* community through a few users. If the user's closeness centrality is high, then the user has more ability in order to control the information from spreading. The influence of user's interactions I^I is defined as

$$I^I(p_i) = D_c(p_i) + CC(p) + B_C(p_i) + C(p_i) + PR(p_i) \tag{12}$$

where $D_c(p_i)$ and $CC(p_i)$ represent the degree centrality and clustering coefficient of participant (user) p_i, respectively. $B_C(p_i)$ and $C(p_i)$ represent the betweenness cen-

trality and closeness centrality of p_i, respectively. Further $PR(p_i)$ is denoted as the pagerank centrality of p_i.

Algorithm 4 top-k influential users.

Input:
A set of all online social network legitimate participants (users) $P = \{p_1, p_2, ..., p_n\}$, maximum number of iterations max, threshold T

Output:
K top-k influential users

Procedure:

```
UI={}, K={}
for (i = 1; i ≤ max; i++) do
    for each participant p_i ∈ P do
        s= User_Influence_score(p_i)
        UI = UI ∪ s
    end for
    Rank the users R based on their influential value UI
    Obtain the top-k influential users R_i at i^th iteration
    Compute the rank distance d_r(k) between R_i and R_i−1 by using (13)
    if d_r(k) ≤ T then
        break
    else
        K=Update(K,R)
    end if
end for
return K
```

3.4.3 Proposed top-k influential users algorithm

The proposed top-k influential algorithm (refer Algorithm 4) is used to identify the most influential users (which are influenced by the social bots) based on tweets and the user's interactions in *Twitter* network. For each user, we need to compute the user influence value (refer Algorithm 3). Moreover, we need to rank the users based on their influence value. Hence, we monitor the ranking value of each user between two consecutive iterations. The rank distance $d_r(k)$ is measured between the ranking of k^{th}-influential user in two consecutive (i.e., at i^{th} and $i^{th} - 1$) iterations and it is defined as

$$d_r(k) = \sum_{i=1}^{K} |R_i(p_i) - R_{i-1}(p_i)| \tag{13}$$

where $R_i(p_i)$ and $R_{i-1}(p_i)$ represent the ranking of influential user p_i at i^{th} and $i^{th} - 1$ iterations, respectively. Let K represents the total number of influential users (which are influenced by social bots). If the difference between the ranking value of user in two consecutive iterations is less than threshold T_f, then the algorithm is terminated and returns the top-k influential users. Moreover, larger T_f lead to high accuracy of identifying the most influential users.

4 Experimental setup and performance evaluation

In this section, the experimental results are presented to evaluate the performance of our proposed *Deep Q-Learning* algorithm by considering three real-world datasets collected from the *Twitter* network, such as *The Fake Project* dataset, [13], *Social Honeypot* dataset [26] and *User Popularity Band* dataset (the dataset is partitioned into four groups based on number of followers) [19]. The details of three different *Twitter* datasets are presented in Table 3. We compare the proposed *Deep Q-Learning* algorithm (with three hidden layers) with the other existing algorithms, such as feed-forward neural network (*FFNN*) [38], deterministic *Q*-Learning (QL) [42] and regularized deep neural network (*RDNN*) [6]. We have evaluated our proposed DQL algorithm for social bot detection and identified top-k influential users. Further, we compare our proposed top k-influential users algorithm with other existing algorithms, such as degree centrality based radius-neighborhood (*DERND*) [3], suspected infected recovered (*SIR*) diffusion model [34] and true-top [46]. Hence, we evaluate the performance of our proposed *Deep Q-Learning* algorithm in terms of precision, recall (true positive rate), false positive rate and f-measure (refer Section 4.1) [47]. We

Table 3 Summary of datasets collected from Twitter

	Dataset name	Human	Bots	Total accounts	Tweets
Dataset 1	The Fake Project	3474	991	4465	9,987,698
Dataset 2	Social Honeypot	19,276	22,223	41499	5,613,166
Dataset 3	User Popularity				
	Band 10M	26	24	50	150,336
	Band 1M	450	296	746	303,517
	Band 100K	740	707	1447	230,577
	Band 1K	794	499	1293	37,679

have conducted experiments using a GPU-based server with *NVIDIA Tesla*, which has 4 *NVIDIA GPUs*. The *CPU* is 2.10 GHz *Intel Xenon E5-2620 v4* processor with 32GB memory. The NVIDIA driver version is 410.79. Further, CUDA 10.0 with cuDNN 7.3.1 is used. The software environment is tensorflow 1.12.0 with Spark 2.4.0 in Ubuntu 16.04.

We consider three different types of social attributes (such as tweet-based attributes, user profile-based attributes and social graph-based attributes) from the three different *Twitter* datasets. We have summarized the outcomes of our proposed *Deep Q-Learning* algorithm by defining the following metrics:

- *True Positive (TP)*: the total number of users detected as social bots, which are actually social bots,
- *True Negative (TN)*: the total number of users detected as legitimate users, which are actually legitimate users,
- *False Positive (FP)*: the total number of users detected as legitimate users, which are actually social bots,
- *False Negative (FN)*: the total number of users detected as social bots, which are actually legitimate users.

