Design of Social Bot Detection and Trust Models for Online Social Networks

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

Greeshma L

(Roll No. 716174)

Under the guidance of

Dr. Rashmi Ranjan Rout

Prof. DVLN Somayajulu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

TELANGANA - 506004, INDIA

OCTOBER 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

THESIS APPROVAL FOR Ph.D.

This is to certify that the thesis entitled, Design of Social Bot Detection and Trust Models for Online Social Networks, submitted by Ms. Greeshma.L [Roll No. 716174] is approved for the degree of DOCTOR OF PHILOSOPHY at National Institute of Technology Warangal.

Examiner

Research Supervisor	**Research Supervisor**
Dr. Rashmi Ranjan Rout	**Prof. DVLN Somayajulu**
Dept. of Computer Science and Engg.	**Dept. of Computer Science and Engg.**
NIT Warangal, India	**NIT Warangal, India**

Chairman

Prof. P RadhaKrishna
Head, Dept. of Computer Science and Engg.
NIT Warangal, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

CERTIFICATE

This is to certify that the thesis entitled, **Design of Social Bot Detection and Trust Models for Online Social Networks**, submitted in partial fulfillment of requirement for the award of degree of **DOCTOR OF PHILOSOPHY** to National Institute of Technology Warangal, is a bonafide research work done by **Ms. Greeshma L [Roll No. 716174]** under our supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Research Supervisor(s)

Dr. Rashmi Ranjan Rout
Associate Professor

Place: NIT Warangal
Date: October, 2020

Prof. DVLN Somayajulu
Professor

DECLARATION

This is to certify that the work presented in the thesis entitled "*Design of Social Bot Detection and Trust Models for Online Social Networks*" is a bonafide work done by me under the supervision of Dr. Rashmi Ranjan Rout and Prof. DVLN Somayajulu, and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/date/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Greeshma L

(Roll No. 716174)

Date: 17-10-2020

ACKNOWLEDGMENTS

It is with great pleasure that I acknowledge my sincere thanks and deep sense of gratitude to my supervisor(s) Dr. Rashmi Ranjan Rout and Prof. DVLN Somayajulu, Department of Computer Science and Engineering for their valuable guidance throughout the course. Their technical perception, profound knowledge, sincere effort in guiding a student and devotion to work have greatly charmed and constantly motivated me to work towards the goal. They always gave me ample time for discussions, reviewing my work and suggesting requisite corrections.

I extend my gratitude to all my Doctoral Scrutiny Committee members for their valuable suggestions and faculty members of Computer Science and Engineering Department. I wish to express my sincere thanks to Prof. R B V Subramanyam, Dr. S Ravi Chandra and Prof. A Venu Gopal for their insightful comments and suggestions during oral presentations. I am immensely thankful to Prof. P RadhaKrishna, Head of Dept. of CSE, for providing adequate facilities in the department to carry out the research work. I wish to express my sincere thanks to Prof. N.V. Ramana Rao, Director, NIT Warangal for his official support and encouragement.

My gratitude to my family for their unconditional love, support and prayers for my success in achieving the goal. Lastly I am deeply indebted to the almighty for giving the opportunity to pursue this line.

Greeshma L

Dedicated to

My Family & Teachers

ABSTRACT

Online Social Networks (OSNs) represent a platform where people (users) share links, news, opinions, promote products and services. Moreover, the users trust OSNs which leads to establish new social relationships and share the information among other OSN users. The information sharing feature is also used by social bots (or spammers) in order to spread fake information. Social bots are automated software programs that control social network user accounts with malicious activities, such as creating multiple fake accounts, spreading spam, and manipulating online ratings. Moreover, social bots also post shortened malicious URLs in the tweet in order to redirect the requests of OSN participants to some malicious web page. Therefore, the detection of social bots in an online social network is an important task.

The thesis focuses on social bot detection and trust models for distinguishing legitimate participants among social bots in OSNs. The challenging issues of social bot detection have been addressed by considering trust model with features, such as tweet-content, user profile, URL, graph and behavioral similarity based features. In this thesis, the proposed methods have detected malicious social bots in order to provide trustworthy information in online social networks. Firstly, a Learning Automata-based Malicious Social Bot Detection (LA-MSBD) algorithm has been presented by integrating a trust model with URL-based features for identifying trustworthy participants (users) in Twitter network. The proposed method computes direct and indirect trust by considering Bayesian learning and Dempster-Shafer theory, respectively. Secondly, a deep Q-network based architecture has been designed by integrating single agent deep Q-learning model with social attributes for social bot detection. A multi-agent deep Q-learning model based on particle swarm optimization (PSO) method is also proposed for detecting social spam bots more accurately. Further, a top-k influential (user) 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. Thirdly, a Deep autoencoder-based Social Botnet Community Detection (DA-SBCD) algorithm has been proposed to detect social botnet communities of social bots with higher malicious behavioral similarity. Further, an

Influential Community Detection algorithm has been proposed and this helps in reducing the spread of spam-content through influential communities in Twitter network. Finally, a Learning Automata based Recommended Trust Path Selection (LA-RTPS) algorithm has been proposed in order to evaluate trustworthy paths in online social networks for trusted-user recommendations. The experimentation using real time datasets illustrates the efficacy of the proposed algorithms.

Keywords: Online social networks (like Twitter), social bots, deep Q-learning, Q-value, particle swarm optimization, trust, learning automata, behavioral similarity, social botnet community detection, deep autoencoder.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

OSN	Online Social Network
DRL	Deep Reinforcement Learning
DQL	Deep Q-Learning
PSO	Particle Swarm Optimization
P-DQ L	Particle swarm optimization based Deep Q-Learning
LA	Learning Automata
MSBD	Malicious Social Bot Detection
SBCD	Social Botnet Community Detection
DA	Deep Autoencoder
NMI	Normalized Mutual Information
HoS	High quality of Social trust
RTPS	Recommended Trust Path Selection

Chapter 1

Introduction

Online social networks (OSNs) attract millions of users to share variety of information (through Internet) related to social activities, e-marketing, jobs, politics, sports and other news (like natural disaster notifications) [1]. Online social networking sites contain huge amount of data (such as data from online reviews, online ratings and discussions forums) which are generated by users (from various communities). The data can be accessed seamlessly due to proliferation of online social network technologies. This in turn provides an additional space (comfort) for an attacker to steal user's personal information and to perform malicious activities (generating fake identities, manipulating online ratings, spreading social spam content and performing phishing attacks) in online social networks [1]. Moreover, such huge amount of data may also contain untrustworthy and fake information [2]. However, finding untrustworthy information manually is a difficult task. Building trust of the participants among online social network users may help to provide good recommendations, credible opinions and online ratings (or reviews) [3]. Therefore, trust should be taken into consideration to avoid untrustworthy information and malicious comments posted in OSNs.

Social bot is an automated computer program that is created to perform (either malicious or non-malicious) activities in *Twitter* network. Like traditional bots (in *Internet* chat), social bots are more common in *Twitter* [4]. Social bots are created with the support of open *APIs* (like *Twitter API*) [5]. Moreover, social bots are mostly used for posting spam tweets, retweets and sharing public opinion in *Twitter*. The recent studies have identified different types of social bots, such as legitimate bots, influence bots and malicious bots [6]. Legitimate bots are used to promote products or services, natural disaster notifications and blog updates [7]. Influence bots are involved to affect users' behavior by manipulating online reviews and ratings [2]. Malicious (or spam) bots are mostly used to spread spam content, post phishing *URLs* and generate fake accounts. However, malicious social bots can also manipulate natural disaster notifications and quality of product by posting fake information or malicious comments [8]. A social botnet is a group of bots which are controlled by a botmaster.

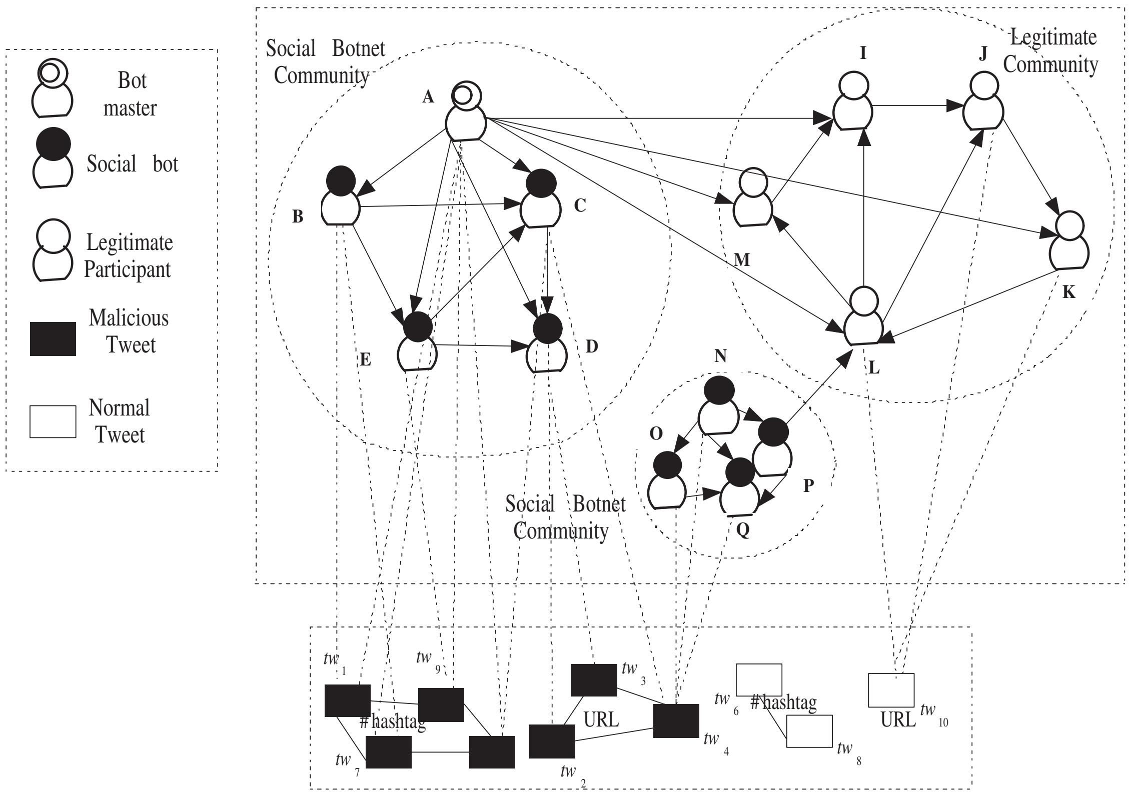


Figure 1.1: An online social network with social botnet and legitimate communities

Fig. 1.1 shows an online social network architecture with social botnet and legitimate communities, where each node represents either a social bot or legitimate user and each directed edge represents a tweeting (or re-tweeting) relationship between two users (or participants). As shown in Fig. 1.1, the botmaster *A* establishes a strong social relationship with other social bots *B, C, D* and *E*, and also with other legitimate users *I*, *K*, *M* and *L* in order to reduce the probability of identification [9]. Moreover, the botmaster creates malicious tweets (with fake information or malicious URL in the tweet) and the social bots re-tweet them. Additionally, the social bots can spread malicious tweets to other legitimate users. Further, social bots may post shortened malicious *URLs* in the tweet. When a user clicks on a shortened malicious *URL*, the user's request will be redirected to intermediate *URLs* associated with malicious servers, which in turn redirect the user to malicious web pages. Then the legitimate user is exposed to an attacker. This leads to online social networks suffering from several vulnerabilities (like phishing attack).

The traditional botnet detection approaches mainly focus on peer-to-peer networks and botnet-based command-and-control protocols [10]. Moreover, these type 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 [11]. However, bots interact among themself by considering the *private message passing features* provided by online social networks in order to establish the relationship among social bots (which are controlled by a botmaster) [12]. Most of the existing works consider only user profile-based features to detect social bots [13], [14]. Moreover, these type of approaches fail to distinguish legitimate users from the new kinds of social bots. The following are the limitations related to social bot detection in online social networks: (i) social bots can reduce trust value of their legitimate neighbors by sending fake and untrustworthy information, (ii) in an online social network, the behavior of user rapidly changes over time and thus, it is important to extract the information that is needed to evaluate a user, (iii) a few malicious users may use tools to create fake accounts and manipulate their influence value on other users 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 detect the social bots (from legitimate users) in online social networks.

The contributions in this thesis are as follows:

- **Detection of phishing bots using learning automata model:** This work presents a trust model (which consists of two components namely, direct trust and indirect trust) to evaluate trustworthiness of tweets (posted by each participant) by using *Bayesian* *learning* and *Dempster-Shafer Theory (DST)*. A *Learning Automata based Malicious Social Bot Detection* algorithm is designed by integrating a trust model with a set of *URL*-based features in order to distinguish malicious social bots from legitimate participants.

- **Deep reinforcement learning models for detecting malicious social bots and influential users:** In this work, a *deep Q-network* architecture has been designed by incorporating a single agent *Deep Q-Learning (DQL)* model using the social attributes (such as tweet-content, user profile and graph-based features) in the *Twitter* network for detection of malicious social bots. A multi-agent deep Q-Learning algorithm has been proposed by using particle swarm optimization method with users' temporal features in order to detect malicious social bots in *Twitter* network. Further, an algorithm has been proposed to identify top-k influential *Twitter* network users (which are influenced by the social bots) based on the tweets and the users' interactions.

- **Detection of Social Botnet and Spam Influential Communities:** This work analyzes the behavioral similarity of the participants by considering four different aspects, such as tweet-content similarity, shared URL similarity, interest similarity and social interaction similarity for identifying similar type of behavior (malicious or non-malicious) among participants in the *Twitter* network. This work considers the important features, such as tweet content, *URL*-based, graph-based, profile-based features and influence value of the neighboring participants in order to evaluate the trust value of each participant. Based on a deep autoencoder model, the proposed algorithm detects social botnet communities with improved precision and recall. Further, an *Influential Community Detection* algorithm has been proposed and this helps in reducing the spread of spam-content through influential communities in *Twitter* network.

- **Determining trustworthy paths using a learning automata model:** This work presents a social trust model with learning automata in order to evaluate trustworthy paths in online social networks for trusted-user recommendations. In addition, *Shannon's* entropy approach is presented to compute utility value for each trustworthy path.

The rest of this chapter is organized as follows. Motivation behind the work has been presented in Section 1.1. In Section 1.2.1, a learning automata based malicious social bot detection algorithm is proposed by integrating a trust model with a set of URL-based features in order to distinguish legitimate participants from social bots. Section 1.2.2 describes a deep Q-network based architecture by integrating single agent deep Q-learning model with social attributes for social bot detection. A multi-agent deep Q-learning model based on particle swarm optimization (PSO) method is also proposed for detecting social bots more accurately. Further, a top-k influential (user) algorithm is also presented in this section. Section 1.2.3 discuses the detection of social botnet communities more accurately in presence of different types of malicious activities in *Twitter* network. A discussion on the spread of spam content and identification of influential communities in Twitter network are also presented in this section. In Section 1.2.4, the importance of trustworthy paths in an online social network has been highlighted. A learning automata-based recommended trust path selection algorithm is also presented in this section. The organization of the thesis has been presented in Section 1.3.

1.1 Motivation and Objectives

In recent years, social bots are emerged as major threats in online social networks. Social bots usually perform malicious activities, such as generating fake identities, manipulating online ratings, spreading social spam content and performing phishing attacks. These types of malicious activities have to be detected and malicious participants should be identified in online social networks. However, social bots can pretend like legitimate participants in order to reduce the probability of identification. Social bots can reduce the trust value of their legitimate neighbors by sending malicious and untrustworthy information. In a typical social network, if neighbors of a participant are trustworthy, the participant is likely to be trustworthy. In this work, two trust parameters are introduced, namely direct trust (i.e., from users' own behavioral patterns while interacting in its neighborhood) and indirect trust (i.e., from belief values that are collected from the neighbors depending on their behavioral patterns) for social bot detection. Social bots can easily manipulate user profile-based features, such as follower ratio, URL ratio and number of retweets. Moreover, social bots can easily manipulate tweet-based features (such as sentimental words, emoticons and most frequent words used in the tweets) by changing the content of a tweet. Especially in *Twitter*, the size of tweet is limited upto 140 characters, social bots may also post shortened malicious *URLs* in the tweets in order to redirect participants (users) to some malicious servers. This leads to phishing attack. Moreover, spam bots are the bots who mainly spread spam-content (i.e., any tweet which is irrelevant or unnecessary information that is being repeatedly sent to user) by sharing, liking or retweeting spam posts [15]. Due to the presence of spam bots, legitimate users may be influenced by fake information (which is repeatedly sent to a legitimate user). Further, the social botnet community detection is an important research challenge where the coordination and cooperation among the social bots (i.e., botnet or bot community) may create strong malicious activities, there by breaking the security perimeter of users. Thus, the present work focuses on the above mentioned observations to detect malicious social bots and social botnet communities more accurately in online social networks by considering important features, such as tweet content, URL-based, graph-based, profile-based, temporal-based and similarity-based features.

In the era of social media, it is demanding to extract trust information and finding trustworthy participants in online social networks. The conventional approaches (like content-based recommendation models) consider social relationships based on comments provided in online social networks. These types of approaches are not taken into consideration for the establishment of social relationships among participants. Moreover, an attacker may act unethically and gets good reputation. Once an attacker gets high trust value then the attacker may provide untrustworthy recommendations. Therefore, social trust information along with recommended influence value of a user (i.e., user ratings) are considered for finding a social trustworthy path. In real-time, certain service providers select a few malicious participants to provide faulty decisions to the services (i.e., trust formation and recommendations) of the other participants. Moreover, the service providers award significant ranking (i.e., increase of ratings) to their own services for selective decision making. Therefore, the *social trust information* should be taken into consideration for avoiding such malicious comments posted in online social networks. The above mentioned challenges motivate the present work towards detecting malicious social bots and determining trustworthy paths in an online social network. The objectives of this dissertation are as follows.

1. Detection of phishing bots using learning automata model with *URL*-based features in *Twitter* network.

2. Detection of malicious social bots using single and multi-agent deep Q-learning models with a set of tweet-content, user-profile, graph-based features and a set of temporal features with spam-content, respectively.

3. Detection of social botnet communities with different types of malicious activities using a deep autoencoder model and behavioral similarity parameters and further detection of spam influential communities in *Twitter* network.

4. Design of a learning automata-based trust model to identify trustworthy paths in online social networks for trusted user recommendations.

1.2 Overview of the Contributions of this Thesis

In this section, an overview of chapter-wise contributions of this thesis has been presented. Each subsection presents summary of contributions of the corresponding chapter.

1.2.1 Detection of Phishing Bots using Learning Automata with URL features in Twitter Network

In this work, *URL*-based features, such as *URL* redirection, frequency of shared *URLs* and spam content in a *URL* are considered for identifying trustworthy participants (users) in *Twitter* network. Let $G = (P, E)$, where *P* represents a participant set $P = \{p_1, p_2, \ldots, p_n\}$ and *E* represents a social relationship set (or directed edges) between the participants (users). If there exists a social relationship between two participants, then they are considered as neighbors (i.e., either followers or followees). In a typical *Twitter* network with

n participants and series of m tweets $tw_{p_i} = \{tw_{i1}, tw_{i2}, ...tw_{im}\}$ posted by each participant p_i, a feature set $F = \{f_1, f_2, ..., f_n\}$ can be constructed from each tweet posted by each participant. In this work, the features are assumed to be independent to each other. Based on the *URL*-based features, the trust parameters are defined in order to evaluate trustworthiness of all tweets posted by each participant.

In *Twitter*, when a participant (user) wants to share a tweet containing *URL(s)* with the neighboring participants (i.e., followers or followees), the participant uses *URL* shortened service (i.e., bit.ly [16]) in order to reduce the length of *URL* (because tweet is restricted upto 140 characters). Moreover, a malicious social bot may post shortened phishing *URLs* in the tweet [11]. When a participant clicks on a shortened phishing *URL*, the request of a participant will be redirected to intermediate *URLs* associated with malicious servers. This in turn redirect the user to malicious web pages. Thus, the legitimate participant is exposed to an attacker. This leads to *Twitter* network suffering from several vulnerabilities (like phishing attack) [17]. Therefore, detection of social bots who post malicious *URLs* in the tweets is a challenging task in *Twitter* network.

To address the above challenges, the malicious behavior of a participant is analyzed by considering features extracted from the posted *URLs* (in the tweets), such as *URL* redirection, frequency of shared *URLs* and spam content in *URL*, to distinguish between legitimate and malicious tweets. The proposed *Learning Automata based Malicious Social Bot Detection (LA-MSBD)* algorithm integrates a trust computational (i.e., evaluation) model with a set of *URL*-based features for detection of malicious social bots. The proposed trust computation model contains two parameters namely, direct trust and indirect trust. The direct trust value is derived from *Bayesian* learning [18] (by considering *URL*-based features) to determine trustworthiness of tweets posted by each participant. In addition to direct trust, belief values (i.e., indicators for determining indirect trust) are collected from multiple neighbors of a participant. This is due to the fact that in case the neighbors of a participant are trustworthy, the participant is likely to be trustworthy. Further, the *Dempster's* combination rule [19] aggregates the belief values provided by multiple 1-hop neighboring participants in order to evaluate indirect trust value of participants in *Twitter* network. Moreover, in this work, the belief values provided by multiple neighboring participants are considered to be independent.

Learning Automata based Malicious Social Bot Detection using Trust Model

A *Learning Automata based Malicious Social Bot Detection* algorithm (LA-MSBD) has been proposed by incorporating a trust evalaution model in order to identify the malicious social bots. Each participant (i.e., each user in Twitter) is represented by a learning automaton in order to determine the trust value of a participant' at time t. At each iteration, learning automata selects a specific action from finite set of actions (i.e., a series of tweets posted by the participant at different time slots) and produces response (or reinforcement signal) in terms of reward and penalty. The proposed *LA-MSBD* detects a participant as a malicious social bot only after executing finite number of learning actions at different time slots.

The performance of the proposed *Learning Automata based Malicious Social Bot Detection* algorithm (*LA-MSBD*) is presented by considering *Social Honeypot* dataset [20] and *The Fake Project* dataset [21]. The proposed *LA-MSBD* algorithm has been compared in two different ways: (i) *LA-MSBD* algorithm with four conventional machine learning algorithms and (ii) *LA-MSBD* algorithm with the existing social bot detection algorithms, such as random forest-based spam detection [22] and neural-network based redirection spam detection (*NN-RS*) [23]. For *The Fake Project* dataset and *Social Honeypot* dataset, the highest precision level of the proposed *LA-MSBD* algorithm is obtained as approximately 95% and 90%, respectively. For *The Fake Project* dataset, the recall (true positive ratio) of the proposed *LA-MSBD* algorithm is found to be around 96% and the recall of the existing algorithm [22] is found to be around 91%.

1.2.2 Detection of Malicious Social Bots using Single-Agent and Multi-Agent Deep Q-Learning Models in Twitter Network

In this work, 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 weekly tweets posted by each user) and social graph based attributes (i.e., the users' interaction with their friends and followers) are considered to identify the suspicious behavior of social bots. Based on these social attributes, the deep Q-learning elements namely, a state vector S_t, a learning action α and reward r_t are defined for each participant (or user) in *Twitter* network. Malicious bots are the bots who mainly spread spam-content (i.e., any tweet which is irrelevant or unnecessary information that is being repeatedly sent to user) by sharing, liking or retweeting spam posts [21], [15]. Moreover, in order avoid detection, malicious bots show variation in posting positive (or negative) sentimental tweets over time. Users' temporal features, such as variation in posting positive (or negative) tweets over time, percentage of dropped followers and spam content in the tweet are considered in order to analyze malicious user's behavioral patterns.

In [24], malicious users have strong intention to manipulate the data (which is used by supervised machine learning algorithm for training the data) in order to avoid detection. Thus, this may lead to misclassification for new sample of data during testing phase. However, the recent studies have illustrated that supervised machine learning algorithms fail to detect social bots in certain situations, such as when training data is more biased [25]. In deep reinforcement learning (*DRL*) techniques (like deep *Q*-learning) with single agent requires much computation time in order to determine an optimal policy [26]. Moreover, the learning process of *DRL* requires more computational resources compared to other machine learning algorithms. The deep *Q*-learning converges slower with high computation and storage space in order to determine and store the *Q*-values for all possible state-action pairs [26]. Therefore, finding an optimal sequence of actions in *Q*-learning (with faster convergence rate) is a challenging issue. Thus, in this work deep reinforcement learning algorithms (with single and multi-agent) has been presented in order to detect malicious social bots more accurately.