Further, we define the following metrics to compare the proposed *Deep Q-Learning* algorithm with all other existing algorithms, namely:

- *Accuracy*: the proportion of true positives and true negatives, which is defined as $\frac{TP+TN}{TP+TN+FP+FN}$,
- *True positive rate (or Recall)*: the proportion of real positives that correspond to predicted positives, which is defined as $\frac{TP}{TP+FN}$,
- *False positive rate*: the proportion of real negatives that correspond to predicted negatives, which is defined as $\frac{FP}{FP+TN}$,
- *Precision*: the proportion of predicted positives that correspond to real positives, which is defined as $\frac{TP}{TP+FP}$.
- *F-measure*: $\frac{2.precision.recall}{precision+recall}$

4.1 Experimental results

Figure 2a, b and c show the convergence performance of our proposed *DQL* algorithm with two different learning

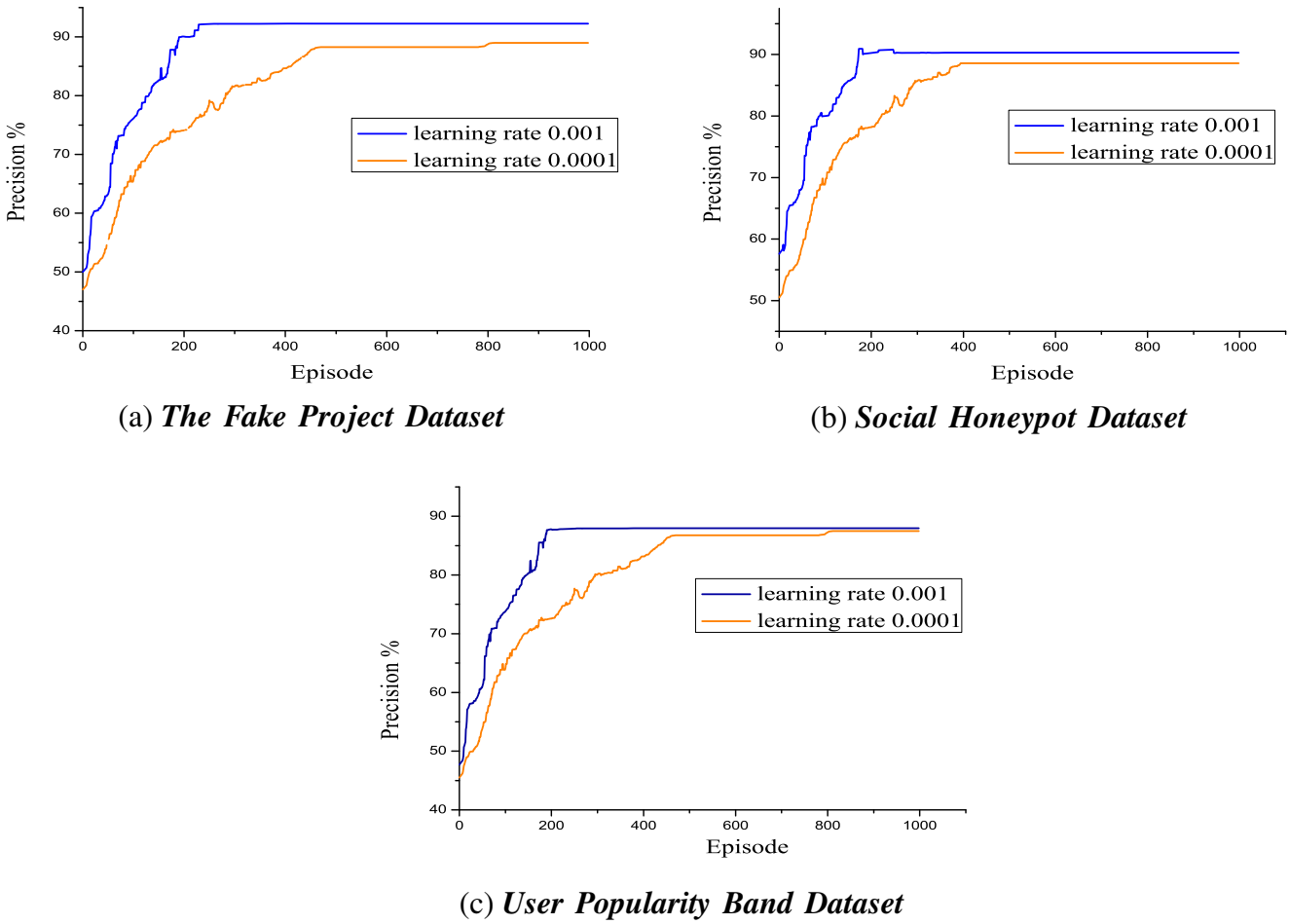
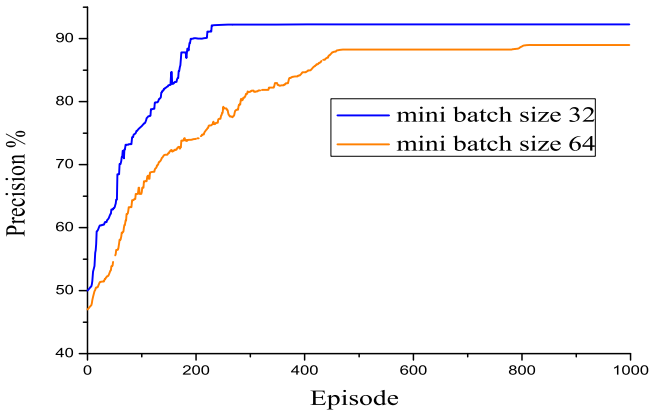


Fig. 2 Comparison of precision value with different learning rate parameter values

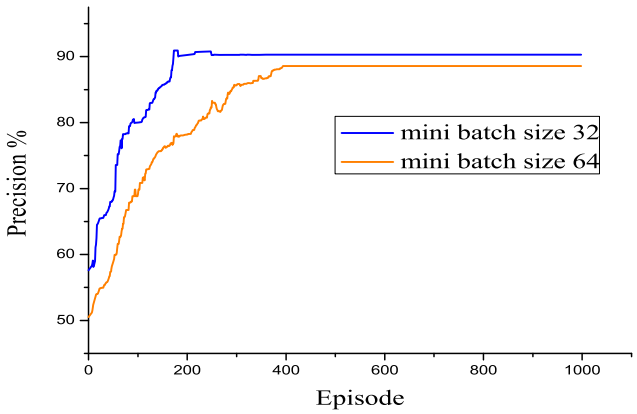
Table 4 List of parameters

Parameter	Value
Learning rate	0.001
Discount factor	0.99
Mini batch size	32
Replay memory size	50,000

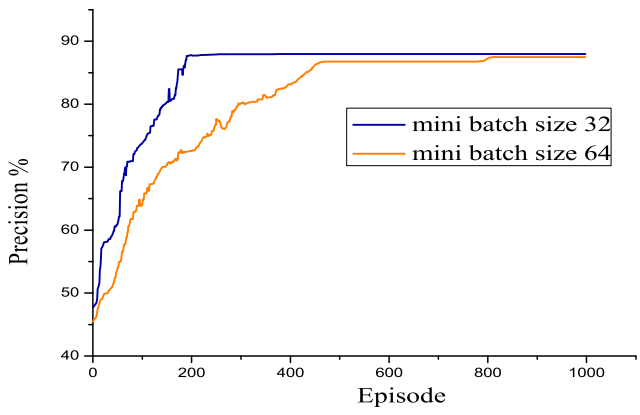
rate parameter values i.e., $\epsilon = 0.001$ and $\epsilon = 0.0001$. We can observe that our proposed algorithm quickly converges with a learning rate of $\epsilon = 0.001$ when compared to $\epsilon = 0.0001$. Moreover, a higher learning rate leads to a local optimum in order to obtain higher precision value. From Fig. 2, it can be observed that for learning rate $\epsilon = 0.001$, the precision value is more than 90% (on an average) for social bot detection. Further, lowering learning rate value below 0.0001 will give lower precision. The convergence of target Q-function is also affected by other parameters, such as discount factor and mini batch size. The parameter values that are used for computing the target Q-values are listed in Table 4. The discount factor γ determines how much weight it provides for future reward (γ value usually lies in [0, 1]). If discount factor $\gamma = 0$, implies that the state-action values represent the current reward. If discount factor γ is approaches to 1, then the state-action values represent a (constant) high reward. Moreover, if discount factor γ is 1 (or exceeds 1), then the state-action values may diverge [16]. Therefore, we have chosen discount factor $\gamma = 0.99$. Figure 2a, b and c show the convergence performance of mini-batch size in the proposed deep Q-learning algorithm. The mini-batch size determines number of experience tuples in each training step. The mini-batch size is usually based on computational system on which the experimentation is being performed [21]. From Fig. 3, we can observe that our proposed algorithm can converge quickly with smaller mini batch size 32 as compared to larger mini batch size 64. It can be observed that for mini-batch size 32, a high precision is achieved (i.e., more than 90% precision, on an average)



(a) ***The Fake Project Dataset***



(b) ***Social Honeypot Dataset***

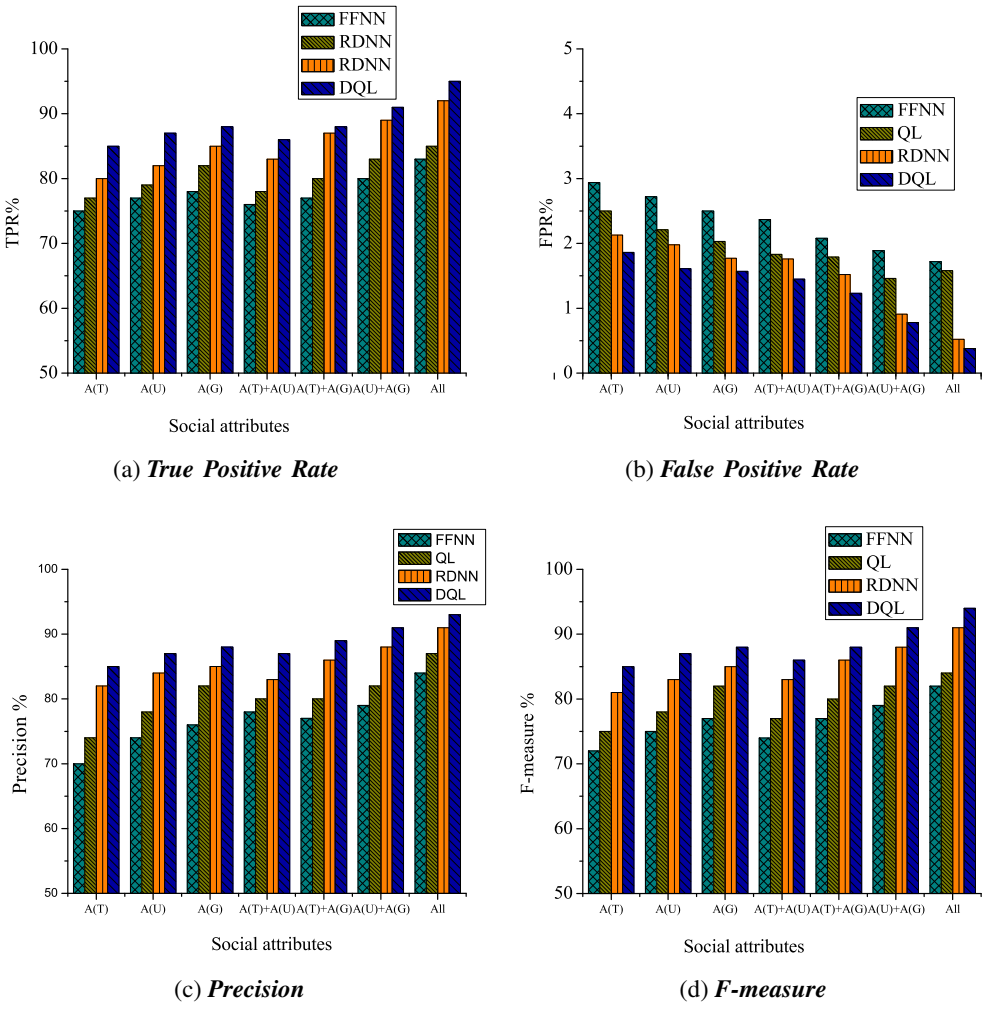


(c) ***User Popularity Band Dataset***

Fig. 3 Comparison of precision value with different mini batch sizes

for social bot detection. Further, increasing mini-batch size (i.e., greater than 64) will give lower precision.