A Single Agent Deep Q-Learning Model for Detecting Social Bots

In the proposed deep Q-learning model, 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. For each user, the social attributes are represented in the form of state vector S, which contains a set of states (i.e., social attributes). For each state-action pair, the system (i.e., deep *Q*-learning model) determines next state and reward function (i.e., social behavior of user). After finding the state-action pair, the agent decides whether the corresponding user is acting as a malicious social bot or a legitimate user.

The performance of the proposed *Deep Q-Learning* algorithm is evaluated in terms of precision, recall (true positive rate), false positive rate and f-measure by considering three real-world datasets from the *Twitter* network, such as *The Fake Project* dataset, [21], *Social Honeypot* dataset [20] and *User Popularity Band* dataset (i.e., the dataset is partitioned into four groups based on number of followers) [27]. The proposed *Deep Q-Learning* algorithm has been compared with the other existing algorithms, such as feed-forward neural network (*FFNN*) [28], deterministic *Q*-Learning (QL) [29] and regularized deep neural network (*RDNN*) [30]. For social bot detection, the proposed algorithm with the tweet-based attributes achieves average of 85% on the precision value, the proposed algorithm with the user profile-based attributes achieves average of 87% on the precision value and the proposed algorithm with the social graph-based attributes achieves average of 88% on the precision value. Therefore, by integrating all the above social attributes, the proposed algorithm have achieved average of 93% on the precision value.

A Multi-Agent Deep Q-Learning Model using Particle Swarm Optimization for Detecting Social Bots

In the proposed particle swarm optimization based deep Q-learning (P-DQL) algorithm , the users' temporal features such as spam-content in the tweet, average number of tweets posted per day, longest user session time without any break and percentage of dropped followers) are represented as state vector S_k. Initially states are given as input to the learning agent (i.e., deep Q-network) in order to obtain Q-values $Q(s_i, A)$ with action sequences $A = \{a_1, a_2, ...\}$ for each state s_i (where $s_i \in S_K$). Each user with set of Q-values is termed as a swarm with set of particles (or a population) and each Q-value $Q(s_i, a_i)$ (i.e., where $a_i \in A$) represents a particle's position. For each particle (i.e., $Q(s_i, A)$), the fitness function (i.e., long-term immediate reward R^{long}) is to be computed in order to determine local and global best particle's positions. Based on local and global best particle's positions, the position and velocity of particles are updated to determine global best action sequences (i.e., swarm updated). Moreover, the entire process will be executed for finite number of iterations until the variation of fitness values becomes negligible (i.e. fitness value unchanged in consecutive iterations). Therefore, the particle swarm optimization (*PSO*) component determines global best action. The learning agent takes each state from global best action and obtains belief-based reward value in order to compute Q-value and target Q-value. After executing a specific action in a state, the learning agent moves to the next state (i.e., which is available in global action sequences) and obtains belief-based reward value. For all possible global action sequences, if the learning agent cannot reach to a terminal state (i.e., identified as a spam bot based on its social behavior) then the participant is identified as a legitimate. Otherwise, user is identified as a spam bot.

The performance of proposed particle swarm optimization based deep Q-learning (P-DQL) algorithm is evaluated by considering two real-time Twitter datasets, such as Social Honeypot dataset and *The Fake Project* dataset. For social spam detection, the proposed *P-DQL* has been compared with adaptive single-agent deep Q-learning (*ADQL*) algorithm and with other existing algorithms such as, *PSO* algorithm [31], feed-forward neural network (*FFNN*) [28], regularized deep neural network (*RDNN*) [30]. The precision values of *P-DQL* and *ADQL* are obtained as approximately 94% and 89%, respectively.

Influence Bots in Twitter

The proposed top-k influential algorithm is used to identify the most influential users (which are influenced by the social bots) based on the tweets and the user's interactions in *Twitter* network. For each user, a user influence score is determined based on two parameters namely, influence of user's tweets and influence of user's interactions in the *Twitter* network.

The proposed top k-influential users algorithm has been compared with other existing algorithms, such as degree centrality based radius-neighborhood (*DERND*) [32], suspected infected recovered (*SIR*) diffusion model [33] and true-top [34]. The performance of the proposed top k-influential users algorithm is evaluated in terms of precision and recall (true positive rate). The proposed algorithm identifies 80% of top-10% influential users which were influenced by the social bots.

1.2.3 Detection of Social Botnet and Spam Influential Communities in Twitter Network

In this work, a *Deep Autoencoder based Social Botnet Community Detection* (DA-SBCD) algorithm has been designed to detect social botnet communities more accurately with different type of malicious activities including sybil bots. Sybil bots are bots who create multiple fake accounts in order to influence legitimate participants [35]. In real-time certain service providers may create multiple fake accounts (i.e., sybil bots) in order to provide faulty decisions to other services and provide significant ranking to their services. The presence of sybil bots may mislead a legitimate user to be influenced by fake information. In this work, a trust-driven random walk model is presented to distinguish legitimate participants among social bots in *Twitter* network. User behavioral similarity parameters (such as tweet-content similarity, URL-shared similarity, interaction similarity and interest similarity) are considered in order to identify similar type of (malicious or non-malicious) behavior among the participants.

Social Botnet Community Detection Algorithm

In the proposed algorithm, the weighted eigenvector centrality measure and friendship-characteristics of communities are considered to detect the presence of a botmaster and social botnet communities, respectively. The proposed algorithm consists of two phases – community formation phase and community reconstruction phase (which identifies the communities more accurately). In the first phase, the weighted signed *Twitter* network graph G' is used for detecting social botnet communities with different types of malicious activities (such as posting malicious tweets, posting or redirecting to malicious *URLs* and creating multiple fake identifies). In the second phase, the proposed architecture is integrated with deep autoencoder model consisting of two sub-phases, namely the encoder and decoder. The proposed model encodes an observed input community c_i with the set of trusted and untrusted weighted edges. In the decoding sub-phase, a reconstructed community structure $\tilde{c_i}$ is determined using the decoding function (i.e., $\tilde{c_i} \approx f(c_i)$) for social botnet community detection with better accuracy.

The proposed algorithm has been compared with two promising recent community detection methods, such as detecting spam communities (*SpamCom*) [36] and *Botnet Discovery* [37]. Two datasets, such as *The Fake Project* dataset [21] and *Social Honeypot* dataset [20] are considered for performance evaluation. The performance of the social botnet community detection algorithm is evaluated in terms of normalized mutual information (NMI), precision, recall, f-measure and g-measure. The proposed algorithm achieves around 90% on precision value and provides up to 8% improvement on the *NMI* value over existing social botnet detection algorithms.

Spam Influential Users and Influential Community Detection

To provide accuracy and veracity of information, identification and reduction of the influence of spam bots (i.e. reduction of negative impact of spreading spam content) is an important task in a *Twitter* network. Most of the existing works [38], [39] focus on spreading trustworthy information in order to reduce the influence of spam content (or fake information) and detect spam initiators (i.e. social spam bots) in online social networks. However, the amount of influence of spam bots on legitimate participants (by frequent interactions) has not been adequately addressed in the existing works[38], [39]. A *Spam Influential Users and Influential Community Detection (SIU-ICD)* algorithm has been proposed to detect the most influential participants (which are influenced by spam bots) in *Twitter* network in order to minimize the spread of spam content.

The proposed *SIU-ICD* algorithm has been compared with two existing algorithms, such as opinion spammer community detection (*OSCD*) [40] algorithm and spammer group detection approach (*SGD*) [41]. For The Fake Project dataset, the highest modularity $\tilde{Q}$ value obtained by *SIM-ICD* is 0.65. Moreover, the proposed *SIU-ICD* algorithm achieves 4-9% improvement on modularity $\tilde{Q}$ over existing spammer community detection algorithms.

1.2.4 Determining Trustworthy Paths using Learning Automata-based Trust Model in Online Social Networks

In this work, a *high-quality of social trust* model with learning automata has been presented in order to determine trustworthy paths in online social networks for trusted-user recommendations. In addition, *Shannon's* entropy approach is presented to compute utility value for each path. In an online social network, multiple recommended trust paths exist between a service provider and the consumers [42]. Moreover, an attacker may act unethically and gets good reputation. Once an attacker gets high trust value then the attacker may provide untrustworthy recommendations. Thus, determining a recommended trustworthy path is a challenging problem in social networks. In a service-oriented system, trust plays a major role for selective decision making and requires a methodology to evaluate the trust paths between the participants who are unknown to each other. A trust evaluation model is formulated with different parameters such as trust information (direct trust and indirect trust), social relationships and recommended influence of a participant for providing an accurate trustworthy recommendations. Further, this work focuses on formulating a trust model for establishing a strong social connection among a group of participants.

Learning Automata based-Recommended Trust Path Selection Algorithm

A *Learning Automata based-Recommended Trust Path Selection* (LA-RTPS) algorithm has been proposed by considering parameters, such as *direct trust*, *indirect trust* (T), *relevance degree* (r) and *recommended influence value* (ρ) of a participant . Each path is represented by a learning automaton in order to determine the trustworthy path at time t. At each iteration, learning automata selects a specific action from a finite set of actions (i.e., a finite set of intermediate participants) and produces response in terms of reward and penalty. The proposed *LA-RTPS* identifies a trustworthy path only after executing finite number of learning actions at different time slots.

The performance of *LA-RTPS* algorithm is evaluated by considering two datasets, such as *Slashdot* dataset and *Epinions* dataset [43] and compared with *MFPB-HOSTP* [44]. The experimental results show that the proposed *LA-RTPS* algorithm provides (recommended) trust path utilities (i.e., a metric in terms of trust, relevance degree and recommended influence value) on an average of 31.48% more than the existing *MFPB-HOSTP* approach. The average execution time of the proposed learning automata based approach is found to be 34.41% less than *MFPB-HOSTP* approach [44].

1.3 Organization of the Thesis

The main focus of this dissertation is to analyze user behavioral patterns and detection of social bots in online social networks. The proposed algorithms achieve improvement in precision, recall and F-measure for detecting social bots. The thesis has been organized into seven chapters.

Chapter 1: In this chapter, a brief introduction to security and threats in online social networks and objectives of the thesis have been presented. It also presents an overview of the major contributions and outline of the thesis.

Chapter 2: In this chapter, existing works on malicious activities and trust evaluation models in online social networks have been discussed. A survey on malicious social bot detection approaches is presented.

Chapter 3: A Learning Automata based Social Bot Detection (LA-SBD) model has been proposed for social bot detection with URL-based features. The trustworthiness of each tweet is evaluated by using *Bayesian learning* and *Dempster-Shafer Theory (DST*). This chapter is completely derived from the following paper:

R. R. Rout, G. Lingam and D. V. L. N. Somayajulu, ”Detection of Malicious Social Bots Using Learning Automata With URL Features in Twitter Network,” *IEEE Transactions on Computational Social Systems*, pp. 1004 - 1018, 2020.

Chapter 4: In this chapter, a single and multi-agent deep Q-learning models are designed to detect social bots. 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. This chapter is derived from the part of the work as presented in the following two papers:

G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, “Adaptive deep q-learning model for detecting social bots and influential users in online social networks," *Applied Intelligence*, Springer, pp. 1–18, 2019 and

G. Lingam, R. R. Rout, D. V. L. N. Somayajulu and S. K. Ghosh, "Particle Swarm Optimization on Deep Reinforcement Learning for Detecting Social Spam-Bots and Spam-Influential Users in Twitter Network," *IEEE Systems Journal*, Accepted.

Chapter 5: In this chapter, a deep autoencoder (DA) model, DA-Social Botnet Community Detection (DA-SBCD) algorithm has been proposed to detect social botnet communities consisting of social bots having higher malicious behavioral similarity. Further, an *Influential Community Detection (ICD)* algorithm has been proposed to reduce the spread of spam-content through influential communities in *Twitter* network. This chapter is derived from the part of the work as presented in the following two papers:

G. Lingam, R. R. Rout, D. V. L. N. Somayajulu, S. K. Das, "Social Botnet Community Detection: A Novel Approach based on Behavioral Similarity in Twitter Network using Deep Learning," *In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security 2020*, pp. 708-718 and

G. Lingam, R. R. Rout, D. V. L. N. Somayajulu, and S. K. Ghosh, "Particle Swarm Optimization on Deep Reinforcement Learning for Detecting Social Spam-Bots and Spam-Influential Users in Twitter Network," *IEEE Systems Journal*, Accepted.

Chapter 6: In this chapter, a Learning Automata based Recommended Trust Path Selection (LA-RTPS) algorithm has been proposed to identify multiple recommended trust paths in online social networks. A trust model named as *High quality of Social trust* (HoS) model has been presented to determine the best trustworthy path in online social networks. This chapter is completely derived from the following paper:

G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, "Learning automata-based trust model for user recommendations in online social networks," *Computers & Electrical Engineering*, Elsevier, pp. 174-88, 2018.

Chapter 7: This chapter summarizes the outcomes of the contributions and future directions for expansion of the work.

Chapter 2

Literature Survey

In this chapter, functionalities in different types of online social networks are discussed. Literature survey on data representation (including features in online social networks) has been discussed. Further, existing works on security issues (which includes malicious activities) and necessity of trust in online social networks are also discussed. A discussion on social bot detection approaches and requirement of learning algorithms has been included in this chapter.

2.1 Types of Online Social Networks

Online social networks such as *Twitter*, *Google+*, *Facebook* and *Instagram* have become most important social media for people to share their information, opinion(s) and to promote their products and services. Online social networks provide a communication platform where users can interact with their friends (or neighbors) in terms replies, comments and posts (or re-posts) [45]. Fig. 2.1 shows a simple social network with set of nodes and edges. The node represents a user (or participant) and edge represents the social relationship between two users. Based on social relationships, the edge can be represented as undirected edge, directed edge and signed directed edge (as depicted in Fig. 2.1). Online social (static or dynamic) communities are formed with people who have similar type of interests or opinions on a specific topic or an event [46]. Online social networks are mainly classified into three different types namely, social connections (such as Facebook, Twitter, Google+ and MySpace), multimedia sharing (such as YouTube and Flickr) and professional (like LinkedIn).

Social Connections: Interacting with friends and family members is one of the impor-

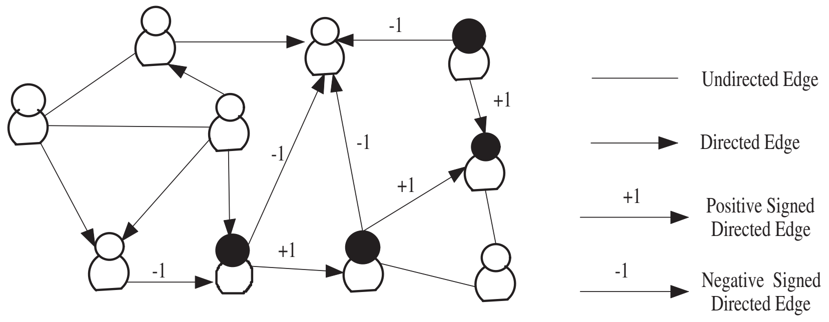


Figure 2.1: An online social network where directed edge represents interactions between two neighboring participants and signed directed edge represents type of relationship (i.e., trusted or untrusted edge) between two participants

tant characteristics of online social networks. The following are the popular online social networks which establish social connections with other online users:

1. *Twitter:* Twitter is an online social networking service on which users can post tweets (or messages), retweet tweets and interact with other online users by expressing their views or opinions. Moreover, the users can post a series of tweets on specific topic or by prefixing hashtag (i.e., #) symbol with word. Similarly, username prefixed with '@' symbol is used for replying or mentioning to other Twitter users.

2. *Facebook:* Facebook provides an online communication platform for users to establish social connections and tries to share information with other Facebook users. In Facebook, users can follow the content posted by other Facebook users without necessarily being a friend (or follower). Moreover, Facebook restricts the users to view their profiles and posts through privacy settings.

3. *Google+:* Google+ provides a platform to upload photos to private cloud album and allows users to create circles of social connections. Circle is a main feature provided by Google+ social network. Moreover, once circle is created then the user can share his/her (private) information only to that circle.

4. *MySpace:* MySpace social network provides a platform to establish social connections related to social entertainment, such as music, movies and games.

Multimedia Sharing: Online social networking websites are providing a platform to upload, view and share video content in online. The following are the most popular online social networks for sharing multimedia information.

1. *YouTube:* YouTube is an online social networking platform which allows users to upload, rate, view and share video content information. Moreover, YouTube allows users to post comments on video and provides subscription option to other users.

2. *Flickr:* Flickr provides a social networking platform to upload and share images (or videos) with other users. Moreover, Flickr provides two types of online social networking user accounts namely, free and pro.

Professional: Professional online social networks provide a platform to establish social relationships with other professional users based on interests or profession. Moreover, professional networks also provide employment-related services (such as job applicants post their curriculum vitae and recruiters post jobs in professional social networks to employ potential candidates). LinkedIn is one of the largest online professional social networks.

In this thesis, Twitter network has been investigated in Chapter 3, Chapter 4 and Chapter 5. In Chapter 5, a weighted signed Twitter network graph has been considered based on the behavioral similarity and trust values between the participants (i.e., OSN accounts) as weighted edges.

2.2 Representation of Data with Features in Online Social Networks

Online social networking user profile contains several distinct features, where the shared data can be represented as features. Moreover, each feature describes about the social behavior of user. For Twitter network, the features are broadly classified into five categories, namely user profile-based, tweet-based, graph-based, temporal-based and similarity-based features.

User profile-based features represent the characteristics behavior of each user, such as location, followers ratio, number of tweets, retweets, likes, comments, followees and followers. Ala et al. [47] have proposed a support vector machine-based optimization algorithm by considering user profile-based features to detect spam profiles in online social networks. Lin et al. [48] have presented a convolutional neural network model with user profile-based features to detect users' stress state in online social networks. In [49], a deep neural network based on long short-term memory (LSTM) architecture is designed by considering user profile-based features to detect social bots in Twitter network. Al et al. [50] have presented user profile-based features to identify the malicious activities in online social networks. Subrahmanian et al. [51] have presented user profile-based features to detect social bots and influence bots in Twitter network.

Tweet-based features describe about syntax, semantic behavior of content in the tweets and URL-based features. The syntax-based features represent percentage of tweets containing the number of links, user mentions, hashtags and special characters ('@', '\$', '%') and emoticons. The semantic-based features represent user's sentimental score, number of languages in which tweets are posted, suspicious words, frequent words, number of positive and negative sentimental in the tweets. Moreover, URL-based features are based on URL redirection chains and lexical properties of URL such as frequency of shared URLs, URL redirection length, relative position of initial URL, http-302 status code and spam content in URL. In [51], the authors have presented tweet syntax and semantic-based features to detect bots in online social networks. Chu et al. [13] have presented an automated classification system using tweet-based features to distinguish benign users among malicious users in Twitter network. In [52], a *URL*-based approach is proposed to detect spam tweets in *Twitter* based on the tweet content and *URL* redirection chains. Hans et al. [23] have detected malicious URL redirections by integrating multilayer perceptron neural network with URL-based features in online social network.

Graph-based features describe about social relationships among the users. For each user, the graph-based features represent clustering coefficient, closeness, betweenness and pagerank centrality measures. In [53], a stegbot detection method is proposed by considering social graph-based features to detect stegbots in multimedia social networks. In [10], a botnet detection approach has been proposed based on graph-based features. Yan [12] has identified different types of malicious users based on graph-based features in online social networks. Liang et al. [54] have extracted social network-topology based and tweet-content based features to develop a Bayesian classifier model which helps o detect rumours in Twitter network.

Temporal-based features contain longest user's session time without any break for at least 5-10 minutes, percentage of dropped followers, average number of tweets posted per day, average time between two consecutive tweets, temporal patterns of posting tweets (or retweets) and inter arrival time between user's click events. Ferrara et al. [55] have analyzed the social behavior of user by considering temporal-based features in Twitter network. Shi et al. [8] have presented temporal-based features and user behavioral transition probability features (such as sharing, liking and commenting) to detect malicious users in online social networks. In [8], it has been shown that quantitative features (like number of hashtags, number of replies and number of comments) help in detecting malicious users.

Similarity-based features are analyzed from the viewpoints of tweet-content similarity, shared URL similarity, interest similarity, and social interaction similarity for identifying similar types of behavior among users in the Twitter network. Davoudi et al. [56] have considered interaction similarity between two users for predicting trustworthy ratings in online recommended system. Zhao et al. [57] have presented a hashtag-based user similarity ranking method to identify the most similar users using Latent Dirichlet Allocation (LDA) method. Further, the authors have proposed a hashtag-based *LDA* model to identify the social relationship between users, topics and hashtags in the tweets. In [54], the authors have presented tweet-content similarity measure using term frequency-inverse document frequency (TF-IDF) for the identification of rumour spreading in Twitter network.

In this thesis, URL-based features are considered in Chapter 3 and Chapter 5. In Chapter 4 and Chapter 5, user profile-based, tweet content-based and graph-based features are taken into consideration to analyze the social behavior of users. Moreover, temporal-based and similarity-based features are considered in Chapter 4 and Chapter 5, respectively.

2.3 Security Issues and Types of Attacks in Online Social Networks

In recent years, most of the users published their daily activities and share their information with friends, family members and colleagues in online social networks. Moreover, user generated data may contain opinions, past experiences and personal information (such as name, location, personal photos, e-mail address and gender). However, such information can allow an attacker to steal user's credential details. In addition, user-generated data may be provided to other social applications (like third-party platforms) and it leads to privacy issue in an online social network. The following are the challenging issues related to protect online users' and their data sharing with other users in online social networks.

1. *Data sharing with blind social connections:* Users may be at risk by sharing their personal information with unknown users (or more specifically with strangers). Moreover, some of the users may not be legitimate accounts (i.e., for example automated accounts which are created by malicious user) or users with malicious intention.

2. *Data sharing with third party based social applications:* Users may interact with several other external applications for desirable purpose. Moreover, malicious third party based social application can access users' data for performing malicious activities.

3. *Data leakage through crawlers and online social networks:* Professional data collectors may crawl users' information through application programming interfaces (APIs) provided by online social networks. Professional data collectors may sale users' information to insurance companies and other online rating agencies. There is a requirement of trusted communication and trust evaluation for personal data sharing in online social networks.

In recent years, various threats such as privacy violations, malware, information leakage and fake profiles (termed as sybils or social bots) are observed in online social networks [1], [58], [59]. In [58], [59], the authors have found that the online social networking

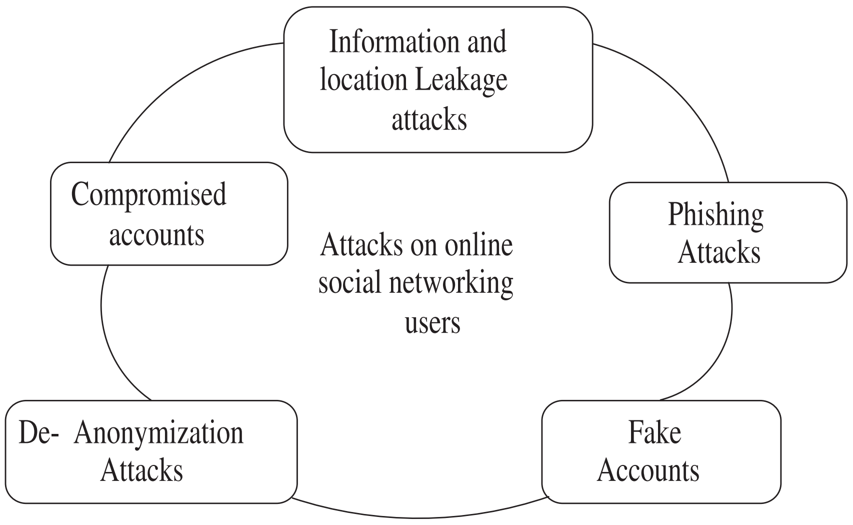

Information and location Leakage attacks
Compromised accounts
Attacks on online social networking users
Phishing Attacks
De- Anonymization Attacks
Fake Accounts

Figure 2.2: Attacks on online social networking user accounts

users reveal their personal information such as date of birth, phone number, email and hometown address. A study by Boshmaf et al. [60] found that Facebook users accept friend request from other unknown users when they have mutual connections. However, by accepting such friend request from unknown users may lead the users to reveal their personal information to unknown users or strangers. Hence, this lead to privacy and leakage problems in online social networks. Moreover, the attacker may have strong motivation to perform malicious activities and there by breaking the security perimeter of users.

A study in [61] found that social network users trust most of social networking websites and they trust other social networking users. Therefore, this trust will lead to establish new social relationships and information sharing among users. A study by Niu et al. [62] found that the trusted behavior of social relationships has became an essential way of spreading social spam content (or malware) and executing phishing attacks. Malware is a malicious software designed to obtain user's credential details and to access private information. Koobface is a malicious software designed to spread fake information to the friends of online social networking users by stealing credentials [63]. In online social networks, malicious users initiate different type of attacks, such as spreading spam content, creating multiple fake accounts, executing phishing attacks and manipulating user opinions. Fig.2.2 shows different type of attacks that can be performed on online social networking user accounts.