We consider a set of social attributes, such as tweet-based attributes (i.e., from the content of each user tweet), user profile-based attributes (from a series of each user's tweets) and social graph-based attributes (i.e., the user establishes the social relationship with their friends and followers), which are denoted as $A(T)$, $A(U)$ and $A(G)$ respectively (discussed in Section 3.2). We compare the social bot detection performance of the proposed *Deep Q-Learning (DQL)* algorithm with other existing algorithms (such as feed-forward neural network (*FFNN*), deterministic Q-Learning (*QL*) and regularized deep neural network (*RDNN*) [6, 38, 42] in terms of precision, recall and f-measure on three different *Twitter* datasets (such as *The Fake Project* dataset, *Social Honeypot* dataset and *User Popularity Band* dataset). From Fig. 4, we can observe that all the algorithms can obtain the best social bot detection performance by considering all the three different types of social attributes. When we consider only user profile-based attributes, the social bot detection performance of the proposed *DQL* algorithm and the *Regularized Deep Neural Network (RDNN)* [6] has been fallen down from 87% to 84% respectively on precision value. From Fig. 4, we can also observe that by considering only tweet-based attributes, the social bot detection performance of all algorithms is drastically reduced when compared to user profile-based attributes. However, by combining the tweet-based attributes with the user profile-based attributes, the social bot detection performance has been improved approximately 5-9% (i.e., from 5% to 9%) on precision value. Therefore, by combining all the social attributes, the social bot detection performance has been improved approximately 4-10% on precision value. From Fig. 5, we can observe that by combining the tweet-based attributes with the user profile-based attributes, the social bot detection performance has been improved approximately 4-8% on precision value. Therefore, by combining all the social attributes, the social bot detection performance has been improved approximately 3-8% on precision value. From Fig. 6, we can observe that by combining the tweet-based attributes with the user profile-based attributes, the social bot detection performance has been improved approximately 4-9% on precision value. Therefore, by



(a) ***True Positive Rate*** (b) ***False Positive Rate***

(c) ***Precision*** (d) ***F-measure***

Fig. 4 Experimental results by considering all possible combinations of social attributes on *The Fake Project* dataset

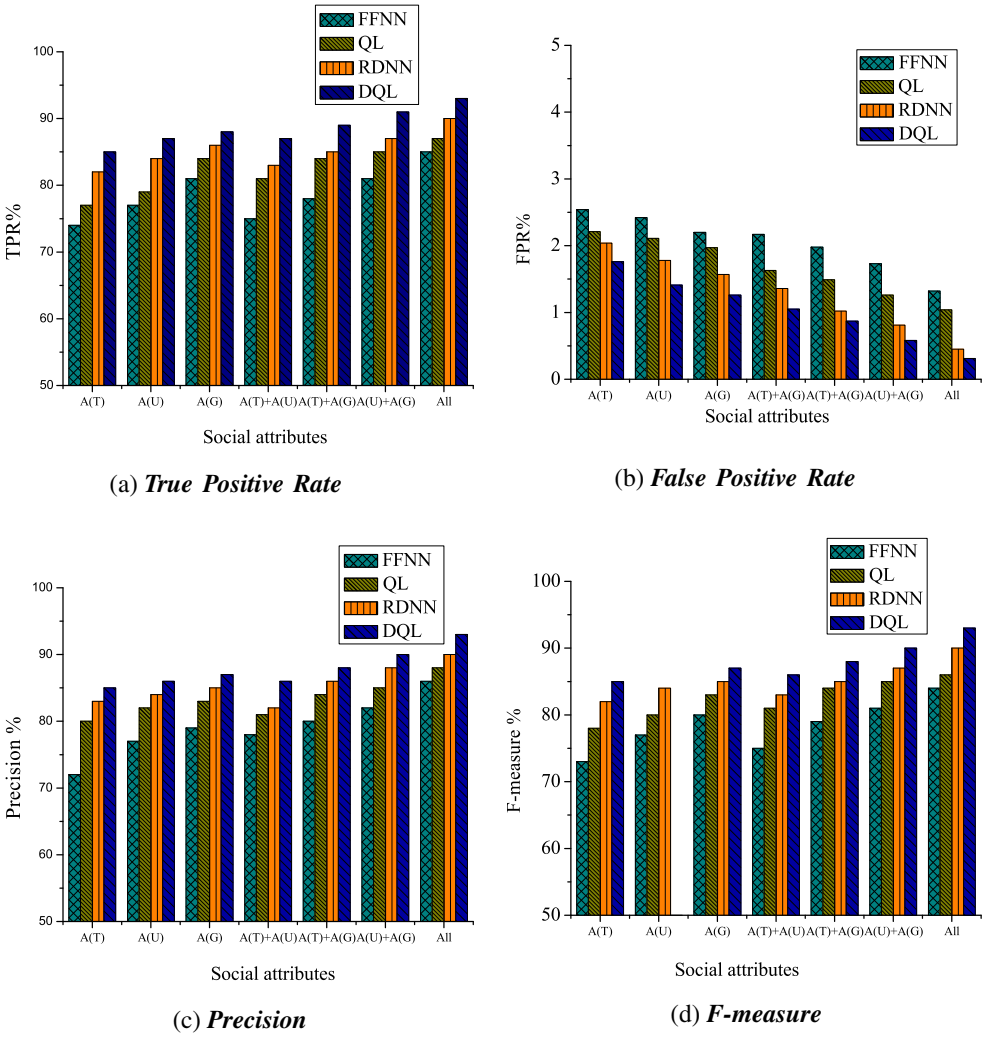


Fig. 5 Experimental results by considering all possible combinations of social attributes on *Social Honeypot* dataset

(a) ***True Positive Rate*** (b) ***False Positive Rate***

(c) ***Precision*** (d) ***F-measure***

combining all of the social attributes, the social bot detection performance has been improved approximately 5-10% on precision value. Table 5 shows performance of proposed *DQL* algorithm for 5-fold cross-validation.

Table 6 shows the average execution time for the proposed *Deep Q-Learning* (*DQL*) algorithm. The average execution time for the *DQL* algorithm is computed with one, two and three hidden layers, which are denoted as *DQL-1, DQL-2* and *DQL-3*, respectively. As the number of hidden layers increase, the average execution time also increases. This is due to fact that the *DQL* consumes more execution time (as number of hidden layers increases) for training target Q-function parameters, such as learning rate, mini-batch size and discount factor.

We evaluate the performance of our proposed top-k influential users algorithm by considering the following metrics.

– *Precision:* The precision value is defined as $\frac{|LU_1(k) \cap LU_2(k)|}{|LU_1(k)|}$, where LU_1 represents the list of legitimate users ranked by the user influence metric (refer Section 3.4, (10)) and LU_2 represents the list of legitimate users ranked based on the user interactions (such as retweets, replies, comments and likes). Further, $LU_1(k)$ and $LU_2(k)$ represents the top-k influential users in LU_1 and LU_2, respectively.
– *Recall:* The recall value is defined as $\frac{|LU_1(k) \cap LU_2(k)|}{|LU_2(k)|}$.

Figures 7, 8 and 9 show that the proposed influential users algorithm has the better recall and precision than other existing algorithms, such as degree centrality based radius-neighborhood (*DERND*) [3], suspected infected recovered (*SIR*) diffusion model [34] and true-top [46] (on all three different *Twitter* datasets). We can observe that as k-value increases, the recall values of all algorithms increase. The experiment results of the proposed algorithm shows that the tweet-based attributes and the user interactions are two important factors in order to influence the user. The precision of our proposed algorithm is approximately 80% as shown in Figs. 7b, 8b and 9b. This implies that the proposed algorithm can identify 80% of top-10% influential

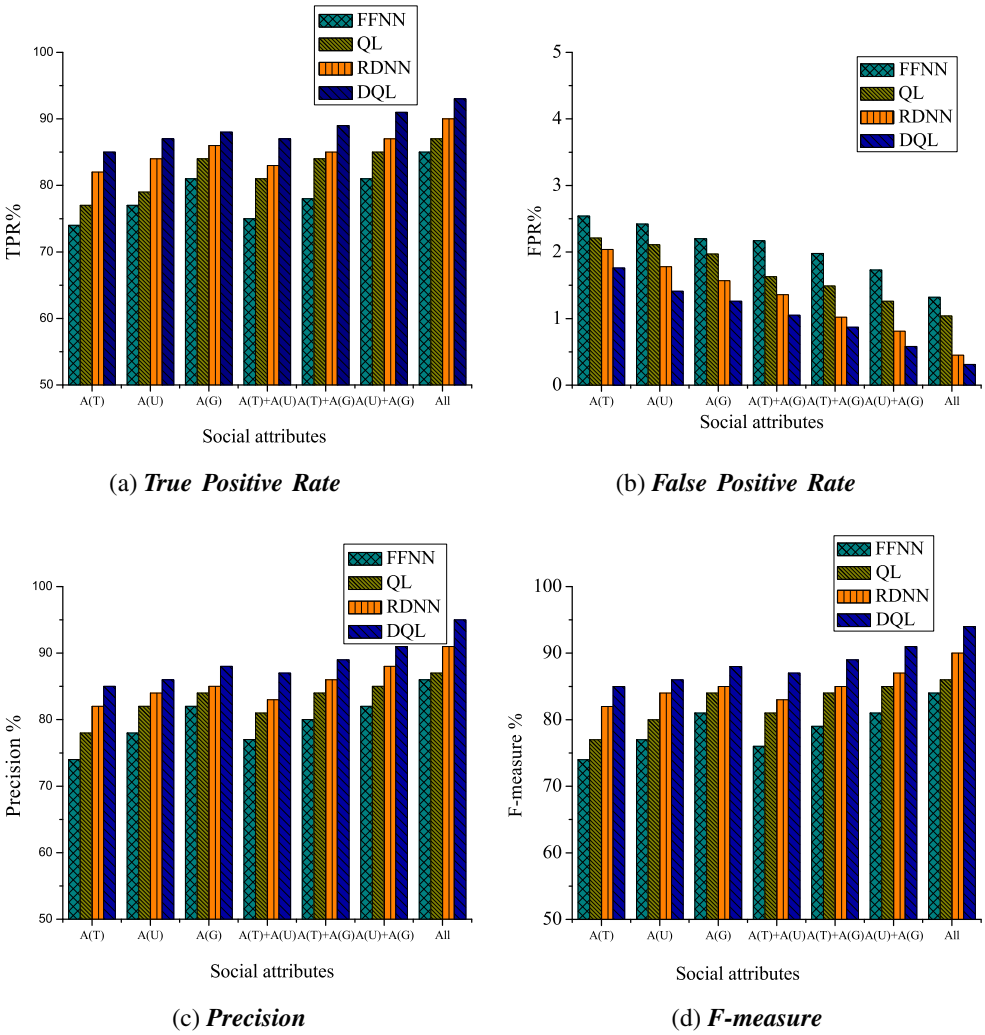


Fig. 6 Experimental results by considering all possible combinations of social attributes on *User Popularity Band* dataset