Compromised accounts: Compromised accounts are originally created by legitimate users but they are hacked and controlled by malicious users. For example, when a legitimate user's account is hacked then the malicious user starts spreading fake information to followers by acting like legitimate user. In [64], the authors have analyzed the malicious behavior of compromised accounts through phishing attack on Twitter network. However, the proposal method fail to detect the compromised accounts which does not perform phishing attack. Egele et al. [65] have designed a system to detect the compromised accounts in online social networks. The authors have tried to determine whether a fake information is posted by an attacker through a compromised account.

De-Anonymization Attacks: In online social networks like Twitter, users' anonymity and privacy can be protected by considering pseudonyms. The de-anonymization attacks consider different methods, such as network topology and capturing cookies to detect the real identification of each user in online social networks [66]. In [67], the authors have presented a technique to de-anonymize online social networking users by capturing their cookies and obtaining group memberships for each user account. Peled et al. [68] have proposed a novel method to detect the real identification of user based on matching user profiles across multiple online social networks.

Information and location Leakage attacks: In recent years, due to the usage of smart mobile devices, most of the users are willing to share their private, personal and location information with their friends and other users in online social networks [58]. A study by Torabi et al. [69] have found that most of users are sharing their health information through online social networks. Further, the authors have observed that few insurance companies are using leaked health-related information to identify clients with health conditions either to deny or increase their premiums. Li et al. [70] have found that most of the Twitter users are mentioning their location information in the tweets. In [71], a framework has been proposed to identify the user's hometown location information based on the content of tweets posted by each user.

Fake Accounts: Fake accounts (sybils or social bots) are automated online social networking accounts which pretend like social behavior of legitimate users in online social networks. Social bot is an automated computer program that is created to perform (either malicious or non-malicious) activities in *Twitter* network [4]. Like traditional bots (in *Internet* chat), social bots are more common in *Twitter* [72]. A social bot is created with the support of open *APIs* (like *Twitter API*) [5]. Moreover, social bots are mostly used for posting spam tweets, retweets and sharing public opinion in *Twitter*. The recent studies have identified different types of social bots, such as legitimate bots and malicious bots [6]. Legitimate bots are used to promote products or services, natural disaster notifications and blog updates. Malicious (or spam) bots are mostly used to distribute spam content, phishing *URLs*, generate fake accounts and manipulate online reviews and ratings [2]. However, malicious social bots can also manipulate natural disaster notifications and quality of product by posting fake information or malicious comments [8]. However, such type of malicious activities can affect online social networks.

A study by Stringhini et al. [73] found that the fraudsters who are selling the legitimate online social networking accounts created in *Twitter* network. Moreover, if a social bot (or malicious user) is willing to buy the legitimate accounts from fraudsters, then the attacker can compromise a larger number of legitimate friends by creating attack edges between the fake accounts and legitimate accounts. Boshmaf et al. [60] have created more than a hundred of Facebook fake accounts (or social bots) to attack Facebook legitimate accounts by sending multiple friend requests. Moreover, the authors have showed that the acceptance rate of social bot friend request is about 80% when there are common friends between social bots and Facebook legitimate users.

Phishing Attacks: In Phishing attack, the malicious user attempts to obtain user's credentials and personal information by pretending like trusted third party. In [74], the authors have found that the legitimate Twitter users who click on shortened URLs are more likely to fall under phishing attack due to the trusted behavior of online social networks. Lee et al. [52] have showed that 85% of phishing attacks target online social networking users. Moreover, a recent study by Niu et al. [62] found that phishing attacks are increased on online social networks by posting suspicious hyperlinks in messages.

Several spam detection approaches have been proposed in *Twitter* network to distinguish non-spam accounts and spam accounts [75], [76], [77], [78]. Moreover, these studies consider user profile features which can easily be modified by malicious bots. To avoid feature manipulation, Yang et al. [79] have considered social relationships between malicious users and with their neighboring users based on closeness centrality. Moreover, profile features and social interaction features may not help in detecting malicious *URLs* which are posted by the participants. Attackers may use malicious *URL* redirection chains in order to avoid detection. Thus, spammers can attack legitimate users by misleading detectors.

In this thesis, URL-based features are considered in Chapter 3 and Chapter 5 to detect malicious social bots through malicious URL redirections. In Chapter 4 and Chapter 5, user profile-based, tweet content-based and graph-based features are considered to detect malicious social bots with spam behavior. Moreover, similarity-based features are considered in Chapter 5 to detect sybil bots (or multiple fake accounts).

2.4 Trust Computational Models in Online Social Networks

Online social networking participants (or users) are unaware of online social threats such as malware attacks [80], phishing attacks [23] and fake profiles (or social bots [81], [8] or sybils [82] [35]). Recently, online social networking sites (or communities) contain tremendous data, such as online reviews, online ratings and discussions which are generated by users (in various communities).

In the literature, various trust evaluation models have been proposed for determining and predicting the trust value. Trust is one of the important aspects to improve the quality of social relationships among users in online social networks. Trust is defined as degree of user's belief based on previous experiences on a specific context, other user's recommendations and relationships [83]. Moreover, trusting a user may affect other users' opinion, reputation and selective decision making on specific product or service. To evaluate trust value using the current experiences can be considered rather than considering only the past experiences. Moreover, trust value changes over time. Thus, trust is time-dependent and dynamic in nature [61]. When the information (or data) is shared among users in online social networks, malicious users take the advantage of information sharing in order to spread the spam content (or fake information) in online social networks. Trust modeling helps to provide veracity of information and helps to penalize the malicious users (or social bots) who try to destruct the system with false information [84]. The trust computational models are broadly classified into two categories, namely evaluation of trust based on social relationships and topic-based trust modeling.

2.4.1 Evaluation of Trust based on Social Relationships

The works presented in [83], [85], [86] are based on social trust relationship models in online social networks. Golbeck et al. [83] have proposed a trust mechanism by inferring binary relationship between two individual participants in a web-based social network. A trust model has been proposed by Walter et al. [85] to integrate dynamic trust value among the participants. The authors have identified two factors: (i) heterogeneity preferences and (ii) knowledge deficiency among the participants. In [86], the authors have analyzed that social relationships including recommendations have a significant impact on the participants in selective decision making for trustworthy services. In [87], Gong et al. have proposed multi-path trust aggregation model by considering weight of path length and trust quality. The trust aggregation model considers direct trust, inter-node relationship and recommended social trust influence value to measure the degree of belief between two users.

In [88], Hamzelou et al. have proposed a model to prevent cascading trust failures in online social networks. The proposed model have considered parameters, such as cascading time, changes in social network topology and connectivity ratio to determine the trust relationship between two users. Further, the proposal has been evaluated by considering trust parameters, such as user's behavioral trust on information propagation and user's emotional sensitivity. In [3], trust-oriented social influence evaluation method has been designed to provide accuracy and veracity of information for selective decision making by considering trust relationship between users, user preferences and social relationships. Wu et al. [89] have constructed a social trust relationship model based on trust score and user preferences in an online social network. Further, the authors have presented a visual group interaction model with trust propagation for selective decision making. In [90], a social trust relationship model with nonlinear optimization technique has been proposed by Liu et al. to detect and eliminate conflicts for selective decision making in social networks. Tan et al. [91] have presented indirect trust model to determine the veracity of good recommendations based on users similarity with one-hop and multi-hop recommendations. Cheng et al. [92] have derived a social trust relationship-based network from Bayesian learning to predict the users' preferences and personalized user-item recommendations.

Recently, user generated data items in online social networks originate the new age of *Big Data* problems [42]. The huge volume of data cannot be processed or analyzed efficiently using statistical tools or traditional data analytic methods. *Big Data* creates many challenging research issues in the context of online social networking analysis [43]. In the new era of *Big Data*, it is challenging to identify the most relevant trust information in online social networks. In [93], a geometric differential learning model has been proposed to handle multimedia *Big Data* in online social networks for video recommendations.

Jamali et al. [94] have proposed a trust walker model for building a trust-based recommendation system. This model combines trust value with the item-based collaborative filtering approach in order to build a recommender system. *Small Blue* [95] is a networking application where upto six hops can be selected in order to find the shortest path between a source and target participants. However, in this application trust value and the participant preference value are not taken into consideration for evaluating a trustworthy service. Hang et al. [96] have proposed a trust path selection approach, where belief is considered as a most relevant trustworthy service. Eirinaki et al. [97] have evaluated trust value of each user based on recommendations received through user interactions. For each user, the authors have identified trustworthy and untrustworthy social relationships in OSNs. In [98], a mechanism based on indirect trust has been presented for removing the untrustworthy recommendations. However, the recommended trust value has not been considered. In the above mentioned approaches, although the trust value is taken into account, they are not applicable to determine trustworthy decision making in online social networks.

2.4.2 Topic-Focused Trust Modeling

Zhao et al. [61] have proposed a novel topic-based trust model to determine the trustworthiness of tweets posted by user in Twitter network. Further, the authors have proposed trust propagation algorithm by considering semantics, social and contextual relationships in Twitter network. Gupta et al.[99] have designed a support vector machine based Tweet-Cred system to evaluate the credibility of each user's tweet. In [100], a semi-supervised framework have been proposed to detect trustworthy users based on profile-based, content-based and graph-based features in Twitter network. Further, the authors have proposed a feature-based ranking trust model for trust propagation on social network graph. Wang et al. [101] have proposed content-based trust model by considering textual features (such as average length of words, replication of text content and fraction of most common words used in the content) and quality-based features (such as popularity, cohesiveness and accuracy) to evaluate the trustworthiness of each user in online social networks.

Kang et al. [102] have evaluated trustworthiness of each tweet based on content and user-level features. The authors have also evaluated trustworthy ratings for each user based on topic based behavioral patterns and retweeting behavior. Todd et al. [103] have evaluated trust value through a n-gram classifier model by considering context and user's meta-data features such as, number of tweets, followers and friends. Castillo et al. [104] have considered a classification model to distinguish tweets as trustworthy and untrustworthy based on propagation and content based features. In [105], a trust model has been designed to detect malicious activities in an online social network. The author have analyzed that low trust value of user indicates that the information spreaded by the user is considered as untrustworthy.

Alrubaian et al. [84] have evaluated trust value based on trustworthiness of content, user expertise and user reputation in order to classify tweets as credible and non-credible tweets. The authors have considered two-topic based datasets (which are crawled from Twitter network) to evaluate the performance of the proposal model. Ikegami et al. [106] have considered a topic-based and opinion based classifier model to evaluate the trustworthiness of tweets posted by each user. Further, the authors have identified topics by considering latent Dirichlet allocation (LDA) and applied sentiment analysis to evaluate opinion of tweet as positive or negative. In [107], Gupta and Ponnurangam have applied different statistical techniques such as linear regression and logistic regression to evaluate and predict the trustworthiness of tweets based on content and user profile-based features.

In [108], a cognitive model is used to detect untrustworthy or fake information spreading in Twitter network. The cognitive model is evaluated based on source credibility, coherence and consistency of tweet.

Due to the availability of tremendous data has attracted an attacker to steal user's personal information and to perform malicious activities (generating fake identities, manipulating online ratings, spreading social spam content and performing phishing attacks) in online social networks and this leads to vulnerabilities [1]. Moreover, such huge amount of data may also contain untrustworthy, fake or irrelevant information. However, identifying untrustworthy information manually is a difficult task. Building trust allows online social network users to gain good recommendations, credible opinions and online ratings (and reviews) [3]. Therefore, trust is used to protect against the attacks (such as social botnet attacks and phishing attacks) and to improve the security in online social networks.

In Chapter 3 and Chapter 5, trust computational models are presented to evaluate trustworthiness of tweets posted by each participant in Twitter network. In Chapter 6, a *High quality of Social trust (HoS)* model is designed to evaluate trustworthy services in online social networks by incorporating attributes, such as trust information (direct trust and indirect trust), social relationships and the participants' recommendations.

2.5 Types of Malicious Activities in OSNs

In online social networks, malicious social bots perform malicious activities, such as *generating fake profiles (or sybils)*, *posting fake (or spam) content*, *posting malicious URLs (or links)* and *fraudulent reviews*. Moreover, each malicious activity can be identified through different techniques which are discussed in the following subsections.

2.5.1 Fake Profiles (or Sybil) Detection

The existing works [109], [82], [110], [111], [35] are based on random walk framework (i.e., each participant moves to one of its neighboring participants with equal probability) to detect sybil users (fake multiple identities), which are under the control of single malicious user. *SybilGuard* approach [109] has been proposed by Yu et al. to control the influences of sybil users in social network. SybilGuard depends on fast mixing properties of social graph and malicious users can generate multiple fake profiles but with limited attack edges (i.e., an edge between legitimate user and sybil). Wei et al. [82] have presented *SybilDefender* mechanism to detect the sybil users in large-scale social networks based on the assumptions that limited number of edges exist between legitimate community and sybil community. In [112], *SybilShield* framework has been proposed to detect legitimate users through a modified random walk approach based on community detection algorithm. Gong et al. [110] have proposed *SybilBelief*, which is a semi-supervised learning approach for detecting sybil users based on *Markov* random fields and belief propagation. Further, this approach fails when the number of edges has increased between legitimate community and sybil community. Yang et al. [111] have proposed *VoteTrust* model for detecting sybils over social network graph. This model restricts the number of requests sent from sybil to legitimate users. However, if sybil can increase the number of attack edges, then the *VoteTrust* model requires high computation and achieves low accuracy. In [35], SybilSCAR method has been proposed to detect sybil users (in online social networks) by considering random walk and belief propagation with neighbor influence. The proposal is scalable, convergent and robust to identify noisy data. However, the existing methods cannot detect other type of sybils such as fake comments, fake likes, fake contents and fake reviews.

In [113], *SybilRank* tool has been developed in order to rank the users based on their probability of being identified as sybil users. This tool reduces the false positive rates by considering trustworthy users. Furthermore, its efficiency is reduced when the trustworthy user establishes the social relationship among sybil users. Boshmaf et al. [114] have designed *Integro* method, which is an improvement over *SybilRank* by considering the social behavioral aspects of each user in order to predict the probability of being identified as sybil. Furthermore, *Integro* is restricted to an undirected social network graph and it achieves low accuracy for detecting sybil among new users.

Mislove et al. [115] have designed *Ostra* method to reduce the unwanted social interactions among users. This model requires the user to classify the message as either relevant or irrelevant message by providing feedback, which is slightly burden to the user. *SocialFilter* has been derived by Sirivianos et al. [9] to improve sybil tolerance of spam mitigation.

However, these existing works are based on the assumption that the edge between legitimate community and sybil community is limited because legitimate users are more likely to be friend of known users.

Tran et al. [116] have designed a decentralized *GateKeeper* to provide sybil-resilient mechanism. The proposal uses ticket distribution process on each user (or node), where each weighted edge represents the number of tickets disseminated through that edge. *SybilInfer* has been presented by Danezis et al. [117], which is a centralized sybil detection approach based on *Bayesian* inference method. Moreover, *SybilInfer* can handle up to thousand of nodes, which is not comparable to the size of large-scale social networks. In [118], Mulamba et al. have proposed *SybilRadar* approach by integrating with the trust model to detect sybil users more accurately. In [119], *SybilFence* framework has been designed to detect sybil users based on the user feedback in online social networks. Moreover, this framework also restrict the social relationships among users who spreads the malicious information or negative feedback.

2.5.2 Spammer based Fake Content Detection

In literature, several spam detection approaches have been proposed in *Twitter* network to detect spam (or fake) content in *Twitter* network [75], [120], [121], [122]. Madisetty et al. [75] have developed ensemble-based convolutional neural networks model by considering user-level features, tweet content-level features and n-gram features to detection spam content (or fake content) in *Twitter* network. In [120], Sedhai et al. have designed a semi-supervised spam detection approach to detect spam tweets (i.e.,fake content) from three different perspectives, such as tweet with blacklisted URLs, tweets posted by untrustworthy user and predicting spam tweets based on multi-classifier model. In [121], the authors have detected spam tweets by considering different classification learning techniques (such as *Naive Bayes*, *Random forest* and *Decision tree*). Fazil et al. [122] have designed a hybrid approach to detect automated spammers in *Twitter* network by considering community features (such as reputation and clustering coefficient features) with user's metadata and social interaction features. Further, the authors have analyzed by applying two-tailed

Z-test to distinguish social behavior of spammers from legitimate users.

In [22], Chen et al. have proposed a learning method from unlabelled tweets (*Lfun*) framework by integrating with *Random forest* classifier method to identify changed spam content from unlabelled tweets. However, the efficiency of the proposal reduces when too old spam tweets are considered in order to detect the unlabelled tweets. Shen et al. [123] have analyzed spammer behavior from the viewpoints of user closeness based on interactions, users' interests and trustworthiness of a user. Further, the authors have proposed a Bayesian spam filtering technique to differentiate spam emails from legitimate emails. In [47], a support vector machine based learning algorithm has been proposed for detecting spammers based on content-based features, profile-based features and user behavior-based features. Further, the authors have also analyzed the most influencing features for detecting the spammers in an online social network. Sometimes, conventional machine learning algorithms cannot capture the variation of spammer's behavior. Wu et al. [78] have applied a deep learning model by considering Word2Vec [124] embedding model to identify the variability of spammer posting spam content. However, the proposal relies only on text features and which may not be efficient to distinguish spam content from non-spam content in *Twitter* network.

Fakhraei et al. [125] have presented a *Markov* random field model by considering users' credibility score based on sequential n-gram features and graph based features in order to classify spammers in social networks. In [126], Wu et al. have detected spammers and spam content by considering posting relationship between users and messages, social interactions between (any) two users and social relationships between messages (in terms of replies, re-posts or comments) in microblogging. Shena et al. [126] have proposed multi-view learning model by integrating classification model with regularization for detecting spammers in *Twitter*. Further, the authors have applied non-negative matrix factorization approach to predict user influence based on the posting behavior between users and tweets on a specific topic. In [127], the authors have analyzed automated spam posting behavior of spammer based on distribution of URLs, co-occurring words and user mentions to differentiate spam tweets among legitimate tweets.

2.5.3 Malicious URLs Detection

In literature [128], [129], the characteristic features of URL are considered as most essential features to classify malicious URLs because URLs contain phishing links. Moreover, the malicious users can manipulate the trust value in order to propagate malicious URLs in social networks [62]. Therefore, the semantic features of URLs are important for detecting the malicious URLs. Especially in Facebook, a malicious user sends friend request to many unknown users and also posts malicious content or malicious URLs to steal user credentials [130]. Therefore, by considering both content and URLs can improve malicious user detection rate.

Chen et al. [18] have considered both content-based and URL features, such as domain rank, URL count, similarity of content among different users to detect malicious URLs using Bayesian classification in online social networks. Akiyama et al. [131] have designed a social honeypot-based monitoring system to identify malicious websites based on malicious behavior of URL redirections. In [128], the authors have identified the malicious URLs using different machine learning algorithms by considering semantic features of URL and network host information. Suleman et al. [132] have extracted hyperlink based features, such as suspicious words in URL, http status count and number of links to distinguish phishing hyperlinks using a genetic algorithm.

Niu et al. [62] have proposed enhanced opinion walk (EOW) algorithm to distinguish trustworthy websites from spam websites based on hyperlinks. *EOW* algorithm is evaluated by integrating with trust model to determine the trustworthiness of each website. Janabi et al. [129] have extracted *URL*-based features (such as *URL* length, *Http*-302 status code and disabling right click) to distinguish legitimate *URLs* from suspicious *URLs*. In [52], a *URL*-based approach is proposed to detect spam tweets in *Twitter* based on the tweet content and *URL* redirection chains. Moreover, as detectors, if dynamic crawlers are used then the malicious user may identify them based on their interactions, *IP* addresses or honey client detection approaches [133].

In [134], Cao et al. have mainly focused on forwarding-based features to detect malicious URLs in online social networks. Moreover, the malicious user need to widely propagate the malicious links because the malicious links may be identified by online social network administrator. Vu et al. [135] have presented a multi-layer anomaly detection method to detect malicious URLs. This method applies n-gram features from URL to extract lexical based features. In [136], a multi-level classification model is presented using convolutional gated-recurrent-unit (GRU) neural network to detect malicious URLs based on text features. This model considers URL as a string and applies character-level embedding to extract features.

2.5.4 Fraudulent Reviews Detection

Online reviews are posted by users (with social content information) to express their opinions about items (or products). For instance, reviews about services, books, news, movies, etc. are categorized as product reviews. Recently, most of the people rely on online product reviews for their selective decision making process. In addition, the online reviews will help the service providers to improve the quality of their products and services. Moreover, the negative online reviews can cause financial loss for a service provider. Especially, in e-commerce websites, any user can post comments about a product as online reviews. Thus, by taking this advantage, malicious user tries to post fraudulent reviews in order to mislead the opinion of user for selective decision making. In literature [137], [138], different approaches are used to detect spammers and fraudulent reviews. These approaches are broadly classified into three different types based on features namely, linguistic patterns in the reviews (which depends on uni-gram, bi-gram or n-gram features) [139], user behavioral patterns (mostly user metadata based features) [140] and user linguistic patterns (which describe users' feelings or opinions about a product or service) [141]. The existing studies [142], [137] have suggested that fraudulent reviews can be identified more accurately by considering review-linguistic based, user metadata-based and user-linguistic based features.

Shehnepoor et al. [142] have proposed a network-based spam detection framework by considering user-behavioral, review-behavioral, user-linguistic and review-linguistic based features to identify spammers and fraudulent reviews in online social networks. Rout et al. [137] have presented a deceptive review detection system with linguistic features, POS features and sentiment score using supervised and unsupervised techniques to identify (labeled and unlabeled) spam reviews in e-commerce websites. In [139], Fusilier et al. have designed a opinion-based spam detection model with character-level n-gram word embedding to distinguish spam reviews among legitimate reviews. The authors have analyzed that spam and legitimate reviews are similar and dissimilar in terms of content and opinion, respectively. This method cannot integrate both character-level and word-level n-gram features to detect spam reviews more accurately. A review spam detection method has been proposed by Ahsan et al. [143] using three classification techniques namely, support vector machine (SVM), multi-layer perceptron (MLP) and stochastic gradient descent (SGD) with term frequency-inverse document frequency (TF-IDF) features of review content.

Jiang et al. [138] have identified that certain service providers are select a few fraudsters to manipulate content and provide faulty decisions to other products and services. Further, the authors have analyzed that the service providers provide significant ranking to their own services for selective decision making. Thanikkal et al. [144] have proposed opinion spam recognition using ontology model to detect faulty reviews. The authors have classified the faulty reviews into three categories namely, non-review (i.e., a review which does not have any opinion), off-product review (i.e., a review which does not describe about the product) and fraudulent review (i.e., a review which is untrustworthy and mislead user with fake information). In [145], Shao et al. have proposed a hybrid spam detection method with deep sentiment analysis to distinguish spam content among genuine content. Further, the authors have identified a set of words which are mostly associated with fake content.

Most of the existing approaches [35], [111] have considered a random walk model (where each participant moves to one of its neighboring participants with equal probability) to detect sybil users (i.e., multiple fake identities), which are under the control of a single malicious user. Such models assumed a limited number of edges exists between the legitimate community and the sybil community because legitimate users are more likely to be a friend of known users. However, a malicious user can compromise a large number of legitimate accounts in order to establish a large number of attack-edges between the legitimate community and the sybil community. Hence, a malicious user can perform different type of malicious activities either by creating multiple fake identities or by compromising legitimate accounts in order to spread spam-content in online social networks.

2.6 Learning Algorithms and Malicious Social Bot Detection Approaches in OSNs

Learning algorithms can identify data patterns from huge volume of data shared in online social networks. Moreover, learning algorithms learn from past experiences and provide accurate results. Learning algorithms are broadly classified into three different types namely, supervised learning, unsupervised learning and reinforcement learning algorithms. Supervised learning algorithms are trained on a class (with a set of specific tasks) and predicts a class. Unsupervised learning helps to find patterns from dataset without pre-existing labeled data. Reinforcement learning algorithms are trained on a reward and predicts a learning action.

Madisetty et al. [75] have presented five different convolutional neural network models by considering tweet features. Gupta et al. [77] have designed a framework for detecting spammers in *Twitter* network using different supervised learning algorithms (such as neural network, gradient boosting, support vector machine and Random Forest). Cao et al. [146] have presented an autoencoder-based unsupervised learning algorithm by incorporating users' content information with network structure for community detection in online social network. Further, the authors have adopted modularity maximization model [147] and normalized cut [148] in order to partition (social) graph into different groups.