(a) ***True Positive Rate*** (b) ***False Positive Rate***

(c) ***Precision*** (d) ***F-measure***

users, which were influenced by the social bots. Moreover, the influential users may attract other legitimate users and become trustworthy users, which affects the entire *Twitter* community. The computation of influence score for each user makes the proposed method consume more time. However, the proposed method identifies the most influential users (which are influenced by social bots) in online social networks more effectively. The proposed method is more efficient than the existing *True top* algorithm because the proposed method is based on various centrality measures that determines the spreading probability of information in *Twitter* network. From Figs. 7b, 8b and 9b, we can observe that *DERND* [3] algorithm cannot effectively identify the influential users because this method gives same precision value as number of the influential users increases. The existing *SIR* diffusion model and *DERND* algorithm identify the influential users based on only semi-local degree centrality and radius-neighboring degree centrality measures, respectively. Moreover, the users with high degree centrality measure may not necessarily have more number of retweets or comments. We observe that the proposed algorithm performs better than the other existing algorithms in terms of tweet propagation under the influence value of each user tweet I^T. Further, the proposed method has a high influence spreading probability based on the influence of user interaction. This means that the proposed

Table 5 Performance of the proposed Deep Q-learning algorithm for 5-fold cross-validation

Dataset	1-fold	2-fold	3-fold	4-fold	5-fold	Average
The fake project	93.24	93.13	94.11	93.18	93.53	93.43
Social honeypot	93.36	93.28	93.51	93.62	94.15	93.65
User popularity	94.09	93.54	94.37	94.62	93.75	94.07

Table 6 Average execution time for the proposed DQL algorithm with different number of hidden layers

Dataset	Execution time in seconds		
	DQL-1	DQL-2	DQL-3
The fake project	2521	2754	2846
Social honeypot	1521	1676	1707
User popularity	904	972	1012

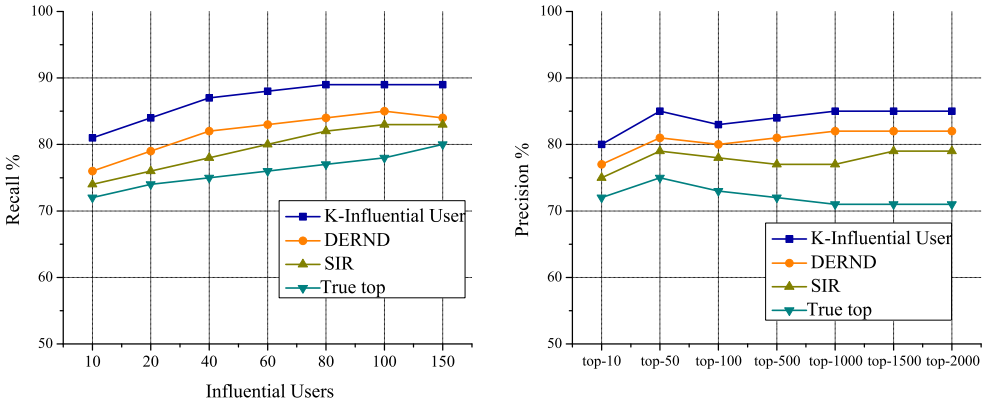


Fig. 7 Top-k Influential Users on *The Fake Project* Dataset

(a) ***Recall*** (b) ***Precision***

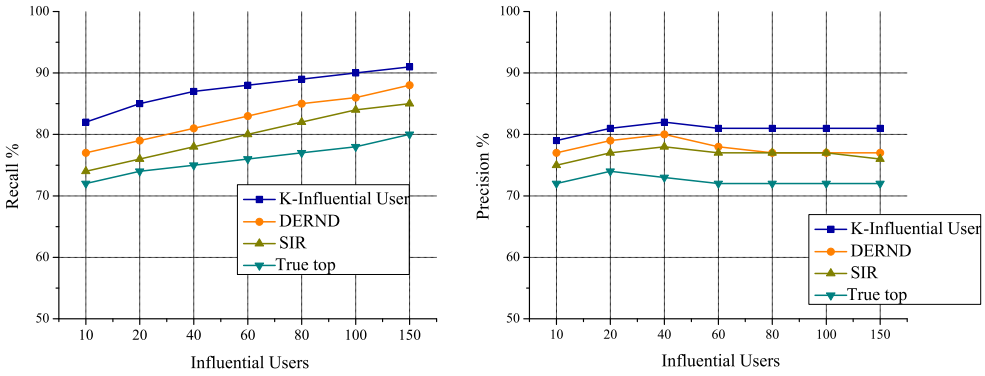


Fig. 8 Top-k Influential Users on *Social Honeypot* Dataset

(a) ***Recall*** (b) ***Precision***

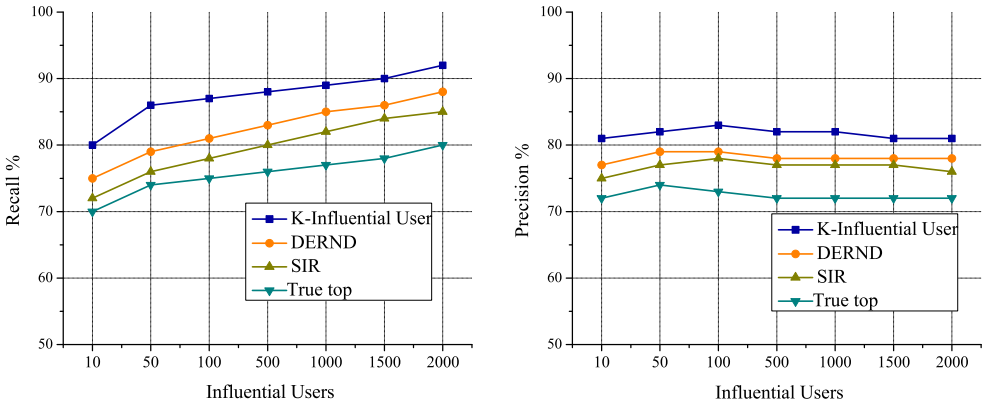


Fig. 9 Top-k Influential Users on *User Popularity Band* Dataset

(a) ***Recall*** (b) ***Precision***

algorithm selects the users which are influenced by social bots so that these users cannot further influence the current users. Moreover, these users are also detected as influential bots. Therefore, we present the top-k influential users by ranking the users based on the user influence score which is measured on user interactions and tweet propagation.

5 Conclusion

In this paper, we have considered a set of social attributes, such as tweet-based attributes, user profile-based attributes and social graph-based attributes, which are used for detecting the social bots. We propose a deep Q-network based architecture by integrating deep Q-learning model with social attributes for social bot detection based on the Q-value function (i.e., state-action value function). Further, a top-k influential user algorithm has been proposed to identify the most influential users based on tweets and the user's interactions. Three different datasets collected from the *Twitter* network, such as *The Fake Project* dataset, *Social Honeypot* dataset and *User Popularity Band* dataset are used to evaluate the performance of our proposed *Deep Q-Learning* algorithm. We have compared the proposed *Deep Q-Learning* algorithm with the other existing algorithms, such as *Feed-Forward Neural Network*, *Deterministic Q-Learning* and *Regularized Deep Neural Network*. For social bot detection, the proposed algorithm with the tweet-based attributes achieves average accuracy of 85% on the precision value, the proposed algorithm with the user profile-based attributes achieves average accuracy of 87% on the precision value and the proposed algorithm with the social graph-based attributes achieves average accuracy of 88% on the precision value. Therefore, by integrating all social attributes we have achieved average accuracy of 93% on the precision value. The experiment results show that the *Deep Q-Learning* algorithm provides 5-9% improvement of precision over other existing algorithms. Further, the experiment results show that the proposed algorithm can identify top-10% influential users with a better precision value (i.e., 80%) when compared with other existing algorithms.

References

1. Ala'M AZ, Faris H, Hassonah MA et al (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104
2. Albladi SM, Weir GR (2018) User characteristics that influence judgment of social engineering attacks in social networks. Human-centric Comput Inf Sci 8(1):5
3. Alshahrani M, Zhu F, Zheng L, Mekouar S, Huang S (2018) Selection of top-k influential users based on radius-neighborhood degree, multi-hops distance and selection threshold. J Big Data 5(1):28
4. Ashfaq AB, Abaid Z, Ismail M, Aslam MU, Syed AA, Khayam SA (2016) Diagnosing bot infections using Bayesian inference. J Comput Virol Hacking Techniq, 1–18
5. Baltazar J, Costoya J, Flores R (2009) The real face of koobface: the largest web 2.0 botnet explained. Trend Micro Res 5(9):10
6. Barushka A, Hajek P (2018) Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. Appl Intell, 1–19
7. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022
8. Boshmaf Y, Muslukhov I, Beznosov K, Ripeanu M (2011) The socialbot network: when bots socialize for fame and money. In: Proceedings of the 27th annual computer security applications conference. ACM, pp 93–102
9. Cesarano C, Dorr B, Picariello A, Reforgiato D, Sagoff A, Subrahmanian V (2004) Oasys: an opinion analysis system. In: AAAI Spring symposium on computational approaches to analyzing weblogs (CAAW 2006), pp 21–26
10. Cheng Z, Caverlee J, Lee K (2013) A content-driven framework for geolocating microblog users. ACM Trans Intell Syst Technol (TIST) 4(1):2
11. Chowdhury S, Khanzadeh M, Akula R, Zhang F, Zhang S, Medal H, Marufuzzaman M, Bian L (2017) Botnet detection using graph-based feature clustering. J Big Data 4(1):14
12. Chu Z, Gianvecchio S, Wang H, Jajodia S (2012) Detecting automation of twitter accounts: are you a human, bot, or cyborg? IEEE Trans Depend Secur Comput 9(6):811–824
13. 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. International World Wide Web Conferences Steering Committee, pp 963–972
14. Dickerson JP, 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 (ASONAM), pp 620–627
15. Ferrara E, Varol O, Davis C, Menczer F, Flammini A (2016) The rise of social bots. Commun ACM 59(7):96–104
16. François-Lavet V, Fonteneau R, Ernst D (2015) How to discount deep reinforcement learning: towards new dynamic strategies. arXiv:1512.02011
17. Freitas C, Benevenuto F, Veloso A, Ghosh S (2016) An empirical study of socialbot infiltration strategies in the twitter social network. Soc Netw Anal Min 6(1):23
18. Gadaleta M, Chiariotti F, Rossi M, Zanella A (2017) D-dash: a deep q-learning framework for dash video streaming. IEEE Trans Cogn Commun Network 3(4):703–718
19. Gilani Z, Wang L, Crowcroft J, Almeida M, Farahbakhsh R (2016) Stweeler: a framework for twitter bot analysis. In: Proceedings of the 25th international conference companion on world wide web. International World Wide Web Conferences Steering Committee, pp 37–38
20. Halfaker A, Riedl J (2012) Bots and cyborgs: Wikipedia's immune system. Computer 45(3):79–82
21. Islam R, Henderson P, Gomrokchi M, Precup D (2017) Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. arXiv:1708.04133
22. Ji Y, He Y, Jiang X, Cao J, Li Q (2016) Combating the evasion mechanisms of social bots. Comput Secur 58:230–249
23. Kaufmann M, Kalita J (2010) Syntactic normalization of twitter messages. In: International conference on natural language processing. Kharagpur
24. Kudugunta S, Ferrara E (2018) Deep neural networks for bot detection. Inform Sci 467:312–322