In [149], a reinforcement learning based trust propagation algorithm has been proposed to identify the trustworthy paths between the source and target participant. The authors have evaluated the trust value between two participants based on the identification of trustworthy path. In [150], a learning automata based particle swarm optimization-influence maximization algorithm is proposed to identify a set of users, who can maximize influence spread in an online social network. Jaradat et al. [151] have proposed a neural network-based reinforcement learning algorithm to minimize privacy propagation in online social networks. Further, the authors have presented a hybrid trust model by considering social semantic relationship between two participants (i.e., OSN accounts).

Malicious social bot detection approaches are broadly classified into different categories namely, machine learning-based and social graph-based approaches. The machine learning-based approaches consider large number of features with different classification techniques in order to distinguish legitimate users among social bots. The social graph-based approaches are based on social network graph with nodes as users and edges as social relationship between users. Two different types of social-graph based approaches have been proposed to detect malicious social bots. First type of approaches are based on the social trust relationship between users. Second type of approaches are based on centrality measures and graph topology. For social botnet detection, the existing works [51], [13] considered user based features, such as sentimental analysis and content based features. Several existing approaches have been proposed for detecting social bots in online social networks [75], [49], [8], [12]. The existing approaches have considered either tweet based features or graph based features for detecting social bots in online social networks.

2.6.1 Machine Learning based Social Bot Detection

The social bots in online social networks (like *Facebook* and *Twitter*) have gained more attention recently. Chu et al. [13] 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. Zhang et al. [81] 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. Davis et al. [152] have designed a system named as *BotorNot* by adopting 10-fold cross validation with random forest classification technique. The authors have identified three types of social bots namely, self-promoters, spammers and social bots who adopt applications for posting content in social media.

Freitas et al. [153] 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. Subrahmanian et al. [51] 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 the proposed program could generate social bots by varying number of parameters for social botnet creation. Ashfaq et al. [154] 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. In [155], a hashing method has been proposed to dynamically differentiate user accounts based on their posting behavior.

Kudugunta et al. [49] 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. Madisetty et al. [75] have developed five different convolutional neural network models by considering tweet features. Gupta et al. [77] have designed a framework for detecting spammers in *Twitter* network using different machine learning algorithms. Shi et al. [8]

have detected malicious social bots based on user behavioral transition probability features (such as sharing, liking and commenting). The authors have shown that malicious social bots can be accurately detected with user behavioral transition probability features when compared to quantitative features (like number of hashtags, number of replies and number of comments). Moreover, these studies consider user profile features which can easily be modified by malicious bots. Moreover, profile features and social interaction features may not detect malicious *URLs* which are posted by the participants.

2.6.2 Graph based Social Bot Detection

Yan [12] 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 as *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. Halfaker et al. [156] 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 [10], 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. Mehrotra et al. [157] have presented an approach to detect fake *Twitter* followers based on centrality measures. Soliman et al. [158] have designed a weighted *AdaGraph* model by integrating with unsupervised technique based on clustering coefficient to detect and predict accuracy of social bot detection. Alarifi et al. [159] have constructed ground truth for 2000 accounts based on 10 expert ratings to distinguish automated accounts (i.e., social bots) among non-automated accounts (i.e., legitimate users). The authors have evaluated their performance by comparing with the ground truth in other existing works.

Boshmaf et al. [60] 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. Ferrara et al. [55] 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. In [53], 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).

Besel et al. [160] have analyzed social botnet attack on *Twitter*. The authors have revealed that usually social bots use *URL* shortening services and *URL* redirection in order to redirect users to malicious web pages. Echeverria et al. [161] have detected, retrieved and analyzed star wars botnet over thousands of users to observe the social behavior of bots. In [162], a social bot hunter model has been presented based on the user behavioral features, such as follower ratio, number of *URLs* and reputation score. In [105], a trust model has been designed to detect malicious activities in an OSN. The author have analyzed that low trust value of user indicates that the information spreaded by the user is considered as untrustworthy. Moreover, in some social networks (like *Twitter*), establishing social interaction with strangers is one of the characteristics.

Learning from the data patterns using supervised learning may not provide accurate results in cases where existing data items are biased and bot behaviour dynamically changes over a period of time. Moreover, reinforcement learning algorithms provide improved learning by repeated interactions with the environment.

2.7 Spam Influential Users and Influential Community Detection Approaches in OSNs

Zhang et al. [34] 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. [163] 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. [32] have proposed *D*-hops model, which considers degree centrality with multi-hop distance measure for identifying top-k influential users in directed and undirected graph.

In [164], 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. [165] have presented topic behavior influence based tree method based on five features (such as message content, hashtags, replies, mentions and retweets) for identifying influential users in *Twitter* network. Singh et al. [150] have proposed a learning automata based particle swarm optimization-influence maximization (LAPSO-IM) algorithm to identify a set of influential users, who can maximize influence spread in an online social network.

Wang et al. [37] have detected bots by considering correlation graph and applied modularity based clustering approach for botnet community detection. In [36], the SpamCom method is proposed to detect spammers communities based on user behavioral features. The proposal method identifies spammers (or social spam bots) based on user behavioral features and applies clique to determine strongly connected botnet communities. Further, the authors have used normalized mutual information (*NMI*) to determine the correctness of detected communities with true communities. Zhuang et al. [166] have detected botnet communities depending on maximum clique detection method with network structure. Dang et al. [41] have presented a method to detect spammer groups in microblogging based on the characteristics of network topology and retweeting networks.

Choo et al. [40] have proposed a spammer group detection (*SGD*) algorithm to detect spammer communities based on content similarity, sentimental score and user interactions. Further, the authors have analyzed the correlations between content spamicity (i.e., the probability of message being spam) and number of reviews posted by spammers. Zhang et al. [167] have proposed a partially supervised learning (PSL) algorithm which uses frequent item set mining and positive unlabeled learning methods to detect spammer communities in online social networks. Wang et al. [168] have presented a graph-based group spam framework to detect spammer reviewer communities based on network-based features. Further, the authors have proposed spammer community-based measures (such as reviewer ratio, multiple reviews and neighbor tightness) to determine spamicity score for each community. Khanchi et al. [169] have presented a botnet detection method using genetic programming to detect bots and their malicious activities. Their proposal method is partitioned into two communities such that positive links are established within community and negative links are established across communities.

Most of the existing approaches [38], [39] focus on spreading trustworthy information in order to reduce the influence of spam content (or fake information) and detect spam initiators (i.e. social spam bots) in *OSNs*. However, the amount of influence of spam bots on legitimate participants (by frequent interactions) has not been adequately addressed.

In this thesis, Chapter 4 identifies the most influential users (which are influenced by the social bots) based on tweets and the users' interactions. In Chapter 5, spam-influential users are identified using the proposed spam influence minimization model and it helps in restricting the flow of illegitimate tweets in Twitter network. Further, in Chapter 5 social botnet community detection methods have been discussed in presence of different types of malicious activities. Further, behavioral similarity and trust values has been considered to detect social botnet communities more accurately.

2.8 Summary

In this chapter, different types of online social networks are discussed. A survey on data sharing security issues and attacks in online social networks has been presented. An exhaustive survey on learning algorithms and trust computational models have been discussed. Different types of malicious activity based detection approaches (such fake profiles, spreading spam-content, phishing attack and fraudulent reviews based detection approaches) have been presented. Malicious social bot detection approaches in online social networks are discussed. Further, spam influential users and spam influential community detection approaches have been presented. In this thesis, trust computational models and social bot detection approaches have been designed for online social networks. Table 2.1 shows the summary of literature on social bot detection approaches and and trust evaluation models in OSNs.

Table 2.1: Summary of literature on social bot detection approaches and trust evaluation models

Reference no.	Outcomes	Assumptions/Approach	Limitations
[22]	Bot detection through spam content	Proposed a model by considering RandomForest classifier method to detect spam content from unlabelled tweets	Older spam tweets are considered in order to detect the unlabelled tweets
[81]	Attacks and countermeasures for social bot detection	A bot retweets the spam tweets which are posted by botmaster and manipulates the influence value of each user	Network features are not taken into consideration

Continued on next page

Table 2.1 – continued from previous page

Reference no.	Outcomes	Assumptions/Approach	Limitations
[14]	Detection of Bot using deep neural network	Considers both content and metadata features to detect bots. These features are given as input to deep neural networks for processing the tweet content	Tweet content and metadata features can easily be modified by malicious bots. Further, this approach is unable to capture dynamic behaviors of bots
[13]	Detection of social bots using classification techniques	Classification system which includes spam based detection component and entropy-based component method	Temporal based features are not taken into consideration
[8]	Detection of Social Bots using Semi-supervised clustering	Semi-supervised clustering method is presented by considering quantitative and transition probability features	Feature ranking method is not addressed in order to identify important features
[35]	Sybil Bot detection	Based on random walk, trust value is assigned to each user. Each user updates its trust value based on its neighbor influence	This method is not more robust to handle noisy data
[111]	Detection of Sybil Bots based on vote trust model	Vote trust model uses PageRank algorithm. Legitimate user is more likely to be a friend of known user rather than strangers	Limited number of attack edges exists between social bot and legitimate communities

Continued on next page

Table 2.1 – continued from previous page

Reference no.	Outcomes	Assumptions/Approach	Limitations
[159]	Social Bot detection based on crowdsourcing method	Social bots are detected based on crowdsourcing method, where the experts are used for constructing ground truth	Human experts fail to detect fake accounts more accurately
[37]	Detection of social botnet communities	Detects botnet communities based on graph-based features and correlations of interactions among users	Unable to detect botnet communities with different types of malicious activities
[61]	Topic-based trust evaluation	If a tweet is trustworthy, then the user who posted it is likely to be trustworthy, and other tweets posted by this user are also likely to be trustworthy.	Malicious user may change its behavior (over time) and again may start posting malicious tweets
[34]	Identifies top-k sybil Influential users	Assumes that incoming retweets, mentions and replies are more trustworthy for measuring influence score rather than considering outgoing social interactions	Considered only limited number of edges exist between sybil botnet and legitimate communities.
[167]	Detection of spammers communities	Proposed *D*-hops model, which considers degree centrality with multi-hop distance measure for identifying top-k influential users	More relevant features are not addressed.

Continued on next page

Table 2.1 – continued from previous page

Reference no.	Outcomes	Assumptions/Approach	Limitations
[40]	Detection of spammers communities	Discovered positive and negative spammers communities through sentiment analysis and content similarity	Characteristics of network topology and retweeting networks are not addressed
[165]	Topic-based trust evaluation	Presented a topic based tree influence method based on five features (such as message content, hashtags, replies, mentions and retweets) for identifying influential users.	Temporal and graph-based features are not addressed.
[89]	Trustworthy propagation	Evaluates the trustworthy services and identifies a trustworthy path between a source and the target participants	Social relationships between participants have significant impact on trust evaluation which has not been addressed.

Several social bot detection approaches have been proposed in *Twitter* network to distinguish legitimate users and social bots [14], [81], [13]. Moreover, these studies consider user profile features which can easily be modified by malicious bots. The user profile features and social interaction features may not help in detecting malicious *URLs* which are posted by the users. Bots may use malicious *URL* redirection chains in order to avoid detection. Thus, bots can attack legitimate users by misleading detectors. In this thesis, Chapter 3 and Chapter 5 considers URL-based features to detect social bots who post malicious *URLs* in the tweets.

In Chapter 5, detection of malicious activities using deep autoencoder model has been presented. In this work, user behavioral similarity is analyzed from the viewpoints of tweet-content similarity, shared URL similarity, interest similarity, and social interaction similarity for identifying similar types of behavior (malicious or non-malicious) among participants in the Twitter network.

In this thesis, malicious social bot detection algorithms are designed to distinguish legitimate users among malicious social bots in Twitter network. In Chapter 3, malicious social bots are detected by considering learning automata model with URL-based features, such as URL redirection, frequency of shared URLs and spam content in URL in order to avoid phishing attack in Twitter network. In Chapter 4, a single-agent deep Q-network architecture has been designed by incorporating a deep Q-learning (DQL) model using social attributes (or features) in the Twitter network for detection of social bots based on updating Q-value function. Further, in Chapter 4, a multi-agent deep Q-learning model is presented to detect social spam bots in online social networks. In the next chapter, detection of malicious social bots using learning automata with URL-based features has been presented to distinguish legitimate participants among malicious social bots in the Twitter network.

Chapter 3

Detection of Malicious Social Bots using Learning Automata with URL Features in Twitter Network

Malicious social bot is a software program that pretends to be a real user in online social networks (OSNs) [8]. Moreover, malicious social bots perform several malicious attacks, such as spread social spam content, generate fake identities, manipulate online ratings and perform phishing attacks [8]. In *Twitter*, when a participant (user) wants to share a tweet containing *URL(s)*, with the neighboring participants (i.e., followers or followees), the participant adapts *URL* shortened service (i.e., bit.ly [16]) in order to reduce the length of *URL* (because for example a tweet is restricted upto 140 characters). Moreover, a malicious social bot may post shortened phishing *URLs* in the tweet [11]. As shown in Fig. 3.1, when a participant clicks on a shortened phishing *URL*, the participant's request will be redirected to intermediate *URLs* associated with malicious servers, which in turn redirect the user to malicious web pages. Then the legitimate participant is exposed to an attacker. This leads to *Twitter* network suffering from several vulnerabilities (like phishing attack).

Several approaches have been proposed to detect spam in Twitter network [75], [76], [77], [78]. These approaches are based on tweet-content features, social relationship features and user profile features. However, the malicious social bots can manipulate profile features, such as hashtag ratio, follower ratio, *URL* ratio and number of retweets. The ma-

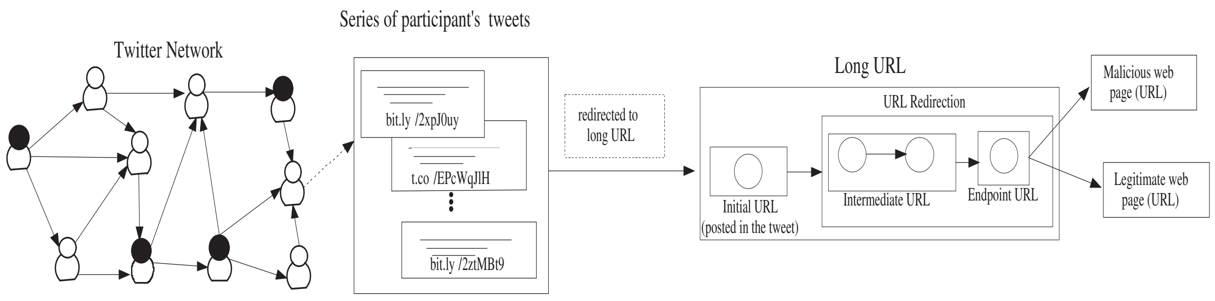


Figure 3.1: Malicious-act on URL shortened service

licious social bots can also manipulate tweet-content features, such as sentimental words, emoticons and most frequent words used in the tweets, by manipulating the content of each tweet [170]. The social relationship-based features are highly robust because the malicious social bots cannot easily manipulate the social interactions of users in *Twitter* network. However, extracting social relationship-based features consumes huge amount of time due to the massive volume of social network graph [12]. Therefore, identifying the malicious social bots from the legitimate participants is a challenging task in *Twitter* network. The existing malicious *URL* detection approaches [171], [128] are based on *DNS* information and lexical properties of *URLs*. The malicious social bots use *URL* redirections in order to avoid detection [172]. However, for detectors, identification of all malicious social bots is an issue because malicious social bots do not post malicious *URLs* directly in the tweets. Thus, it is important to identify malicious *URLs* (i.e., harmful *URLs*) posted by malicious social bots in *Twitter*.

Most of the existing approaches [13], [22] are based on supervised learning algorithms, where the model is trained with the labeled data in order to detect malicious bots in online social networks. However, these approaches rely on statistical features instead of analyzing social behavior of users [173]. Moreover, these approaches are not highly robust in detecting the temporal data patterns with noisy data (i.e., where the data is baised with untrustworthy or fake information) because the behavior of malicious bots changes over time in order to avoid detection [174], [175]. This motivated us to consider one of the reinforcement learning techniques (like learning automata model) to handle temporal data patterns. In this work, a learning automata model is designed to detect the malicious social bots with improved precision and recall.

In this chapter, the malicious behavior of participant is analyzed by considering features extracted from the posted *URLs* (in the tweets), such as *URL* redirection, frequency of shared *URLs* and spam content in *URL*, to distinguish between legitimate and malicious tweets. To protect against the malicious social bot attacks, a *Learning Automata based Malicious Social Bot Detection (LA-MSBD)* algorithm has been proposed and this integrates a trust computational model with a set of *URL*-based features for detection of malicious social bots. The proposed trust computational model contains two parameters namely, direct trust and indirect trust. The direct trust value is derived from *Bayesian* learning [18] (by considering *URL*-based features) to determine trustworthiness of tweets posted by each participant. In addition to direct trust, belief values (i.e., indicators for determining indirect trust) are collected from multiple neighbors of a participant. This is due to the fact that in case neighbors of a participant are trustworthy, the participant is likely to be trustworthy. Further, the *Dempster's* combination rule [19] aggregates the belief values provided by multiple 1-hop neighboring participants in order to evaluate indirect trust value of participants in *Twitter* network. Moreover, in this contribution, the belief values provided by multiple neighboring participants are considered to be independent. The proposed *LA-MSBD* algorithm helps to detect malicious social bots accurately (in terms of precision, recall, F-measure and accuracy) in *Twitter*. The major contributions of this chapter are as follows:

- Analyze the malicious behavior of a participant by considering *URL*-based features, such as *URL* redirection, relative position of *URL*, frequency of shared *URLs* and spam content in *URL*.
- Evaluate trustworthiness of tweets (posted by each participant) by using *Bayesian* learning and *Dempster-Shafer Theory (DST)*.
- Design of a *Learning Automata based Malicious Social Bot Detection (LA-MSBD)* algorithm by integrating a trust model with set of *URL*-based features.
- Performance evaluation of the proposed *LA-MSBD* algorithm using two *Twitter* datasets, namely *The Fake Project* dataset [21] and *Social Honeypot* dataset [20] in terms of

precision, recall, F-measure and accuracy for malicious social bot detection in *Twitter* network.

The remaining portion of this chapter is organized as follows: Section 3.1 presents the problem formulation. Section 3.2 provides a *Learning Automata based Malicious Social Bot Detection* algorithm. Section 3.3 presents the experimental results. Finally, the work is summarised in Section 3.4.

3.1 Problem Formulation

In this section, some basic terminologies are defined followed by problem formulation. The notations as used in this chapter are listed in Table 3.1.

Given a *Twitter* network $G = (P, E)$, where *P* represents a participant set $P = \{p_1, \ldots, p_n\}$ and *E* (i.e., $E \subseteq P \times P$) represents a social relationship set (or directed edges) between the participants (users). If there exists a social relationship between two participants, then they are considered as neighbors (i.e., either followers or followees). According to a given *Twitter* network with n participants and series of m tweets $tw_{p_i} = \{tw_{i1}, tw_{i2}, ...tw_{im}\}$ posted by each participant p_i, a feature set $F = \{f_1, f_2, ..., f_n\}$ can be constructed from each tweet posted by each participant. In this work, the features are assumed to be independent to each other. Based on the *URL*-based features (such as *URL* redirection, frequency of shared initial *URLs* and spam content in *URL*), trust parameters are defined in order to evaluate trustworthiness of all tweets posted by each participant.

The aim is to design a framework by considering feature set to evaluate the trust value of each *OSN* account (i.e., participant) and to detect malicious social bots in *Twitter* network effectively and efficiently. Further, two trust components namely, direct and indirect trust are defined to determine trust value of each participant.

Definition 1 (Direct Trust): Direct trust is defined as belief value of all tweets posted by each participant and denoted as $T^D_{p_i}(t)$. Let $T^D_{p_i}(t) \in [0, 1]$ represents the direct trust value of participant p_i and $T_{tw_{ij}}(t) \in [0, 1]$ represents trustworthiness of j^{th} tweet posted by i^{th} participant at time t. If $T^D_{p_i}(t) = 0$, it implies that all the tweets posted by participant p_i

contain completely fake or malicious information. If $T_{p_i}^{D}(t) = 1$, it implies that participant p_i consistently posts trustworthy information in the tweets.

Definition 2 (Indirect Trust): Indirect trust is a belief value of tweets posted by all one-hop neighboring participants of participant p_i at time t (denoted as $T_{p_i}^{ID}(t)$). If neighbors of a participant are trustworthy, then the participant is more likely to be trustworthy. The $T_{p_i}^{ID}(t) \in [0, 1]$ represents the indirect trust value of participant p_i. If $T_{p_i}^{ID}(t) = 0$, it implies that all the tweets posted by all the neighbors of participant p_i contain completely fake or malicious information. If $T_{p_i}^{ID}(t) = 1$, it implies that all the tweets posted by all the neighbors of participant p_i contain completely trustworthy information.

Table 3.1: Notations

Symbol	Description
$T_{p_i}(t)$	the trust value of participant p_i at time t
$T_{p_i}^{D}(t)$	the direct trust of participant p_i at time t
$T_{p_i}^{ID}(t)$	the indirect trust of participant p_i at time t
$tw_{p_i}(t)$	$tw_{p_i}(t) = \{tw_{i1}(t), tw_{i2}(t), ...tw_{im}(t)\}$, the series of *m* tweets for each participant p_i at time t
$tw_{ij}(t)$	the participant p_i posting j^{th} tweet at time t
$T_{tw_{ij}}(t)$	the trustworthiness of participant p_i posting j^{th} tweet at time t
$T^{D}(t)$	$T^{D}(t) = \{T_{tw_{i1}}(t), ..., T_{tw_{im}}(t)\}$, the set of trustworthiness of all tweets posted by i^{th} participant at time t
$T^{ID}(t)$	$T^{ID}(t) = \{T_{p_1}^{D}(t), ..., T_{p_L}^{D}(t)\}$, the set of direct trust values of all one-hop neighboring participants of p_i at time t
F	$F = \{f_1, f_2, ..., f_n\}$, the feature set
$\vec{F}$	$\vec{F} = \{\vec{f_1}, \vec{f_2}, ..., \vec{f_n}\}$, the feature ranking vector
$b_{p_1}(A)$	the belief value of participant p_1 with assumption A
β	$\beta = \{\beta_1, \beta_2, ...\beta_n\}$ represent set of reinforcement signal, where $\beta_i \in \{0, 1\}$

Problem (Malicious social bot detection): Given a *Twitter* network $G(t) = (P(t), E(t))$ with series of m tweets $tw_{p_i}(t) = \{tw_{i1}(t), tw_{i2}(t), ...tw_{im}(t)\}$ posted by each participant

p_i at different times $t \in 1, 2, ...\tau$, where $P(t)$ is a set of participants and $E(t)$ is a set of social relationships (directed edges) at time t. Let $T_P(t)$ represents the set of trust values (by considering both direct trust and indirect trust) of all the participants for all the posted tweets by the participants at time t. The goal of trust evaluation is to determine the trustworthiness of each tweet posted by participant and to identify malicious social bots in *Twitter*. The objective is to compute two functions:

$$f : \{G(1), G(2), \ldots, G(\tau)\} \to \{T_P(1), T_P(2), \ldots, T_P(\tau)\}$$
$$and$$
$$g : \{T_P(1), \ldots, T_P(\tau)\} \to C = \{\text{Legitimate, Malicious bot}\}$$

to determine the set of trust values of all the participants for all the posted tweets (by the participants) at time t (i.e., denoted as $T_P(t)$) and determine the class C of a participant p_i (as either legitimate or malicious social bot).

3.2 Detection of malicious social bots using Learning Automata along with URL-based features

In this section, firstly, a framework is proposed for analyzing the tweets posted by participants in *Twitter* network. In addition, a trust model is presented with several features that are extracted from *URLs* (which are posted by the participants in the tweets) for evaluating trust value of each participant in *Twitter*. Finally, a *Learning Automata based Malicious Social Bot Detection* (LA-MSBD) algorithm is proposed to identify the malicious social bots.