25. Kusy M, Zajdel R (2015) Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network. IEEE Trans Neural Netw Learn Syst 26(9):2163–2175
26. Lee K, Eoff BD, Caverlee J (2011) Seven months with the devils: a long-term study of content polluters on twitter. In: ICWSM, pp 185–192
27. Lee S, Kim J (2013) Fluxing Botnet command and control channels with url shortening services. Comput Commun 36(3):320–332
28. Lee S, Kim J (2013) Warningbird: a near real-time detection system for suspicious urls in twitter stream. IEEE Trans Depend Sec Comput 10(3):183–195
29. Leibo JZ, Zambaldi V, Lanctot M, Marecki J, Graepel T (2017) Multi-agent reinforcement learning in sequential social dilemmas. In: Proceedings of the 16th conference on autonomous agents and multiagent systems, pp 464–473
30. Lingam G, Rout RR, Somayajulu DV (2018) Learning automata-based trust model for user recommendations in online social networks. Comput Electric Eng 66:174–188
31. Liu W, Zhang L, Tao D, Cheng J (2017) Reinforcement online learning for emotion prediction by using physiological signals. Pattern Recognition Letters
32. Ma Q, Ma J (2017) A robust method to discover influential users in social networks. Soft Comput, 1–13
33. Perera RD, Anand S, Subbalakshmi K, Chandramouli R (2010) Twitter analytics: architecture, tools and analysis. In: Military communications conference, 2010-MILCOM 2010. IEEE, pp 2186–2191
34. Sheikhahmadi A, Nematbakhsh MA, Zareie A (2017) Identification of influential users by neighbors in online social networks. Physica A: Stat Mech Appl 486:517–534
35. Sirivianos M, Kim K, Yang X (2011) Socialfilter: introducing social trust to collaborative spam mitigation. In: INFOCOM, 2011 proceedings. IEEE, pp 2300–2308
36. Subrahmanian V, Azaria A, Durst S, Kagan V, Galstyan A, Lerman K, Zhu L, Ferrara E, Flammini A, Menczer F (2016) The darpa twitter bot challenge. Computer 49(6):38–46
37. Subrahmanian VS, Reforgiato D (2008) Ava: adjective-verb-adverb combinations for sentiment analysis. IEEE Intell Syst 23(4):43–50
38. Tang R, Fong S, Deb S, Vasilakos AV, Millham RC (2018) Dynamic group optimisation algorithm for training feed-forward neural networks. Neurocomputing 314:1–19
39. Teng TH, Tan AH, Zurada JM (2015) Self-organizing neural networks integrating domain knowledge and reinforcement learning. IEEE Trans Neural Netw Learn Syst 26(5):889–902
40. Venkatachalam N, Anitha R (2017) A multi-feature approach to detect stegobot: a covert multimedia social network botnet. Multimed Tools Appl 76(4):6079–6096
41. Wagner C, Mitter S, Körner C, Strohmaier M (2012) When social bots attack: modeling susceptibility of users in online social networks. Making Sense Microposts (# MSM2012) 2(4):1951–1959
42. Wei Q, Lewis FL, Sun Q, Yan P, Song R (2017) Discrete-time deterministic q-learning: a novel convergence analysis. IEEE Trans Cybern 47(5):1224–1237
43. Wei W, Xu F, Tan CC, Li Q (2012) Sybildefender: defend against sybil attacks in large social networks. In: INFOCOM, 2012 proceedings. IEEE, pp 1951–1959
44. Wu J, Sha Y, Li R, Liang Q, Jiang B, Tan J, Wang B (2017) Identification of influential users based on topic-behavior influence tree in social networks. In: National CCF conference on natural language processing and Chinese computing. Springer, pp 477–489
45. Yan G (2013) Peri-watchdog: hunting for hidden Botnets in the periphery of online social networks. Comput Netw 57(2):540–555
46. Zhang J, Zhang R, Sun J, Zhang Y, Zhang C (2016) Truetop: a sybil-resilient system for user influence measurement on twitter. IEEE/ACM Trans Network 24(5):2834–2846
47. Zhang J, Zhang R, Zhang Y, Yan G (2016) The rise of social botnets: attacks and countermeasures. IEEE Transactions on Dependable and Secure Computing

Greeshma Lingam received B.Tech. degree in computer science and engineering from Maheshwara Engineering College, JNTU, Hyderabad, India and M.Tech. degree in computer science and engineering from DRK College of Engineering and Technology, JNTU, Hyderabad, India. She is currently pursuing Ph.D. at the National Institute of Technology Warangal, India. Her primary research area includes social networks and Big Data.

Rashmi Ranjan Rout received Ph.D. from Indian Institute of Technology (IIT) Kharagpur, WB, India. He is currently working as an Associate Professor in the Department of Computer Science and Engineering at National Institute of Technology (NIT), Warangal. His research interest includes online social networks, P2P networks, delay tolerant networks, wireless sensor networks, and Internet of Things.

D. V. L. N. Somayajulu is currently on deputation as Director for Indian Institute of Information Technology Design and Manufacturing (IIITDM), Kurnool, Andhra Pradesh. Prior to joining this post, he served as professor of Computer Science and Engineering, National Institute of Technology (NIT), Warangal and Chair, Electronics & ICT Academy, NIT Warangal. He obtained his Ph.D. from Indian Institute of Technology (IIT) Delhi, India. His research interest includes Databases, Data Analytics, Information Extraction, Query Processing, Big Data and Privacy.