3.2.1 Proposed Framework for Detecting Malicious Social Bots

As shown in Fig. 3.2, the proposed framework consists of three components namely data collection, feature extraction and learning automata model. To collect tweets posted by participants (users), the tweets can be crawled using *Twitter Streaming APIs* [176]. The data collection component (i.e., phase) consists of three sub-components (i.e., sub-phases)

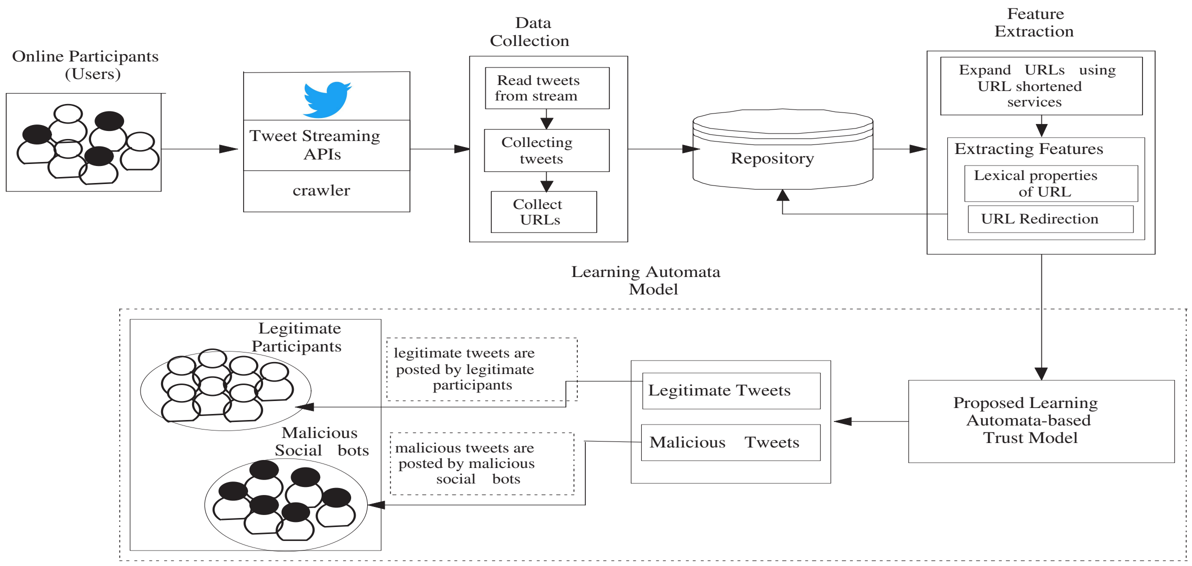


Figure 3.2: Proposed framework for detecting malicious social bots

namely reading tweets from *Twitter* streaming, collecting tweets and *URLs*. Moreover, the collected tweets and collected *URLs* are stored in a repository. The feature extraction consists of two sub-components namely, expanding shortened *URLs* and extracting feature set. Whenever feature extraction component obtains a shortened *URL* from the repository, it is converted into a long *URL* using *URL* shortened services (such as t.co, bit.ly and tinyurl.com) [177]. For each *URL* (posted by the participant in the tweet), several features are extracted that are based on the lexical properties of *URLs* (such as spam content, presence of -, @ and # symbols in domain name) along with the features of *URL* redirection (such as *URL* redirection length and relative position of initial *URL*). Further, these features are given as input to the proposed learning automata model for malicious social bot detection. The proposed learning automata model is integrated with a trust evaluation model. Moreover, the trust model determines the probability of a tweet containing any malicious information (such as *URL* redirection, frequency of *URLs* and spam content in *URL*). Finally, after evaluating the malicious behavior of a series of tweets posted by a participant, tweets are classified as malicious and legitimate tweets. However, malicious tweets are likely to be posted by malicious social bots. This helps in distinguishing malicious social bots from benign participants.

3.2.2 Feature Extraction

The accuracy of malicious social bot detection approach is based on several features which are extracted from *Twitter* network. In the proposed approach, *URL*-based feature set $F = \{f_1, f_2, ...f_{11}\}$ has been considered and the features are described in Table 3.2 (similar features are mentioned in [128], [52], [23]). These features are derived from *URL* redirection chains and the lexical properties of *URLs*. Moreover, the *URL*-based features are used in evaluating the trustworthiness of tweets posted by the participants. For example, malicious social bots usually have long *URL* redirects to avoid detection [23]. The malicious *URLs* are not usually placed at the end of *URL* redirection chain because social bots have to redirect *Twitter* users to different web pages in order to perform phishing attack [52]. Therefore, in this work, *URL* redirection length, relative position of initial *URL* and suspicious words in a *URL* are considered in order to detect malicious tweets.

Algorithm 3.1 Feature Ranking()

```
1: for each feature $f_i \in F$ do
2:     Compute weight $wt_{f_i} = G(f_i)/\sum_{i=1}^{n} G(f_i)$
3: end for
4: $\vec{F} \rightarrow$ Construct a feature ranking vector with weights $wt_{f_i}$ associated with each important feature $f_i$
```

Feature ranking algorithm is presented in Algorithm 3.1. Feature ranking algorithm (with a weight function) helps to identify the most important features based on the the weights associated with each feature. The weight function for i^{th} feature is defined as $wt_{f_i} = G(f_i)/\sum_{i=1}^{n} G(f_i)$, where n is the number of features, $G(f_i)$ represents the information gain value (i.e., the amount of information which is gained for a feature) of each feature f_i and it is computed based on *Shannon's* entropy model (Line 1-3). Using the weight function (Line 2, Algorithm 3.1), a set of features will be identified as important features and other set of features will be identified as less influential features based on available *Twitter* network dataset. For example (i.e., for some dataset), spam content, *URL* redirection length and relative position of initial *URLs* may be the most important features, whereas *URL* without host name and presence of symbols (like @, - and #) may be the least influential features for identifying malicious information in tweets. However, the actual set

Table 3.2: URL-based features

Category	Description and functionalities of each feature
Spam content	Tweets (or *URLs*) which are posted by legitimate participants are considered as legitimate tweets (or *URLs*). Initially, malicious social bots may act like legitimate participants. However, after attaining the (friend) acceptance request from other legitimate users, malicious bot post spammy words in the tweets (or *URLs*).
Presence of -, @ and # symbols in domain name	Malicious social bots usually add -, @ and # symbols in the domain name. Malicious social bots want *Twitter* users to feel that they are using a legitimate URL.
URL redirection length	*URL* redirection length represents the number of URL redirections which are performed by user until he/she reaches to landing web page.
Frequency of *URLs*	Most frequently appearing *URLs* (which are blacklisted) are usually considered as malicious. Moreover, this feature is computed as $\frac{n}{s}$, where $'n'$ is the number of times the initial *URL* appearing in the tweet and $'s'$ be the size of tweet. Malicious social bots usually have higher $\frac{n}{s}$ value than that of legitimate participants.
Relative position of initial *URL*	Malicious *URLs* are generally placed at the beginning of *URL* redirection chains. Malicious social bots redirect *Twitter* users to different web pages. Moreover, this feature is computed as $\frac{r_p}{l}$, where r_p is the relative position of an *URL* and l is length of *URL* redirection.
HTTP-302 status code	HTTP-302 is a status code which is returned by a server when a user makes *HTTP* request for an online web page. Moreover, *HTTP*-302 represents a *URL* redirection. Malicious social bots usually use *HTTP URL* redirection status code for redirecting the users to malicious website.
Number of different endpoint *URLs*	Number of different endpoint *URLs* is considered as one of the features because landing to multiple endpoint *URLs* is one of the malicious activities which is usually performed by malicious social bots.
PageRank	PageRank is a measure which determines the importance of web page used in the Internet and its value lies within a range of 0 and 1. Moreover, PageRank is based on the weight assigned to each linked pages and the number of incoming links.
Domain expiration	Most of the malicious websites usually expire within a short time duration. Moreover, most of the trustworthy websites regularly pay for many years in advance.
Abnormal *URL*	Abnormal *URL* represents the *URL* without any hostname. Moreover, this feature is extracted from *WHOIS* database [178].
Number of different domain names	Domain name or *IP* address is the most important part of *URL*. Moreover, usually the malicious *URL* redirection happens with different domain *IP* addresses.

of important features will be determined based on higher weight values on a given dataset. Finally, feature ranking vector $\vec{F} = \{\vec{f_1}, \vec{f_2}, ..., \vec{f_n}\}$ is constructed based on weights associated with each important feature (i.e., where important features will be chosen with higher weight values and the actual number of important features in $\vec{F}$ will be less than or equal to the number of features in initial feature set F) (Line 4, Algorithm 3.1).

3.2.3 Trust Computational Model

In *Twitter* network, there is an uncertainty in evaluating the trust value of participants because social trust relationship changes over time. To address this uncertainty, two methods namely, *Bayesian* learning and *Dempster-Shafer Theory* (*DST*) are considered. The proposed trust computational model contains two parameters - direct trust and indirect trust, where the former is derived using *Bayesian* learning and the latter is derived based on *Dempster-Shafer Theory (DST)*.

Let $G = (P, E)$ be a *Twitter* network, where $P = \{p_1, p_2, ..., p_n\}$ represents a set of participants (users) and $E \subseteq P \times P$ is the set of directed edges representing social relationships among participants. The trust value of participant p_i at time t (denoted as $T_{p_i}(t)$) is obtained using

$$T_{p_i}(t) = \alpha T_{p_i}^{D}(t) + (1 - \alpha) T_{p_i}^{ID}(t) \tag{3.1}$$

where $T_{p_i}^{D}(t)$ and $T_{p_i}^{ID}(t)$ are the direct and indirect trust of participant p_i respectively, at time t. Further, $\alpha \in [0, 1]$ represents weight assigned to $T_{p_i}^{D}$ and it is computed by adopting *Shannon's* entropy based trust model.

3.2.3.1 Direct Trust Computation

For malicious social bot detection, the direct trust value is evaluated based on identifying the malicious behavior of a participant in terms of posting malicious *URLs* in the tweets. It is assumed that participant p_i posts malicious information in the j^{th} tweet $tw_{ij}(t)$ at time t. The distrust value of participant p_i posting j^{th} tweet at time t (denoted as $DT_{tw_{ij}}(t)$) is given by

$$DT_{tw_{ij}}(t) = pr(C = malicious|tw_{ij}(t)) \tag{3.2}$$

Two classes namely, malicious and legitimate are considered to train a classifier in order to identify the malicious tweets. Here, a *Bayesian* classifier is applied in order to achieve better precision. In *Bayesian* classification, the class *C* to which a tweet belongs will be determined. The probability that tweet $tw_{ij}(t)$ belongs to class C (*i.e.*, malicious) is denoted as $pr(C = malicious|tw_{ij}(t))$. The feature set is considered with 11 features (for performance evaluation) which are extracted from each tweet (refer Section 3.2.2). Further, the feature ranking vector $\vec{F}$ is constructed with the weights associated with each important feature (as presented in Algorithm 3.1). Therefore, the tweet $tw_{ij}(t)$ is represented as a feature ranking vector $\vec{F} = \{\vec{f_1}, \vec{f_2}, ..., \vec{f_n}\}$. From *Bayesian* learning [179], the probability that a tweet (represented by feature ranking vector $\vec{F}$) is malicious will be determined as:

$$\begin{aligned}
pr(C = malicious|tw_{ij}(t)) &= pr(C = malicious|\vec{F}) \\
&= \frac{pr(C = malicious) \times pr(\vec{F}|C = malicious)}{pr(\vec{F})} \\
&= pr(C = malicious) \times pr(\vec{F}|C = malicious)\Big/ \\
&\Big(pr(C = malicious) \times pr(\vec{F}|C = malicious) + \\
&pr(C = legitimate) \times pr(\vec{F}|C = legitimate)\Big)
\end{aligned} \tag{3.3}$$

The weights are assigned to each feature before computing the trust value of a tweet, because features play a vital role for evaluating the trustworthiness of tweets posted by each participant. The features are ranked based on their associated weights (as discussed in Section 3.2.2). Ranking these features play a vital role to detect malicious social bots. Therefore, the probability of each feature is multiplied with its associated weight wt_{f_i} (refer Algorithm 3.1).

$$\begin{aligned}
pr(\vec{F}|C = malicious) &= pr(\vec{f_1}, \vec{f_2}, ..|C = malicious) \\
&= \prod_{i=1}^{n} pr(\vec{f_i} = x_i|C = malicious) \times wt_{f_i}
\end{aligned} \tag{3.4}$$

In Equation (3.4), each feature is conditionally independent. The probabilities $pr(\vec{f_i} = x_i|C = malicious)$ are estimated as the ratio of number of tuples containing a malicious class C (in training data tuples) having the value x_i for feature $\vec{f_i}$ (i.e., $\vec{f_i}$ is a ranked feature) and number of tuples containing a malicious class C (in training data tuples). Substituting Equation (3.4) in Equation (3.3), the probability $pr(C = malicious|tw_{ij}(t))$ (which is $DT_{tw_{ij}}(t)$ as given in Equation (3.2)) can be obtained. The trust value of j^{th} tweet posted by i^{th} participant at time t (denoted as $T_{tw_{ij}}(t)$) is determined by

$$T_{tw_{ij}}(t) = 1 - DT_{tw_{ij}}(t) \tag{3.5}$$

Therefore, the direct trust value of participant p_i (denoted as $T^D_{p_i}(t)$) at time t is computed as

$$T^D_{p_i}(t) = \frac{\sum_{j=1}^{m} T_{tw_{ij}}(t)}{m} \tag{3.6}$$

where $T_{tw_{ij}}(t)$ is the trustworthiness of j^{th} tweet posted by i^{th} participant at time t and m represents the number of tweets posted by p_i.

Algorithm 3.2 Direct Trust Computation()

```
 1: $T^D(t) = \phi$
 2: for each tweet $tw_{ij}$ j=1 to m do
 3:     if participant $p_i$ has posted $j^{th}$ tweet $tw$ to one of its neighboring participant (friend)
        then
 4:         $F \leftarrow$ Extract feature set of $tw_{ij}$
 5:         $\vec{F} \leftarrow$ Feature_ranking($F$)
 6:         for each feature $\vec{f_i} \in \vec{F}$ do
 7:             Compute $pr(\vec{f_1}, .., \vec{f_n}|C = malicious)$ using Equation (3.4)
 8:         end for
 9:         Compute $T_{tw_{ij}}(t)$ using Equation (3.5)
10:         Concatenation of $T^D(t)$ with value $T_{tw_{ij}}(t)$ and $T^D(t)$ is updated with the con-
            catenated values
11:     end if
12: end for
13: Compute direct trust $T^D_{p_i}(t)$ using Equation (3.6)
```

Direct trust computation algorithm (as discussed in Algorithm 3.2) takes a series of tweets posted by a participant p_i at time t as input. A set of features F are extracted from each j^{th} tweet tw_{ij} posted by a participant p_i at time t (Line 2-4). Based on the extracted features, a feature ranking vector $\vec{F}$ is constructed with the weights associated with the features (Line 5). The probability of each feature in $\vec{F}$ belonging to malicious class C is determined using Equation (3.4). Further, the distrust value of a tweet is computed by substituting Equation (3.4) in Equation (3.3), and finally from Equation (3.2) (Line 6-8). The trust value of each j^{th} tweet posted by a participant p_i (i.e., denoted as $T_{tw_{ij}}(t)$) is determined using Equation (3.5) (Line 9). Moreover, each trust value $T_{tw_{ij}}(t)$ is concatenated with a set $T^D(t)$ (where $T^D(t)$ is initially ϕ as shown in Line 1, Algorithm 3.2) in order to determine direct trust value of a participant p_i at time t (Line 10). Then $T^D(t)$ is updated with the concatenated values. The entire process is repeated for m number of tweets posted by each participant p_i. Finally, the direct trust value $T^D_{p_i}(t)$ of a participant p_i at time t is determined using Equation (3.6) by considering $\{T_{tw_{i1}}(t), ..., T_{tw_{im}}(t)\}$ (i.e. from the set $T^D(t)$) (Line 13, Algorithm 3.2).

3.2.3.2 Indirect Trust Computation

The indirect trust is determined by considering belief values of all 1-hop neighbors of a participant p_i. Although the direct trust value is important in evaluating the trustworthiness of participant, the belief values collected from multiple neighboring participants are also helpful in evaluating the trustworthiness of participant. Moreover, if legitimate participants randomly add malicious social bots as their friends, then the tweets posted by legitimate participants are likely to be considered as malicious because the legitimate participants are influenced by the malicious social bots [180]. Hence, the belief values collected from the multiple neighboring participants can reduce the bias in the trust value of a participant. The belief value of each one-hop neighboring participant is considered as conditionally independent. The direct trust value as computed using Equation (3.6) will be the belief value of a participant p_i at time t and it is used by *Dempster-Shafer* theory (DST) [19], where *Dempster's* weighted combination rules are applied to determine indirect trust value of a participant in *Twitter* network.

In this work, let X={malicious, legitimate} and A be an assumption that neighbors of a participant p_i are legitimate. Based on assumption A, the neighbors of p_i belongs to one

Algorithm 3.3 Indirect Trust Computation()

```
 1: $T^{ID}(t) = \phi$
 2: if participant $p_i$ has one or more one-hop neighboring participants then
 3:     for each $p_k \in NB(p_i)$ do; $NB(p_i)$-neighbors of $p_i$
 4:         $T^D_{p_k}(t) \leftarrow$ Direct_Trust_Computation($p_k$)
 5:         Concatenation of $T^{ID}(t)$ with value $T^D_{p_k}(t)$ and $T^{ID}(t)$ is updated with the concatenated values
 6:     end for
 7:     Compute indirect trust $T^{ID}_{p_i}(t)$ by using Equation (3.11)
 8: else
 9:     $T^{ID}_{p_i}(t)$ = 0
10: end if
```

of the states power set $2^X = \{\{\}, A$={legitimate}, $\bar{A}$={malicious} and $\mu = X\}$. Then the indirect trust value of participant p_i is computed from the belief value of multiple one-hop neighbors of p_i based on assumption A. If participant p_i has a neighbor p_1, then the belief value of p_1 with an assumption A (that participant p_1 is legitimate) is denoted as $b_{p_1}(A)$ and it is computed as the direct trust of p_1 using Equation (3.6) i.e., $b_{p_1}(A) = T^D_{p_1}(t)$. According to assumption A, the belief value $b_{p_1}(\bar{A}) = 0$. From *Dempster-Shafer Theory* [19], [181] $\sum_{m \in 2^X} b_{p_1}(m) = 1$, then the belief value $b_{p_1}(\mu) = 1 - T^D_{p_1}(t)$. If participant p_1 is legitimate, then participant p_i is likely to be legitimate. Therefore, the belief values are shown as:

$$\begin{aligned} b_{p_1}(A) &= T^D_{p_1}(t) \\ b_{p_1}(\bar{A}) &= 0 \\ b_{p_1}(\mu) &= 1 - T^D_{p_1}(t) \end{aligned} \tag{3.7}$$

Similarly, if participant p_i has a neighbor p_2, then the belief value of p_2 with an assumption A (that participant p_2 is legitimate) is denoted as $b_{p_2}(A)$. The belief values are computed as follows:

$$\begin{aligned} b_{p_2}(A) &= T^D_{p_2}(t) \\ b_{p_2}(\bar{A}) &= 0 \\ b_{p_2}(\mu) &= 1 - T^D_{p_2}(t) \end{aligned} \tag{3.8}$$

From the belief values of 1-hop neighboring participants of p_i, the participant p_i is identified as either legitimate or malicious social bot based on the trust value of neighbors of p_i (say p_1 and p_2). Therefore, the belief of p_1 and p_2 are combined by using *Dempster's* rule [19] that is given by:

$$\begin{aligned} b_{p_1}(A) \oplus b_{p_2}(A) &= \frac{1}{k}\Big[b_{p_1}(A)b_{p_2}(A) + b_{p_1}(A)b_{p_2}(\mu) \\ &\quad + b_{p_1}(\mu)b_{p_2}(A)\Big] \\ b_{p_1}(\bar{A}) \oplus b_{p_2}(\bar{A}) &= \frac{1}{k}\Big[b_{p_1}(\bar{A})b_{p_2}(\bar{A}) + b_{p_1}(\bar{A})b_{p_2}(\mu) \\ &\quad + b_{p_1}(\mu)b_{p_2}(\bar{A})\Big] \\ b_{p_1}(\mu) \oplus b_{p_2}(\mu) &= \frac{1}{k}\Big[b_{p_1}(\mu)b_{p_2}(\mu)\Big] \end{aligned} \tag{3.9}$$

$$\begin{aligned} \text{where, } k = \Big\{ & b_{p_1}(A)b_{p_2}(A) + b_{p_1}(A)b_{p_2}(\mu) + b_{p_1}(\mu)b_{p_2}(A) + \\ & b_{p_1}(\mu)b_{p_2}(\mu) + b_{p_1}(\bar{A})b_{p_2}(\bar{A}) + b_{p_1}(\bar{A})b_{p_2}(\mu) + b_{p_1}(\mu)b_{p_2}(\bar{A}) \Big\} \end{aligned} \tag{3.10}$$

Therefore, the indirect trust value of p_i at time t (denoted as $T^{ID}_{p_i}(t)$) is derived by combining the belief values [182] of all 1-hop neighbors (of p_i) and is determined as:

$$T^{ID}_{p_i}(t) = b_{p_1}(A) \oplus b_{p_2}(A) \oplus ...... \oplus b_{p_L}(A) \tag{3.11}$$

where, participant $p_1, p_2, \dots, p_L$ represents neighbors of p_i and the belief aggregation operator (i.e., $\oplus$) is commutative and associative [181].

Indirect trust computation algorithm is discussed in Algorithm 3.3. If participant p_i has one or more one-hop neighboring participants, then the indirect trust value of participant p_i at time t requires the belief values of its neighbors (say $p_1, p_2, ...p_L$). The belief value of each p_i neighbor is determined using the direct trust computation (as discussed in Algorithm 3.2) (Line 1-4). Moreover, each neighboring participant's direct trust value $T^D_{p_k}(t)$ is concatenated with a set $T^{ID}(t)$ (where $T^{ID}(t)$ is initially ϕ as shown in Line 1, Algorithm 3.3) in order to determine indirect trust value of a participant p_i at time t (Line 5). Then $T^{ID}(t)$ is updated with the concatenated values. The entire process is repeated for each neighboring participants (i.e., for L number of neighbors). Finally, the indirect trust value of p_i is computed by combining the belief values (i.e., direct trust values) of all 1-hop neighbors of p_i using Equation (3.11) by considering $\{T^D_{p_1}(t), ..., T^D_{p_L}(t)\}$ (i.e. from the set

$T^{ID}(t)$) (Line 7, Algorithm 3.3). Otherwise, the indirect trust value $T_{p_i}^{ID}(t)$ is set to 0 (Line 9).

For example, if participant p_i has two neighbors (say p_1 and p_2), then the result of combining two belief functions is computed as follows (using Equation 3.9): $b_{p_1}(A) = 0.5$, $b_{p_1}(\bar{A}) = 0$, $b_{p_1}(\mu) = 0.5$, $b_{p_2}(A) = 0.8$, $b_{p_2}(\bar{A}) = 0$, $b_{p_2}(\mu) = 0.2$, $b_{p_i}(A) = b_{p_1}(A) \oplus b_{p_2}(A) = 0.5 * 0.8 + 0.5 * 0.2 + 0.8 * 0.5 = 0.90$, $b_{p_i}(\bar{A}) = b_{p_1}(\bar{A}) \oplus b_{p_2}(\bar{A}) = 0 * 0 + 0.2 * 0 + 0.5 * 0 = 0$, $b_{p_i}(\mu) = 0.5 * 0.2 = 0.10$. The indirect trust value of p_i is 0.90.

3.2.4 Learning Automata based Malicious Social Bot Detection using Trust Model

In this section, firstly the motivation is discussed to applying learning automata model for malicious bot detection. Later, the *Learning Automata based Malicious Social Bot Detection* (LA-MSBD) algorithm has been proposed by considering direct and indirect trust components with various *URL*-based features in a *Twitter* network.

3.2.4.1 Motivation Behind Trust Computation and Learning Automata Model

Malicious social bots usually send malicious or fake content to their legitimate neighboring participants. Thereby, malicious bots can reduce the trust value of their legitimate neighbors. This motivated us to consider two trust parameters, namely direct trust (i.e., from users' own behavioral patterns while interacting in its neighborhood) and indirect trust (i.e., from belief values that are collected from the neighbors depending on their behavioral patterns) for malicious social bot detection. In *OSNs*, the behavior of malicious social bot changes with time. For constructing ground-truth, involving human experts may not always provide genuine interpretation by manually observing users' behavioral patterns [183]. Moreover, most of the existing supervised machine learning algorithms are not suitable when a malicious user manipulates data with noisy patterns (i.e., where data is baised with fake information) [175]. This motivated us to design a learning automata model (which is one of the reinforcement learning techniques) to detect the participants who are posting malicious information in the tweets. A participant cannot be considered as legitimate by posting a series of legitimate tweets at particular time slot *t*. This is due to the fact that the participant may change its behavior (with time) and start posting malicious tweets again, this in-turn misleads the detection of malicious bots. The motivation behind using learning automata is to detect a participant as a malicious social bot only after executing finite number of learning actions. In this work, finite number of learning actions represents a series of tweets posted by a participant at different time slots.

3.2.4.2 Learning Automata Model

In the proposed model, learning automata is defined as $< LA, A, pr, \beta, r_s, F >$, where $LA = \{la_1, la_2, ..., la_n\}$ and each learning automata la_i is associated with each participant (user) in *Twitter* network. Let $A = \{a_1, a_2, ...a_n\}$ represents set of actions, where each action a_i represents a series of tweets posted by participant p_i in *Twitter* network. Let $pr = \{pr_1, pr_2, \ldots.pr_n\}$ be the set of action probability values of participant p_i posting malicious information in the tweets at different time slots. Let $a_k(t)$ represents a learning action selected by a automata at time t. The learning actions are performed on an environment and produces either a penalty or a reward. In addition, $\beta = \{\beta_1, \beta_2, ..., \beta_n\}$ represents a set of reinforcement signals for each participant p_i at different time slots, where $\beta_i \in \{1, 0\}$. If $\beta_i = 1$, it implies that the proposed learning automata model identifies the malicious information from series of tweets posted by a participant at time slot *t*. The term r_s represents the learning algorithm in order to update reinforcement signal which is based on either penalty (ρ) or reward (r). For each tweet, success or failure of identifying a tweet as malicious will be predicted. Thus, $\beta_i = 0$ implies that the learning algorithm r_s fails to identify the malicious tweets. Further, $F : pr \times \beta \rightarrow pr$ represents the updation of action probability values with respect to the current action probability value and the response from the environment (i.e., β). If the learning automaton obtains a reward (i.e., when $\beta_i = 1$), then the probability value $pr(t+1)$ remains constant (unchanged). Otherwise, if the learning automaton obtains a penalty (i.e., when $\beta_i = 0$) then the probability value $pr(t+1)$ (at next time slot $t+1$) is updated as follows [184]:

$$pr(t+1) = \begin{cases} (1-\eta)pr(t), \beta_i = 0 \\ pr(t), \beta_i = 1 \end{cases} \quad \text{where } \eta \text{ is a constant.} \tag{3.12}$$

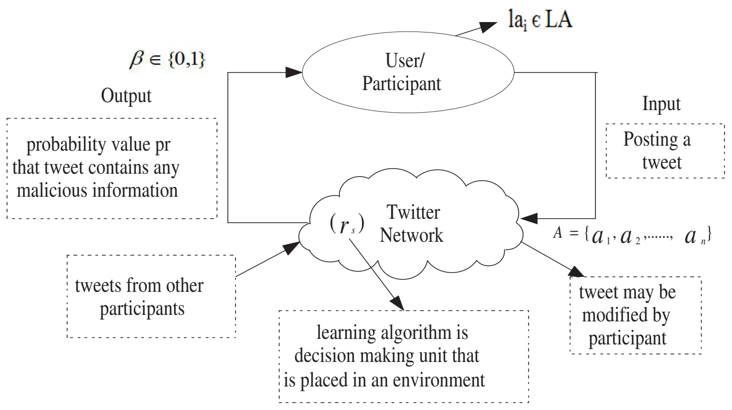

la$_i$ є LA
$\beta \in \{0,1\}$
Output
User/ Participant
Input
probability value pr that tweet contains any malicious information
Posting a tweet
(r_s)
Twitter Network
$A = \{a_1, a_2, \ldots, a_n\}$
tweets from other participants
learning algorithm is decision making unit that is placed in an environment
tweet may be modified by participant

Figure 3.3: Learning Automata model for social bot detection in *Twitter* network

In *Twitter* network, a tweet may contain malicious information which may lead to several vulnerabilities (like performing phishing attack). Malicious social bots are usually used for posting (or re-posting) spam (or malicious) tweets. In the proposed model, if a participant p_i posts a tweet to a participant p_j in *Twitter* network, then the learning algorithm r_s determines whether the tweet contains any malicious information, such as malicious *URL* redirection, frequency of shared *URLs* and spam content in *URL* (which are described in Section 3.2.2). In this work, reward r is defined as the probability of tweet which contains malicious information. The participant who posts malicious information in the tweet is rewarded for his/her postings and trust value of the participant will be reduced. Further, if a participant is continuously posting retweeted tweets with malicious information, then the participant will gain high reward for his/her malicious postings. Therefore, as the reward of the participant increases, then the distrust value also increases.

3.2.4.3 Proposed Learning Automata based Malicious Social Bot Detection Algorithm

A *Learning Automata based Malicious Social Bot Detection* algorithm (LA-MSBD) (refer Algorithm 3.4) has been presented by incorporating trust computational model in order to identify the malicious social bots. For each participant p_i, the learning automata is activated

Algorithm 3.4 Learning Automata based Malicious Social Bot Detection

```
Input:
    Set of participants $P = \{p_1, ..., p_n\}$ in Twitter, $\tau$: Number of time slots, $T_f$: Threshold value, $\epsilon$: Reward parameter
Output:
    $T$: a set of trust values of all legitimate participants with list of legitimate participants, $S_b$: a set of malicious social bots
Assumptions:
    Let $LA = \{la_1, la_2, ...la_n\}$ be set of learning automata, where $la_i$ represents learning automata for each participant.
begin
1:  $S_b = \phi$, $\beta = \phi$, $T = \phi$
2:  Learning automata is activated for each participant $p_i$
3:  for each participant $p_i \in P$ do
4:      for $t = 1, 2, ..., \tau$ do
5:          $T^{D}_{p_i}(t) \leftarrow$ Direct_Trust_Computation()
6:          $T^{ID}_{p_i}(t) \leftarrow$ Indirect_Trust_Computation()
7:          Compute trust value of $p_i$ ($T_{p_i}(t)$) using Equation (3.1)
8:          Compute action probability value $pr(t) = 1 - T_{p_i}(t)$
9:          if $T_{p_i}(t) < T_f$ then
10:             Concatenation of set $\beta$ with a string 1 and $\beta$ is updated with the concatenated values
11:         else
12:             Concatenation of set $\beta$ with a string 0 and $\beta$ is updated with the concatenated values
13:         end if
14:         Compute $pr(t + 1)$ using Equation (3.12)
15:     end for
16:     if (no. of 1's in $\beta >$ no. of 0's in $\beta$) then
17:         $S_b = S_b \cup \{p_i\}$ // $p_i$-malicious social bot
18:         reward $p_i$ using $T_{p_i}(t) = T_{p_i}(t) - \varepsilon$
19:     else
20:         $p_i$ is legitimate and added into the legitimate list of participants.
21:         Concatenation of set $T$ with the value $T_{p_i}(t)$ and $T$ is updated with the concatenated values
22:     end if
23:     $\beta = \phi$
24: end for
25: return $T$ with list of legitimate participants and $S_b$
```

and the trust value of p_i at time t (denoted as $T_{p_i}(t)$) is computed using Equation (3.1) (Line 1-7). Moreover, the Direct_Trust_Computation() (as mentioned in Line 5, Algorithm 3.4) will be determined using Algorithm 3.2 and Indirect_Trust_Computation() (as mentioned in Line 6, Algorithm 3.4) will be determined using Algorithm 3.3. Initially, the action probability value is computed as $1 - T_{p_i}(t)$ in order to identify the malicious information in the tweets posted by participant p_i at time slot t (Line 8). If $T_{p_i}(t) < T_f$, then p_i is rewarded by concatenating reinforcement signal set β with a string 1. This implies that the proposed *LA-MSBD* algorithm identifies a participant as a malicious social bot at time slot t. Otherwise, p_i is penalized by concatenating β with a string 0. Then β is updated with the concatenated values. The action probability value is updated for the next time slot using Equation (3.12). The entire process is repeated for finite number of learning actions at different time slots (Line 9-15). Therefore, reinforcement signal β is obtained as a string of 0's and 1's from different time slots t. Moreover, the proposed *LA-MSBD* algorithm identifies a participant as a malicious social bot if the number of 1's in β is greater than the number of zeroes in β. Moreover, once a participant is detected as a malicious social bot then the participant is rewarded by reducing the trust value of the participant with reward constant ϵ (where $\epsilon \in [0, 0.05]$). Otherwise, the participant is considered as a legitimate and added into the legitimate list of participants (Line 16-22). Moreover, the reinforcement signal set β is reset to null after identifying each participant as a legitimate or social bot (Line 23). Therefore, learning automata returns a set of trust values of all the legitimate participants with list of legitimate participants and set of malicious social bots in *Twitter* network (Line 25).

The malicious social bots (or spammer bots) can manipulate profile features (such as number of tweets, retweets and followers) by using pseudo-random generator functions [170]. Moreover, the malicious social bots can manipulate the content of each tweet. When malicious social bots perform phishing attack, the proposed *LA-MSBD* algorithm analyzes the malicious behavior of tweet (containing *URL*) by considering *URL*-based features (like *URL* redirection, frequency of shared *URLs* and spam content in *URL*) for bot detection. However, malicious social bots cannot manipulate *URL* redirection chains and spam content in *URL* because their intention is to perform phishing attack through *URL* redirection. Hence, the proposed method detects malicious social bots (by considering *URL*-based features) from phishing attack.

The time complexity of the proposed *Learning Automata based Malicious Social Bot*

Detection (LA-MSBD) Algorithm 3.4 is $O(n\tau km)$, where *n* is the number of participants (or users), τ is the number of time slots, *m* is the number of tweets posted by each participant at a particular time slot and k is the number of neighbors of a participant p_i. In the Algorithm 3.4, for each time slot *t*, the time complexity of direct trust computation Algorithm 3.2 is $O(\hat{F}m)$, where $\hat{F}$ is the number of features (in a feature ranking vector $\vec{F}$ as defined in Section 3.2.2). The time complexity of indirect trust computation Algorithm 3.3 is $O(\hat{F}km)$. Therefore, the time complexity for one time slot is $O(\hat{F}m + \hat{F}km)$ which can be rewritten as $O(km)$ by assuming $\hat{F}$ as a constant. For the number of time slots in Algorithm 3.4, the time complexity of each participant is $O(\tau km)$. Therefore, the time complexity of the proposed *LA-MSBD* algorithm is $O(n\tau km)$.

3.3 Performance Evaluation

In this section, the performance of the proposed *Learning Automata based Malicious Social Bot Detection* algorithm (*LA-MSBD*) is evaluated by considering two Twitter datasets namely *Social Honeypot*[1] dataset [20] and *The Fake Project*[2] dataset [21]. *The Fake Project* dataset and *Social Honeypot* dataset contain the labels for tweets (i.e., as legitimate and malicious tweets) and users (i.e., legitimate users and malicious bots). *The Fake Project* dataset contains 3,474 legitimate participants and 1000 malicious social bots (i.e., folder named as *traditional_spambots_1*). *Social Honeypot* dataset contains 19,276 legitimate participants and 22,223 social bots. For example, *Social Honeypot* dataset contains a set of labeled legitimate users and content polluter tweets (i.e., bots tweets). For *Social Honeypot* dataset, it has been considered that the tweets posted by legitimate users are (implicitly) labeled as legitimate tweets, and all tweets posted by content polluters are malicious tweets. Moreover, *Social Honeypot* dataset contains a text file as *'legitimate users tweets.txt'* with samples in the form of *'UserID, TweetID, Tweet, CreatedAt'*. Further, all the *URLs* in the tweets are collected in order to extract the *URL* based features. Using Algorithm 3.1, the feature ranking vector is constructed by choosing a set of features which have higher weight

[1]http://infolab.tamu.edu/data/
[2]http://mib.projects.iit.cnr.it/dataset.html

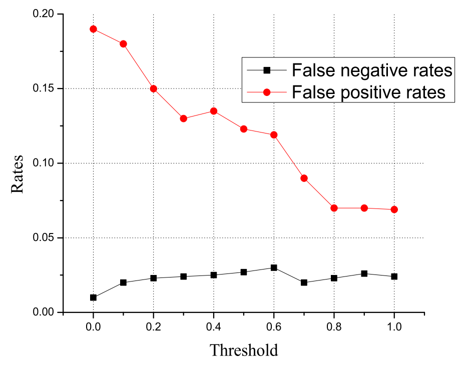
values for the above datasets. The details of *Social Honeypot* and *The Fake Project* datasets are presented in Table 3.3.

The proposed *LA-MSBD* algorithm identifies the participants as either legitimate participants or malicious social bots by considering *URL*-based features. Moreover, each *URL* feature is conditionally independent. If a tweet is determined to be malicious based on one *URL* feature and legitimate based on another *URL* feature, then the probability of the tweet being malicious is computed by using Equation (3.3). If a malicious tweet and a legitimate tweet both are posted by the same participant p_i, then the trustworthiness of each tweet will be determined by using Equation (3.5). Later, the trust value of participant p_i will be determined by using Equation (3.6). Moreover, if a participant posts a series of tweets with malicious information (or spam content in the tweet), then the participant is identified as malicious social bot. For example, if a participant p_i is marked as a malicious social bot in January 2019, then based on his/her malicious behavior in subsequent months, the proposed *LA-MSBD* algorithm identifies user as either a malicious social bot or legitimate.

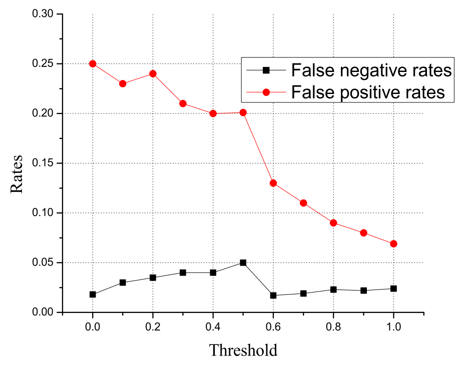
Table 3.3: Summary of Twitter Datasets

Dataset Name	Legitimate Users	Malicious Social Bots	Legitimate Tweets	Malicious Tweets
The Fake Project	3474	1000	8,377,522	145,094
Social Honeypot	19,276	22,223	3,259,693	2,353,473

Fig. 3.4 shows the variation of false negative and false positive rates for different threshold values. The false negative rates start decreasing when the threshold value is around 70 percent and 60 percent for *The Fake Project* dataset and *Social Honeypot* dataset, respectively. This implies that the proposed *LA-MSBD* algorithm can achieve high detection rate (i.e., in terms of recall) to detect malicious social bots for the above threshold values. Therefore, 70 percent as threshold value and 60 percent as threshold value have been considered for *The Fake Project* dataset and *Social Honeypot* dataset, respectively. However, the threshold value may vary depending on the nature of the dataset. The proposed *LA-MSBD* algorithm has been compared in two different ways: (i) *LA-MSBD* algorithm with four conventional machine learning algorithms and (ii) *LA-MSBD* algorithm with the exist-



(a) The Fake Project Dataset



(b) Social Honeypot Dataset

Figure 3.4: Variation of false negative and false positive rates for different threshold values.

ing social bot detection algorithms, such as learning from unlabeled tweets (*Lfun*) [22] and neural-network based redirection spam detection (*NN-RS*) [23].

3.3.1 Comparison with Conventional Machine Learning Algorithms

Firstly, the proposed *LA-MSBD* algorithm is compared with four conventional machine learning algorithms (such as *Support Vector Machine*, *Multilayer Perceptron*, *Logistic Regression* and *Random Forest*) by considering *URL*-based features with 5-fold cross-validation for malicious social bot detection. A library named as *scikit-learn* (with the respective packages) is used for the following four conventional machine learning algorithms.

Support Vector Machine (SVM): *SVM* is one of the popular supervised machine learning methods in order to classify the data. Moreover, *SVM* is used for both linear and non-linear separation of data by constructing a hyperplane in order to reduce the over-fitting of data [185]. For *SVM*, *LinearSVC* is used with *RBF* kernel (from sklearn package) by considering *URL*-based features.

Multilayer Perceptron (MLP): A *MLP* is a feed-forward neural network with at least three layers namely, input layer, hidden layer and output layer. *MLP* uses back propagation for training and network weights can be adjusted in order to minimize error between actual and predicted output [186]. For *MLP*, *MLPClassifier* is used with stochastic gradient descent (from sklearn.neural_network package) by considering *URL*-based features.

Logistic Regression (LR): *LR* uses logistic function to predict the outcome in terms of binary value (such as true/false, win/lose and yes/no) [187]. For *LR*, *LogisticRegression* is used (from sklearn.linear_model package of scikit-learn library) by considering *URL*-based features.

Random Forest (RF): RF is an ensemble learning algorithm, which implies that algorithm uses other machine learning algorithms in order to achieve better performance. Moreover, *RF* algorithm is used for constructing multiple decision trees based on random subsets of features [188]. For *RF*, *RandomForestClassifier* is used (from *sklearn.ensemble* package) by considering *URL*-based features.

The performance of the proposed *LA-MSBD* algorithm is evaluated in terms of F-measure, precision, recall and accuracy by taking set of *URL*-based features into consideration. The results are summarize for the proposed *LA-MSBD* algorithm by considering the following evaluation metrics: *True Positive (TP)*: Participants detected as malicious social bots are really malicious social bots. *False Positive (FP)*: Participants detected as malicious social bots are really legitimate participants. *False Negative (FN)*: Participants detected as legitimate participants are really malicious social bots. *True Negative (TN)*: Participants detected as legitimate participants are really legitimate participants.

The outcomes of the proposed *LA-MSBD* algorithm are compared with other existing algorithms based on the following metrics:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{3.13}$$

$$\text{F-measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.14}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.15}$$

Table 3.4 and Table 3.5 shows the comparison of the results of the proposed *Learning Automata based Malicious Social Bot Detection* (LA-MSBD) algorithm with four conven-

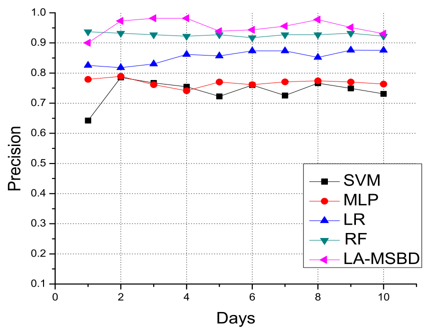
Table 3.4: Comparison performance of the proposed *LA-MSBD* algorithm with the existing algorithms for 5-folds cross validation of The Fake Project Dataset

Algorithm	one-fold	two-fold	three-fold	four-fold	five-fold	Average Precision (in %)
LA-MSBD	94.17	95.73	96.09	93.56	97.32	95.37
RF	91.74	92.71	93.20	92.23	93.68	92.71
LR	85.67	87.37	85.19	87.59	87.56	86.67
MLP	76.23	77.58	78.11	77.45	76.75	77.22
SVM	74.94	76.72	75.45	76.65	75.99	75.95

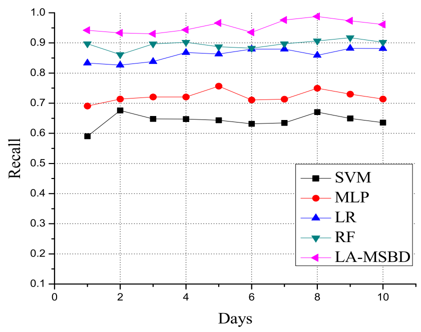
Table 3.5: Comparison performance of the proposed *LA-MSBD* algorithm with the existing algorithms for 5-folds cross validation of Social Honeypot Dataset

Algorithm	one-fold	two-fold	three-fold	four-fold	five-fold	Average Precision (in %)
LA-MSBD	90.0	92.85	91.81	90.91	93.32	91.77
RF	87.68	88.20	87.18	88.23	86.74	87.60
LR	84.18	84.20	83.18	84.23	83.74	83.90
MLP	75.52	76.79	75.01	73.16	75.68	75.23
SVM	72.92	71.12	70.12	72.01	72.35	71.70

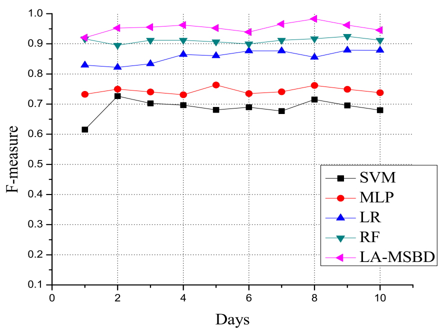
tional machine learning algorithms (such as *SVM, MLP, LR, RF*) for 5-fold cross validation on two datasets (such as *Social Honeypot* dataset and *The Fake Project* dataset) by considering *URL*-based features. For *SVM*, the malicious social bot detection performance in terms of precision is around 74%, which is a moderate value for binary data classification. The experimental results illustrate that the proposed *LA-MSBD* algorithm gives better performance when compared to conventional machine learning algorithms in terms of precision. The reason is that the data labeled by crowd-sourcing methods (i.e., which uses human intelligence or experts to label participant as either a bot or a legitimate) cannot efficiently provide genuine interpretation by manually observing behavioral patterns of a participant with its neighboring participants. Further, the trust value of a participant changes over time based on their behavioral patterns. Moreover, the proposed *LA-MSBD* algorithm is based on the malicious behavior of a participant (at different time intervals) and the belief values collected from the multiple one-hop neighboring participants (refer Section 3.2.3.2). Therefore, by incorporating direct and indirect trust parameters with learning automata model, the proposed *LA-MSBD* algorithm has achieved approximately 95% (for *The Fake Project* dataset) on precision with 5-fold cross validation. Therefore, the learning process conventional machine learning algorithms have achieved less precision when compared to the proposed *LA-MSBD* algorithm. Further, the precision value obtained for *The Fake Project* dataset is better than *Social Honeypot* dataset because *Social Honeypot* dataset contains noisy and untrustworthy information in its user content features than *The Fake Project* dataset.



(a) Precision



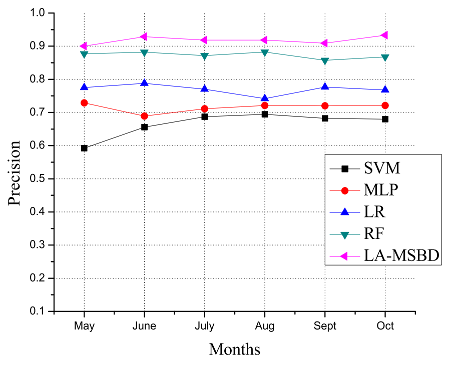
(b) Recall (True Positive Rate)



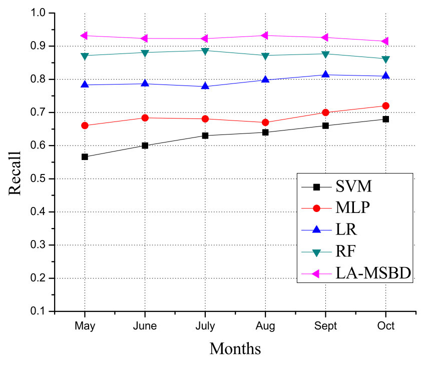
(c) F-measure

Figure 3.5: Malicious social bot detection on The Fake Project Dataset.

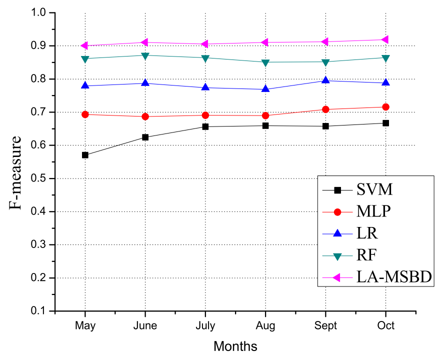
Fig.3.5 and Fig.3.6 show the performance of social bot detection algorithms in terms of precision, recall (true positive rate) and F-measure on different time slots. Fig.3.5(a) and Fig.3.6(a) illustrate that the *LA-MSBD* algorithm shows better performance in terms of precision as compared with the four conventional machine learning algorithms such as



(a) Precision

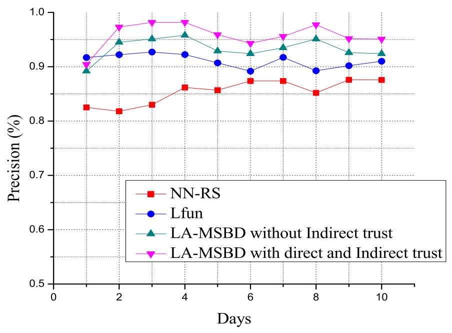


(b) Recall (True Positive Rate)

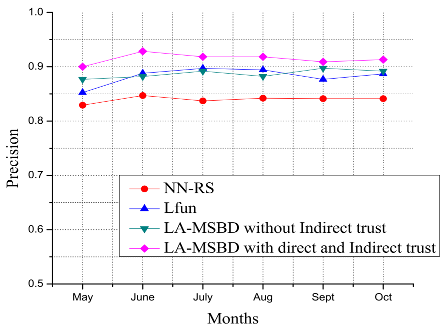


(c) F-measure

Figure 3.6: Malicious social bot detection on Social Honeypot Dataset.

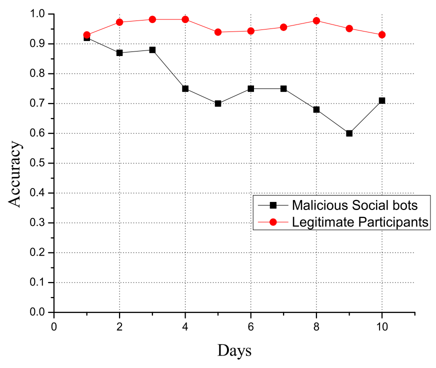


(a) The Fake Project Dataset

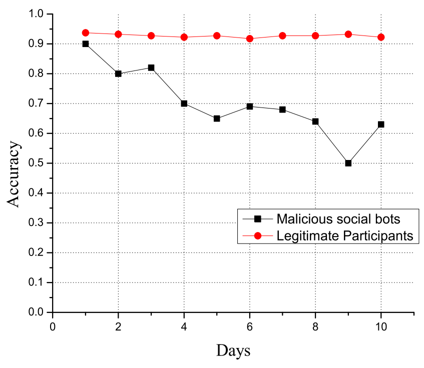


(b) Social Honeypot Dataset

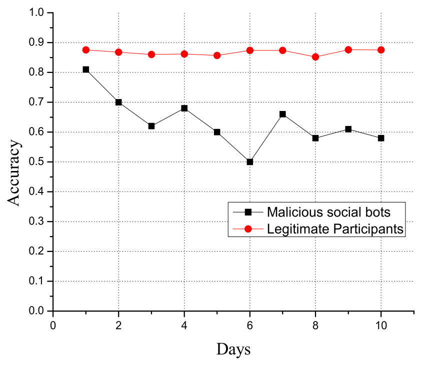
Figure 3.7: Comparison (performance in terms of precision) of the proposed *LA-MSBD* algorithm with the existing algorithms.



(a) LA-MSBD



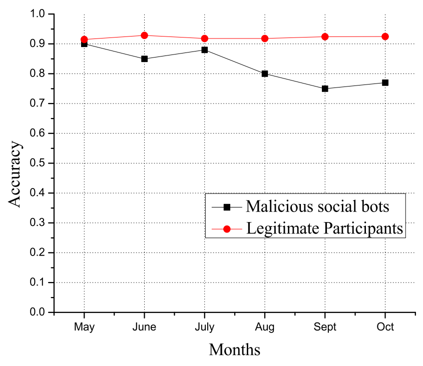
(b) Lfun



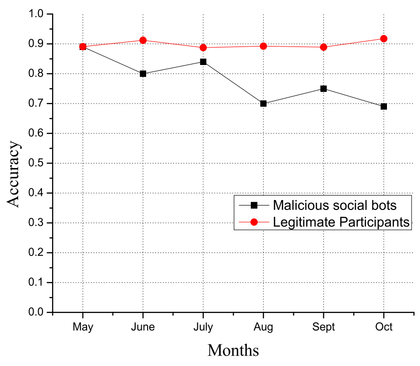
(c) NN-RS

Figure 3.8: Detection rates (in terms of accuracy) on The Fake Project Dataset.

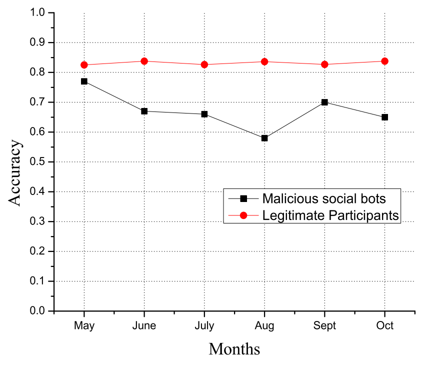
SVM, MLP, LR, RF. Moreover, in terms of recall (refer Fig. 3.5(b) and Fig. 3.6(b)) and F-measure (refer Fig. 3.5(c) and Fig. 3.6(c)), the proposed *LA-MSBD* algorithm outperforms the existing algorithms. For *The Fake Project* dataset, it has been observed that the precision level of proposed *LA-MSBD* algorithm is 10% higher than *Logistic Regression (LR)* when performing social bot detection for the 10^{th} day. From Fig.3.6(a), can be observed that the precision level of proposed *LA-MSBD* algorithm is around 10% higher than *Logistic Regression (LR)* while performing social bot detection on *Social Honeypot* dataset for the month of *October*. Fig. 3.5(c) and Fig.3.6(c) show that the F-measure of the proposed is above 90% for all days and months. The *Support Vector Machine (SVM)* has lowest F-measure of social bot detection that is below 75%. For *The Fake Project* dataset, the highest precision level of the proposed *LA-MSBD* algorithm is around 95% and the highest



(a) LA-MSBD



(b) Lfun



(c) NN-RS

Figure 3.9: Detection rates (in terms of accuracy) on Social Honeypot Dataset.

precision level of *Random Forest (RF)* is around 92%. For malicious social bot detection in *Social Honeypot* dataset, the highest precision level of the proposed *LA-MSBD* algorithm is around 90% and the highest precision level of *Random Forest (RF)* is around 87%. This is due to the fact that the proposed *LA-MSBD* algorithm executes for finite set of learning actions to update the action probability value and achieves the advantages of incremental learning. Hence, the learning automata model with trust component identifies the malicious tweets which are posted by malicious social bots.

3.3.2 Comparison with the Existing Social Bot Detection Algorithms

The proposed *LA-MSBD* algorithm provides better performance when compared with the existing algorithms, such as learning from unlabeled tweets (*Lfun*) [22] and neural-network based redirection spam detection (*NN-RS*) [23]. From Fig. 3.7, it can be observed that the proposed *LA-MSBD* algorithm with direct and indirect trust computation has highest precision among the three approaches. Further, it has been observed that the proposed *LA-MSBD* algorithm with direct and indirect trust achieves 2-3% improvement on precision value over the proposed *LA-MSBD* with only direct trust. The first reason is that the indirect trust of participant is derived from *Dempster-Shafer Theory (DST)* by considering the belief values of all 1-hop neighbors. The second reason is that malicious social bots may be trustworthy towards one participant and may be malicious towards other participants.

Table 3.6: Comparison of performance on two dataset in terms of Precision (P), Recall (R) and F-measure

Algorithm	The Fake Project			Social Honeypot			Third party based features	Search engine based features
	P	R	F-measure	P	R	F-measure		
NN-RS	87.75	85.38	86.54	84.53	82.34	83.42	no	no
Lfun	92.71	90.88	91.78	89.29	87.94	88.60	no	no
LA-MSBD	95.37	96.12	95.74	91.77	90.98	91.37	yes	yes

In the proposed *LA-MSBD*, search engine based features (like *PageRank* and domain expiration) are taken into consideration for detection of malicious social bots because most of the popular legitimate *URLs* may appear within the top-k search engine results. Moreover, comparison has been done in terms of recall, precision and F-measure to detect malicious social bots. Table 3.6 shows the comparison of the proposed algorithm with other existing approaches. The existing *Lfun* and *NN-RS* approaches obtain lower precision and recall when compared to the proposed *LA-MSBD* algorithm. For *The Fake Project* dataset, the recall of the proposed *LA-MSBD* algorithm is 96.12 percent and the recall of the existing *Lfun* is 90.88 percent. In the proposed *LA-MSBD* algorithm, third party based features (such as *DNS* information and *WHOIS* database[3]) are taken into consideration for detection of malicious social bots. Thus, the recall of the proposed algorithm is higher than the existing algorithms. For *The Fake Project* and *Social Honeypot* datasets, it can observed that

[3]http://whois.domaintools.com

the precision of the proposed *LA-MSBD* algorithm provides 3 percent improvement over the existing *Lfun*.

Fig. 3.8 and Fig. 3.9 show the accuracy of legitimate participant and malicious social bot detection for three different approaches namely, *LA-MSBD*, *Lfun* and *NN-RS*. From Fig. 3.8(a), Fig. 3.8(b) and Fig. 3.9(a), it can be observed that accuracy of legitimate participant detection is stable and it is more than 90% for the proposed *LA-MSBD* and the existing *Lfun* approaches. From Fig. 3.8(c), it can be observed that accuracy of legitimate participant is more than 85% for the existing *NN-RS* approach. From Fig. 3.8, it is observed that the accuracy of malicious social bot detection fluctuates. This is due to the fact that bots post less number of malicious tweets (at particular time slot t) in order to avoid detection. The malicious social bot detection (i.e., in terms of accuracy) for the proposed *LA-MSBD* algorithm is around 92% on the first day and decreases to 70% on 5^{th} day (as shown in 3.8(a)). The malicious social bot detection for the existing *Lfun* approach has been decreased from 90% on first day to almost 65% on 5^{th} day (as shown in Fig. 3.8(b)). Moreover, the malicious social bot detection for the existing *NN-RS* approach has been decreased from 81% on first day to almost 60% on 5^{th} day (as shown in Fig. 3.8(c)). Similarly, the results for the different approaches can be compared accordingly for different days (as shown in Fig. 3.8). From Fig. 3.9, it can be observed that the accuracy of malicious social bot detection fluctuates. This is due to the fact that the malicious social bots have reduced the length of *URL* redirection chains because long *URL* redirection chains are usually considered as suspicious. The malicious social bot detection for both the proposed *LA-MSBD* and the existing *Lfun* algorithms are around 90% on *May* month and the accuracy reduces to 75% on *September* (as shown in Fig. 3.9(a)). However, in other months, the accuracy of the proposed *LA-MSBD* algorithm is better than the existing *Lfun* approach. Moreover, the social bot detection for the existing *NN-RS* approach decreases from 85% on first month to almost 60% on other months (as shown in Fig. 3.9(c)). On an average, the proposed *LA-MSBD* algorithm provides better accuracy over the existing algorithms namely, *Lfun* and *NN-RS*.

3.4 Summary

In this chapter, the proposed trust computation model contains two parameters, namely, direct trust and indirect trust. Moreover, the direct trust is derived from Bayesian learning, and the indirect trust is derived from the Dempster–Shafer theory (DST) to determine the trustworthiness of each participant accurately. A *Learning Automata based Malicious Social Bot Detection (LA-MSBD)* algorithm has been designed by integrating a trust model with a set of *URL*-based features in order to distinguish malicious social bots from legitimate participants. Moreover, the proposed LA-MSBD algorithm detects a participant as a malicious social bot only after executing a finite number of learning actions. In the next chapter, single-agent and multi-agent deep reinforcement learning (i.e., deep Q-learning) models have been presented.

Chapter 4

Deep Reinforcement Learning Models for Detecting Social Bots and Influential Users

Most of the existing approaches are designed on learning algorithms which rely on statistical features in order to detect social bots in *OSNs* [14], [8], [189]. In [24], [80], malicious users have strong motivation to manipulate the data (which may be used by supervised machine learning algorithm) in order to avoid detection. Thus, this may lead to misclassification for new sample of data during testing phase. However, the recent studies have illustrated that supervised machine learning algorithms fail to detect social bots in certain situations, such as when training data is more biased [25]. Thus, a reinforcement learning algorithm is applied in order to detect social bots more accurately.

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 the 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 [190]. 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 [191], [192], [193] 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 [29]. To overcome this problem, single-agent and multi-agent deep *Q*-learning models are considered by using *Q*-value function (i.e., state-action value function) through a deep *Q*-network. 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 [194].

In the proposed deep Q-learning models, different types of social attributes (such as, tweet-based attributes, user profile-based attributes, social graph based attributes and temporal attributes (or features)) are given as input to the deep *Q*-network. For each user, the social attributes are represented in the form of state vector S, which contains a set of states (i.e., social attributes). For each state-action pair, the system (i.e., deep *Q*-learning model) determines next state and reward function (i.e., social behavior of user). After finding the state-action pair, the agent decides whether the corresponding user is acting as a malicious social bot or a legitimate user. The major contributions of this chapter are as follows:

- A single agent deep *Q*-network based architecture is proposed 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).

- A multi-agent deep Q-learning model based on particle swarm optimization (PSO) method is also proposed for detecting social bots more accurately.

- A top-k influential (user) 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 two datasets collected from the *Twitter* network, such as *The Fake Project* dataset [21] and *Social Honeypot* dataset [20].

The rest of this work is organized as follows: Section 4.1 presents the problem formulation. In Section 4.2, a single agent *Deep Q-Learning* algorithm has been proposed for detecting the social bots. Section 4.3 presents a multi-agent deep Q-Learning model using particle swarm optimization for detecting social bots. Section 4.4 identifies the most influential users in an online social network. The experimental results are discussed in Section 4.5. Finally, the summary of this work is presented in Section 4.6.

4.1 Problem Formulation

Given a *Twitter* network $G = (P, E)$ where $P = \{p_1, ..., p_n\}$ is the set of users (or participants) and $E = \{e_1, ..., e_n\}$ is the set of social relationships (or directed edges) between users. For, each participant p_i, different attributes (or features) (such as, tweet-based attributes, user profile-based attributes, social graph based attributes and temporal attributes) are represented as a state vector S_i. In deep reinforcement learning, the agent transition movements from current state to next state is termed as a learning action set A (which is discussed in Section 4.3.2). Based on state and action pairs, positions $Q(s, a)$ and velocities $V(s, a)$ are modeled in order to determine an optimal action. The objective is to design single and multi-agent deep Q-learning models by considering state vector with optimal action sequences in order to detect social bots more accurately in *Twitter* network. Further, the goal is to identify the most influential users (which are influenced by the social bots) in Twitter network.

Problem Statement: Given a participant $p_i \in P$, the features (of p_i) are represented by state vector S_i with optimal action. The single-agent and multi-agent deep reinforcement learning algorithms determine the group ($\hat{G}$) that the participant p_i belongs to. Further, to minimize influence of spreading spam content, top-k influential users are identified in Twitter network. The objective is to determine two functions f and g namely,

$$f : P =< S_i >\rightarrow \hat{G} = \{\text{Social bots, Legitimate participants}\}$$

and

$$g : \hat{G} = \{\text{Social bots, Legitimate participants}\} \rightarrow \{\text{top-k influential users}\}$$

4.2 A Single Agent Deep Q-Learning Model for Detecting Social Bots

In this section, a set of social attributes and an attribute ranking algorithm are presented to analyze the behavior of social bots. In addition, a deep Q-network architecture has been designed, which incorporates the proposed *Deep Q-Learning* algorithm and social attributes from the *Twitter* network for social bot detection.

4.2.1 Deep Q-Network Architecture

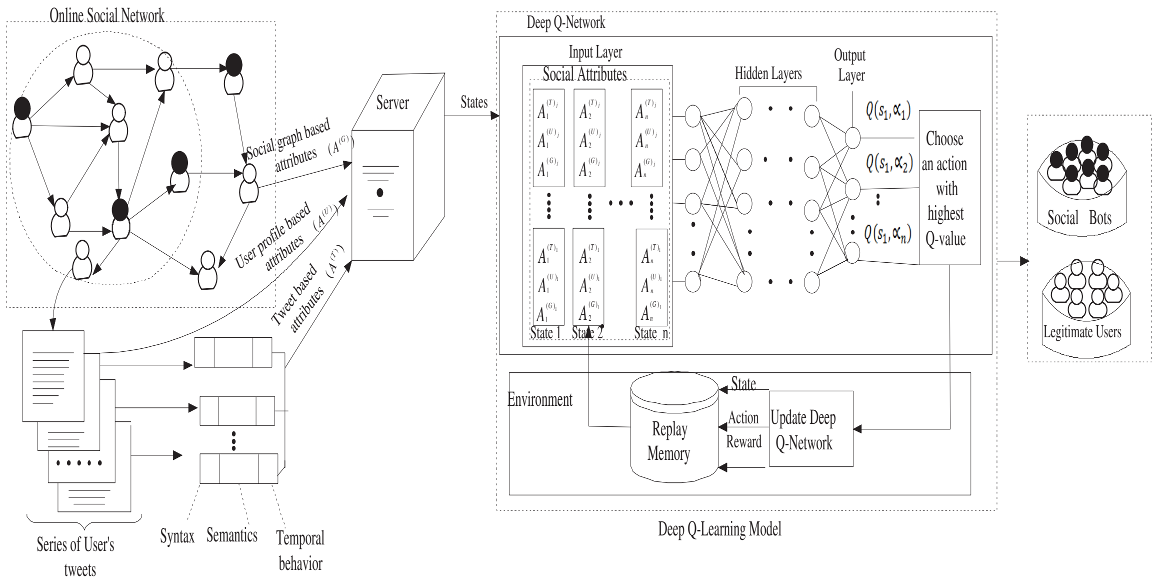


Figure 4.1: Deep Q-learning architecture for social botnet detection

Fig.4.1 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.4.1 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.4.1) also stores the collected data in the form of state vector S (which is discussed in Section 4.2.3), 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 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 4.2.3). 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.

4.2.2 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 user profile-based attributes and social graph based attributes.

4.2.2.1 Tweet-based Attributes

Tweet-based attributes are extracted 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.

A lexical normalization technique is used on *Twitter* data to obtain the individual words (or tokens) [195]. Further, the individual words are classified as either positive or negative

Table 4.1: Overview of Tweet-based Attributes

Category	**Length**	**Description**
Syntax (Sy)	1	Whether the tweet contain any emoticons (e.g., ☹)
	5	Total number of URLs, replies, retweets 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 tweeted about the user
	1	Computing the user's sentimental score
Temporal behavior (Tm)	1	Timestamp of each user' tweet

Table 4.2: Overview of User profile-based Attributes

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-$\frac{\lvert tw_{URL}\rvert}{\lvert tw\rvert+\lvert tw_{URL}\rvert}$ (where $\lvert tw\rvert$ is the total number of tweets posted by user's and $\lvert tw_{URL}\rvert$ is the total number of user's tweets containing URLs)
	1	Hashtag ratio-$\frac{\lvert tw_{\#}\rvert}{\lvert tw\rvert+\lvert tw_{\#}\rvert}$ (where $\lvert tw_{\#}\rvert$ is the total number of tweets posted by user's starting with $\#name$)
	1	Mention ratio-$\frac{\lvert tw_{@}\rvert}{\lvert tw\rvert+\lvert tw_{@}\rvert}$ (where $\lvert tw_{@}\rvert$ 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- $log\frac{\lvert followers\rvert+1}{\lvert friends\rvert+1}$
	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

emotions based on the user's tweets. Moreover, punctuation symbols ('#','?', '!', '....', '.'), special characters ('@', '\$', '%') and emoticons (i.e.,☹) in the tweet are to be extracted.

Geo-tagged user's tweets gives the information about location of the posted tweets [196]. *Latent dirichlet allocation* is used to identify the most frequent words or topics posted by the *Twitter* users [197]. A sentimental analysis framework and an opinion analysis system are adopted in order to compute user's sentimental score based on the tweet [198], [199].

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

$$A_{p_i}^{(T)} =<< Sy_i^j >, < Sm_i^j >, < Tm_i^j >> \tag{4.1}$$

4.2.2.2 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 work, user profile based attributes are extracted from a series of user's tweets with sampling time per-week basics. Moreover, the tweet size is limited up to 140 characters [81]. Hence, user profile-based attributes are defined 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

$$A_{p_i}^{(U)} =<< TB_i >, < UI_i >> \tag{4.2}$$

4.2.2.3 Social Graph-based Attributes for Tweet Propagation

The social graph-based attributes mainly focus on the social relationships among the users. Hence, social graph-based attributes, such as clustering coefficient, closeness centrality, betweenness centrality and pagerank centrality are to be defined for each user.

1. *Clustering coefficient:* A social graph $G = (P, E)$, where P represents a set of participants (users) and E represents a set of directed edges (or links). A directed edge e_{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.

2. *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 $C(p_i) = \frac{1}{\sum d(p_i,p_j):p_j \in P}$.

3. *Betweenness Centrality:* Betweenness centrality is a measure of 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

4. *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 [12]). 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

$$A_{p_i}^{(G)} =< CC(p_i), B_C(p_k), C(p_i), PR(p_i) > \tag{4.3}$$

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

$$A_{p_i} =<< A_{p_i}^{(T)} >, < A_{p_i}^{(U)} >, < A_{p_i}^{(G)} >> \tag{4.4}$$

Therefore, the social attributes determines the social bots among legitimate users.
As shown in Algorithm 4.1, for each participant (user) in an online social network, three

Algorithm 4.1 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:
 1: Initialize S={}
 2: for each participant $p_i \in P$ do
 3:     $< A^{(T)} >$= Extract_Tweet_Based_Features($p_i$)
 4:     $< A^{(U)} >$= Extract_User_Profile_Based_Features($p_i$)
 5:     $< A^{(G)} >$= Extract_Graph_Based_Features($p_i$)
 6:     $A_{p_i}$=$<< A^{(T)}_{p_i} >, < A^{(U)_{p_i}} >, < A^{(G)_{p_i}} >>$
 7:     $\bar{A}_{p_i}$=Normalization(A)
 8:     pv=PCA($\bar{A}_{p_i}$)
 9:     for each attribute $a_i \in \bar{A}_{p_i}$ do
10:         Compute geometric mean $wt_{a_i} = \frac{(\prod_{j=1}^{n} a_{ij})^{1/n}}{\sum_{i=1}^{n}(\prod_{j=1}^{n} a_{ij})^{1/n}}$
11:         S = $wt_{a_i}$
12:     end for
13: end for
14: S=Create a list of ranked states (social attributes) with respect to pv
15: return S
```

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 tweet. 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. A normalization technique (*z-score*) is used, 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 equal important for social bot detection. Further, principal component analysis (*PCA*) method is applied 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).

4.2.3 Proposed Single Agent Deep Q-Learning Model for Detecting Social Bots

An adaptive single agent deep *Q*-learning algorithm has been proposed by considering the following elements:

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

$$S_t =< \{s_t^{i1}, s_t^{i2}, ...s_t^{ij}\} > \tag{4.5}$$

  Here, each participant (user) $p_i \in P$ is associated with a set of social attributes A (refer Section 4.2.2 Equation 4.4). 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 4.2.2 Equation 4.4).

- *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:

$$r_t = \beta x_t^{ij} + c \tag{4.6}$$

  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 reaches 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 4.2 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.....$\tau$ do
 4:         Initializes state vector $S_t$ and begins with a state $s_t^{ij}$
 5:         Randomly choose a learning action $\alpha_t$
 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 $Q(s_t^{ij}, \alpha_t; w) = Q(s_t^{ij}, \alpha_t) +$
    $\epsilon\{r + \gamma Q(\overline{s}_t^{ij}, argmax_{\alpha'}[Q(\overline{s}_t^{ij}, all\_actions; w)])\}$
 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 $Loss(w) =$
    $E\left[(y - Q(s_t^{ij}, \alpha_t; w))^2\right]$
11:         Deep Q-network is updated with a learning rate parameter $\epsilon, w^- = \epsilon w + (1 -$
    $\epsilon)w^-$
12:     end for
13:     i++
14: end while
15: return Set of all social bots
```

The proposed *Deep Q-Learning* algorithm (refer Algorithm 4.2) 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 $Q(s_t^{ij}, \alpha_t; w)$, (ii) Compute the target *Q*-values y, (iii) Deep *Q*-network is updated by reducing the loss function $Loss(w)$ 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).

4.3 A Multi-Agent Deep Q-Learning Model using Particle Swarm Optimization for Detecting Social Bots

In conventional deep reinforcement learning (*DRL*) techniques (like deep *Q*-learning), an agent learns to reach the goal state through several episodes (i.e., path from initial state to goal state). Moreover, the learning process of *DRL* requires more computational resources compared to other machine learning algorithms [200], [26]. The deep *Q*-learning converges slower with high computation and storage space in order to determine and store the *Q*-values for all possible state-action pairs [26]. Therefore, finding an optimal sequence of actions in *Q*-learning (with faster convergence rate) is a major challenge.

Detection of web bots and traditional network bots have been addressed in [201], [202] using Particle swarm optimization (*PSO*). Moreover, *PSO* can be used to model each particle tagged with social behaviors in a social network. Further, in swarm intelligence, each particle's (a state) social behavior plays a vital role to improve the convergence performance of *PSO* [203]. In *PSO*, the temporal features (such as average number of tweets posted per day, longest user session time and percentage of dropped followers) are tuned to obtain optimal action. *PSO* has been applied to maximize social spam bot detection accuracy and to minimize learning action sequences in order to reach a goal state at faster rate with less number of iterations. The novelty of the proposed approach lies on the way the parameters are tuned using PSO for selecting an optimal action in an environment with multiple learning agents. This motivated us to integrate deep Q-learning with *PSO* in order to reduce high computation and also the learning agent stores only global best action sequences into replay memory instead of storing all possible state-action pairs. Moreover, based on situations, social spam bots may dynamically change their behavior to avoid bot detection. Thus in order to distinguish legitimate *OSN* accounts (i.e., participants) from social spam bots more accurately, users' temporal features (such as average number of tweets posted per day, longest user session time and percentage of dropped followers) are considered to analyze their behavioral patterns.

4.3.1 Particle Swarm Optimization based Deep Q-Learning Architecture

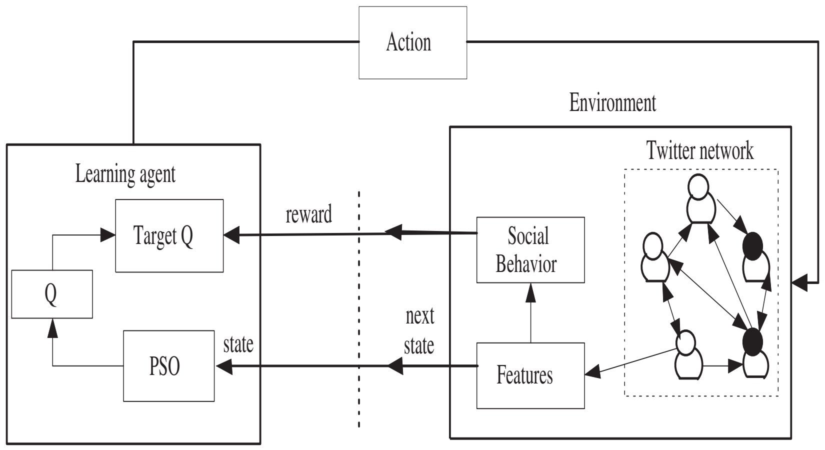


Figure 4.2: Architecture of P-DQL

Fig. 4.2 shows the architecture of the proposed particle swarm optimization based deep Q-learning (*P-DQL*). The environment represents a set of (malicious and non-malicious) users with series of tweets posted by each user (or participant). From the tweets posted by each user, the features (such as spam-content in the tweet, average number of tweets posted per day, longest user session time without any break and percentage of dropped followers) are extracted and represented as set of states. Initially states are given as input to the learning agent. The particle swarm optimization (*PSO*) component determines global best action sequences (as depicted in Fig. 4.3). The learning agent takes each state from global best action sequences and their respective belief-based reward value (which is discussed in Section 4.3.2) in order to compute Q-value and target Q-value. After executing a specific action in a state, the learning agent moves to the next state (i.e., which is available in global action sequences) and obtains belief-based reward value. For all possible global action sequences, if the learning agent cannot reach to a terminal state (i.e., identified as a spam bot based its social behavior) then the participant is identified as a legitimate. Otherwise, user is identified as a spam bot. Thus, the proposed *P-DQL* architecture is used to classify *Twitter* users as either spam bots or legitimate participants.

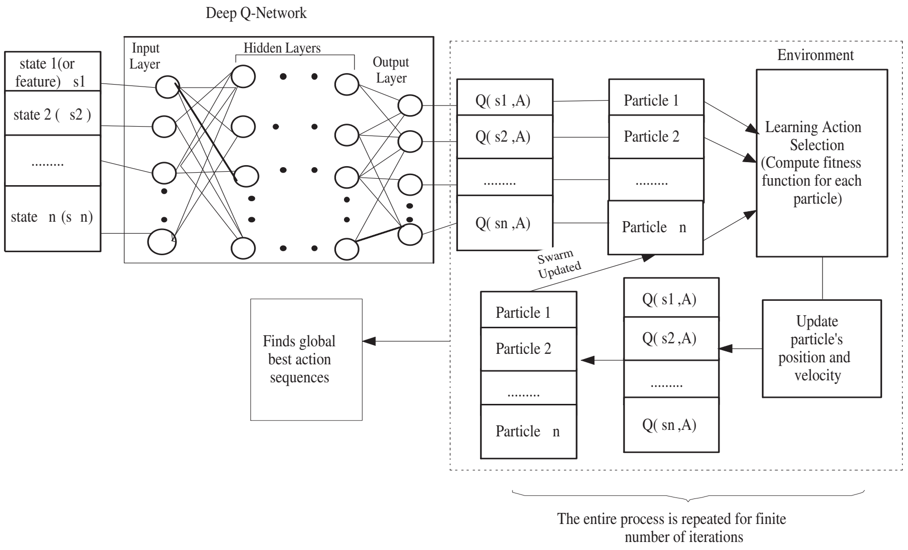


Figure 4.3: PSO framework with deep Q-network

Fig. 4.3 shows the proposed particle swarm optimization (PSO) framework with deep Q-network. The proposed *P-DQL* contains one of the elements as state vector (or a set of states) S_k which represents the k^{th} participant features that are extracted from *Twitter* network. Moreover, a state vector S_k is given as input to deep Q-network in order obtain Q-values $Q(s_i, A)$ with action sequences $A = \{a_1, a_2, ...\}$ for each state s_i (where $s_i \in S_K$). Each user with set of Q-values is termed as a swarm with set of particles (or a population) and each Q-value $Q(s_i, a_i)$ (i.e., where $a_i \in A$) represents a particle's position. For each particle (i.e., $Q(s_i, A)$), the fitness function (i.e., long-term immediate reward R^{long}) is to be computed (using Equation (4.17)) in order to determine local and global best particle's positions. Based on local and global best particle's positions, the position and velocity of particles are updated to determine global best action sequences (i.e., swarm updated). Moreover, the entire process will be executed for finite number of iterations until the variation of fitness values becomes negligible (i.e. fitness value unchanged in consecutive iterations).

Table 4.3: States

State	Description of a user behavioral patterns
s_0	Variation in posting positive (or negative) tweets over time
s_1	Longest session time without any break for at least 5-10 minutes
s_2	Percentage of dropped followers
s_3	Posting spam content in the tweet
s_4	Posting average number of tweets per day
s_5	Average time between two consecutive tweets
s_6	Temporal patterns of posting tweets (or retweets)
s_7	Inter arrival time between user's click events
s_8	Entropy of user's metadata features (such as posts, followers and reposts)

4.3.2 Updation Strategy of Q-value based on Particle Swarm Optimization

In optimization problem, population-based approaches are used to determine an optimal solutions [204]. Particle swarm optimization (*PSO*) utilizes features of swarm (or population) by considering local and global behavior of an agent [203]. For reinforcement learning, the optimal policies (i.e., the agent determines which action will be performed when agent is in a particular state) can be identified by considering updation strategies used in population-based approaches [205]. Especially in *Twitter* network, spam bots may influence other legitimate participants by tweets, number of followers and number of likes, comments or replies from participants. In this work, the behavior of social spam bot is analyzed with temporal features (such as average number of tweets posted per day, longest user session time without any break and percentage of dropped followers per week) and spam content in the tweet. Further, *PSO* is applied for an updation strategy of Q-value. A deep *Q*-learning method consists of four elements (namely, $(S, A, r(s, a), \gamma)$) are as follows:

- S is a set of states (or state vector) which represents the user's temporal features (or social attributes). The state set for k^{th} participant is denoted as S_k which is defined as

$$S_k = \{s_1, ..., s_l\} \tag{4.7}$$

For each participant, l states $(s_0, s_1, ...s_l)$ are considered (as listed in Table 4.3). The terminal state is defined as social behavior of feature (or state) with malicious intention (or activity). Moreover, if the agent can reach out a k^{th} terminal (or goal) state after performing a set of actions then the participant (or user) is identified as spam bot.

- A is a set of learning actions and defined as $A = \{a_1, a_2, ...\}$. The agent transition movements from current state to the next state is termed as a learning action. Moreover, at each current state, the learning agent determines the behavior of user in a current state s_i based on user's temporal features. Moreover, the learning agent (i.e., deep Q-network) learns to find an optimal action sequences which can reach terminal state with less number of iterations.

- $r(s_i, a_i)$ defines a current reward value $r(s_i, a_i)$ (where $s_i \in S_k$ and $a_i \in A$) which is determined based on social behavior of user (while performing an action a_i) in a state s_i. The current reward $r(s_i, a_i)$ is defined as

$$r(s_i, a_i) = \beta sb(s_i, a_i) + \upsilon \tag{4.8}$$

where $\beta \in [0, 1]$ represents learning parameter and the term $sb(s_i, a_i)$ represents social behavior of user in a state s_i (while performing an action a_i). If state s_i is terminal state then it gains reward and υ is set to 1. Otherwise, if state s_i is not terminal state then it gains penalty and υ is set to -1. In *Twitter* network, the uncertainty is involved in the probability of state being malicious to avoid detection where spam bots may randomly change their behavior over time. Therefore, a belief-based reward value $r(s_i, a_i, b)$ is defined in order to improve the trustworthiness of a state s_i and it is defined as

$$\begin{aligned} r(s_i, a_i, b) &= -H(b) + r(s_i, a_i) \\ &= -blogb + r(s_i, a_i) \end{aligned} \tag{4.9}$$

where b represents the belief value (or probability value) that state s_i is being malicious and the term $-H(b)$ represents negative entropy reward which maximizes future reward [206]. To determine the dynamic behavior of spam bots, belief value b is considered and it determines the probability that participant p_i changes its behavior (or state s_i) at time t. Based on the social behavior of each state, the probability that state s_i with belief value b (such that state contains malicious behavior) at time $t+1$ (i.e., which is denoted as $pr_{s_i}(X_{t+1} = b)$) is defined as follows [207], [208]:

$$pr_{s_i}(X_{t+1} = b) = \frac{1 + \sum_{s_j \in NB(s_i,b)_t} V_{ji} sb(s_j)}{1 + \sum_{s_k \in NB(s_i)_t} V_{ki} sb(s_k)} \tag{4.10}$$

where X_{t+1} represents the belief value b of state s_i at time $t{+}1$ and $NB(s_i)$ represents the neighboring states of s_i and $NB(s_i, b)_t$ represents the neighboring states of s_i having belief value b at time t. For each particle (or state), the learning agent (i.e., deep Q-network) considers the velocity V_{ji} i.e., rate at which agent moves from state s_j to the next state s_i (and whose value lies between 0 and 1). Further, the term $sb(s_j)$ represents the social behavior of state s_j. In this work, maximum likelihood estimation (*MLE*) method [209] is considered and it is used to determine unknown parameter from a given probabilistic model. The advantage of using *MLE* method provides faster convergence especially when the sample size increases [208]. In order to determine the immediate reward for each state-action pairs, the social behavior sb (i.e., unknown parameter) of each state being malicious has to be estimated. The maximum likelihood function $ML(E', sb)$ is defined as follows [208], [209]:

$$ML(E', sb) = log \prod_{(s_i,b)_t \in E'} pr_{s_i}(X_{t+1} = b) \tag{4.11}$$

where E' represents the experience tuple which consists of a series of four deep Q-network elements namely, state s_i, action a_i, reward $r(s_i, a_i)$ and next state s_j (i.e., $E' =< s_i, a_i, r(s_i, a_i), s_j, a_j, r(s_j, a_j), s_k ...... >$). Substituting Equation (4.10) in

Equation (4.11), which can be expressed as follows:

$$ML(E', sb) = \sum_{(s_i,b)_t \in E'} log\left(1 + \sum_{s_j \in NB(s_i,b)_t} V_{ji} sb(s_j)\right) * 1 - \sum_{(s_i,b)_t \in E'} log\left(1 + \sum_{s_k \in NB(s_i)_t} V_{ki} sb(s_k)\right) \tag{4.12}$$

The belief b that present state s_i is influenced by its previous neighboring state's s_j behavior (i.e., which is denoted as $b_{(s_j,s_i)_t}(sb)$ or $b_j(sb)$) and it is defined as follows [208]:

$$b_{(s_j,s_i)_t}(sb) = b_j(sb) = \frac{V_{ji} sb(s_j))}{1 + \sum_{s_k \in NB(s_i,b)} V_{ki} sb(s_k)} \tag{4.13}$$

For any $b \in [0,1]$ and $\sum_{s_j \in NB(s_i,b)} b_j(sb) = 1$, then

$$\sum_{s_j \in NB(s_i,b)} b_j(sb) + \frac{1}{1 + \sum_{s_k \in NB(s_i,b)} V_{ki} sb(s_k)} = 1 \tag{4.14}$$

Substituting *L.H.S* of Equation (4.14) in Equation (4.12) and $ML(E, sb)$ can be expressed as:

$$\begin{aligned} ML(E', sb) = & \sum_{(s_i,b)_t \in E'} \sum_{s_j \in NB(s_i,b)} b_j(sb) * log\left(1 + \sum_{s_j \in NB(s_i,b)_t} V_{ji} sb(s_j)\right) \\ & - \sum_{(s_i,b)_t \in E'} \sum_{s_j \in NB(s_i,b)} b_j(sb) log(V_{ji} sb(s_j)) + \sum_{(s_i,b)_t \in E'} \sum_{s_j \in NB(s_i,b)} b_j(sb) log(V_{ji} sb(s_j)) \\ & + \sum_{s_j \in NB(s_i,b)} \frac{1}{1 + \sum_{s_k \in NB(s_i,b)} V_{ki} sb(s_k)} log\left(1 + \sum_{s_j \in NB(s_i,b)_t} V_{ji} sb(s_j)\right) \\ & - \sum_{(s_i,b)_t \in E'} log\left(1 + \sum_{s_k \in NB(s_i)_t} V_{ki} sb(s_k)\right) \end{aligned} \tag{4.15}$$

From Equation (4.16), it can be obtained as follows:

$$ML(E', sb) = -\sum_{(s_i,b)_t \in E'} \sum_{s_j \in NB(s_i,b)} b_j(sb) * log\frac{V_{ji}sb(s_j)}{(1+\sum_{s_j \in NB(s_i,b)_t} V_{ji}sb(s_j))}$$
$$= -\sum_{(s_i,b)_t \in E'} \sum_{s_j \in NB(s_i,b)} b_j(sb) * log(b_j(sb)) \quad (4.16)$$

where $b_j(sb) * log(b_j(sb))$ represents entropy reward, which is used to determine belief-based reward value $r(s_i, a_i, b)$ (as defined in Equation (4.9)).

- γ: discount factor $\gamma \in [0, 1]$ determines importance of future rewards. When the discount factor γ reaches to one then the performance of learning process will improve and the number of learning steps will be significantly reduced [210].

The long-term immediate reward is the summation of all belief-based reward values for a sequence of actions. The goal of using particle swarm optimization based deep Q-learning (*P-DQL*) algorithm is to find the optimal action sequences which maximize the long-term immediate reward. Therefore, the long-term immediate reward R_i^{long} for i^{th} state (i.e., fitness function) is defined as

$$R_i^{\text{long}} = \sum_{l=1}^{L} \gamma^{L-l} r(s_i, a_l, b) \quad (4.17)$$

where L is the number of learning actions and $r(s_i, a_l, b)$ is the belief-based reward value (i.e., which is determined using Equation (4.9)).

For social spam bot detection, consider a set of Q-values (or state-action pairs) $Q^i = \{Q(s_i, a_{i1}), ..., Q(s_i, a_{iL})\}$ represents the position of i^{th} particle with L number of learning actions. Let $V^i = \{V(s_i, a_{i1}), ..., V(s_i, a_{iL})\}$ represents the velocity of i^{th} particle in a population, where $V(s_i, a_{ij})$ represents the state transition probability values while learning agent is moving from one state to other. Let p_b^i represents local best solution which is identified by i^{th} particle and g_b represents global best solution which is identified among all particles in a population (i.e., based on long-term immediate reward R^{long}). At each j^{th} iteration, the Q-values (i.e., position of particles) are updated as follows:

Algorithm 4.3 Particle Swarm Optimization (PSO)

```
Input:
    Initial values of all particles with Q(s,a)
Output:
    Optimal solution
1:  for j=1 to $itr_{max}$ do
2:      for i=1 to $\kappa$ do // $\kappa$ number of particles
3:          Compute the long-term immediate reward $R^{\text{long}}$ by using Eq. (4.17)
4:          // Update particle's local best position
5:          if $R^{\text{long}} < R_b^{\text{long}}$ or $R_b^{\text{long}} == -1$ then
6:              $p_b^i(j) = Q_j^i(s,a)$
7:              $R_b^{\text{long}} = R^{\text{long}}$
8:          end if
9:          // Update particle's global best position
10:         if $R^{\text{long}} < R_g^{\text{long}}$ or $R_g^{\text{long}} == -1$ then
11:             $g_b(j) = Q_j^i(s,a)$
12:             $R_g^{\text{long}} = R^{\text{long}}$
13:         end if
14:         Update particle's velocity using Equation (4.18)
15:         Update particle's position using Equation (4.19)
16:     end for
17: end for
18: return $g_b$=Optimal Solution
```

$$\begin{aligned} V_{j+1}^i(s,a) = wV_j^i(s,a) + c_1 ran_1(p_b^i(j) - Q_j^i(s,a)) \\ +c_2 ran_2(g_b(j) - Q_j^i(s,a)) \end{aligned} \tag{4.18}$$

$$Q_{j+1}^i(s,a) = Q_j^i(s,a) + V_{j+1}^i(s,a) \tag{4.19}$$

$$Q_j^i(s,a) = Q_j^i(s,a) + \epsilon \left\{ r(s_i,a_i,b) + \gamma Q^i \left(\hat{s}, \max_a[Q(\hat{s},a)] \right) - Q_j^i(s,a) \right\} \tag{4.20}$$

where $r(s_i, a_i, b)$ is the belief-based reward value obtained when learning agent moves from current state s to the next state $\hat{s}$ and ϵ is the learning rate (where $0 < \epsilon \leq 1$). Let $V_j^i(s,a)$ and $V_{j+1}^i(s,a)$ represent the current velocity of i^{th} particle at j^{th} iteration and new velocity of i^{th} particle at $(j+1)^{th}$ iteration, respectively. The term w represents weight

Algorithm 4.4 PSO based deep Q-learning (P-DQL) algorithm for social spam bot detection

```
    Input:
        Initialize deep Q-network Q(s,a), velocities $V(s,a)$, B=$\phi$, replay memory $r_m$ and
    $P = P'$, a set of participants $\{p_1, ....p_n\}$ (i.e., Twitter user accounts)
    Output:
        B, set of social spam bots and P, set of legitimate participants
 1: while $P \neq \phi$ do
 2:     $\beta = \{\}$ and $PL \leftarrow \{p_i\}$, where $p_i \in P$
 3:     for t = 1, 2.....$\tau$ do
 4:         $F_{p_i}(t) \leftarrow$ Extract temporal features and spam content in each tweet posted by
    $p_i$
 5:         $S_i(t)$= $F_{p_i}(t)$
 6:         for each state $s_t$ in $S_i(t)$ do
 7:             Initialize a state $s_t$ from state vector $S_i(t)$
 8:             $a_t$=Particle_Swarm_Optimization()
 9:             Execute action $a_t$ and obtain belief-based reward value $r(s_i, a_i, b)$ and next
    state $\hat{s}_t$
10:             Store the experience tuple $e_t =< s_t, a_t, r(s_t, a_t, b), \hat{s}_t >$ into $r_m$
11:             Get a sample $< s_t, a_t, r(s_t, a_t, b), \hat{s}_t >$ from $r_m$
12:             Compute target $Q$-value $y$ for each mini-batch transition
13:             if ($\hat{s}_t$ reaches $k^{th}$ goal state) then
14:                 $y = r(s_i, a_i, b)$
15:                 $\beta = \beta.append(1)$ // $\beta$ is appended with a string 1
16:                 break
17:             else
18:                 $y = r(s_i, a_i, b) + \gamma$ Q($\hat{s}$, arg max$_{\hat{a}} Q(\hat{s}, \hat{a})$)
19:             end if
20:             Update deep $Q$-network by minimizing Loss= $(y - Q(s,a))^2$
21:             $s_t = \hat{s}_t$
22:         end for
23:     end for
24:     if ($|\beta| > \psi$) then
25:         $P = P - PL$
26:         $B = B \cup PL$
27:     end if
28:     $p_i \leftarrow p_{i+1}$
29: end while
30: return $B$ and $P$
```

parameter, c_1 and c_2 represent learning rate parameters and ran_1 and ran_2 represent random numbers (where ran_1 and $ran_2 \in [0, 1]$). The first term (in Eq. (4.18)) $wV_j(s, a)$ represents the inertia of the particle and the second term (in Eq. (4.18)) $c_1 ran_1(p_b^i(j) - Q_j^i(s, a))$

represents that a participant (user) p_i learns the personal experience from the tweet posted by user (i.e., by choosing the temporal features and spam content in the tweet which are extracted from each tweet). The term $Q^i_j(s,a)$ represents the i^{th} particle Q-value that has obtained at j^{th} iteration from the personal experience of choosing learning action a in state s. The term $p^i_b(j)$ represents the local best position of personal experience that particle has obtained at j^{th} iteration by choosing the optimal Q-value with respect to state s and learning action a. The third term (in Eq. (4.18)) $c_2 ran_2(g_b(j) - Q(s,a))$ represents that participant (user) p_i learns the social experience from the social interactions among other users'. Further, $g_b(j)$ represents the global best position of particle that has been obtained from social experiences of other users' at j^{th} iteration. The term $Q^i_j(s,a)$ and $Q^i_{j+1}(s,a)$ represent the current Q-value (i.e., position) of i^{th} particle at j^{th} iteration and new Q-value of i^{th} particle at $(j+1)^{th}$ iteration, respectively.

A *Particle Swarm Optimization (PSO)* algorithm (refer Algorithm 4.3) has been proposed to find optimal action sequences. Q-values (s,a) and velocities $V(s,a)$ will be initialized for all state-action pairs. For each particle (i.e., representing a state), the long-term immediate reward R^{long} is computed by equation (4.17). The global best action sequences is identified from all particles which has highest long-term immediate reward value. Moreover, if the current long-term immediate reward value is better than local best action sequences, then identify the current immediate reward position as local best action sequences. Otherwise, the position of local best action sequences remains unchanged. Based on R^{long}, the local best p^i_b and global g_b best action sequences are determined at each iteration. Further, *Q*-values and velocities are updated by Equation (4.19) and Equation (4.18), respectively. This process is executed repeatedly until the variation of long-term immediate reward becomes negligible.

A *Particle Swarm Optimization based Deep Q-Learning (P-DQL)* algorithm (refer Algorithm 4.4) has been proposed for social spam bot detection. For *P-DQL* algorithm, $Q(s,a)$ values in deep *Q*-network are initialized for all state-action pairs. For each participant $p_i \in P'$ (i.e., P' is the set of all participants), the spam content and temporal features are extracted. Each participant feature set F_{p_i} is associated with state vector S_i. The learning agent initializes a beginning state s from a state vector and finds an optimal action sequences for a state based on particle swarm optimization algorithm (refer Algorithm 2). For each state, the learning agent stores an optimal action into an experience tuple $e_t =< s_t, a_t, r(s_t, a_t, b), \hat{s_t} >$ into replay memory r_m. The deep Q-network is updated by minimizing loss, Loss= $(y - Q(s, a))^2$ where $y = r(s_i, a_i, b) + \gamma$ Q($\hat{s}$, arg max$_{\hat{a}}Q(\hat{s}, \hat{a})$). Moreover, if current state s reaches k^{th} goal state (i.e., detected as a bot based on state behavior) then the participant is detected a social spam bot. A set of social spam bots are identified from *Twitter* network.

The proposed *P-DQL* algorithm converges faster (with multiple learning agents) to find an optimal sequence of actions in order to reach-out a goal state. Moreover, the proposed algorithm requires less storage space because the learning agent (i.e., proposed *P-DQL* algorithm) stores only global best sequences into replay memory instead of storing all possible state-action pairs. In particle swarm optimization, the temporal features (such as average number of tweets posted per day, longest user session time and percentage of dropped followers) are tuned to obtain optimal action sequences.

4.4 Influence Bots in Twitter

In order to influence the user in the *Twitter* network, the social bots may post malicious information in 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 work, the most influential users are identified, where they 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

$$UI_{p_i} = \begin{cases} I^T(p_i) + I^I(p_i) & \text{if participant (user) } p_i \text{ follows participant } p_j \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

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

4.4.1 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

$$I^T(p_i) = Co(tw) + Li(tw) + RT(tw) + RE(tw) \tag{4.22}$$

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.

4.4.2 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 4.2.2.3). If the user's degree centrality is more, 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

Algorithm 4.5 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 $tw$ ∈tweets do
3:          Compute influence value of user's tweet $I^T_{p_i}$ by using Equation (4.22)
4:      end for
5:      Compute influence of user's interactions $I^I_{p_i}$ by using Equation (4.23)
6:      Compute user influence score $UI_{p_i}$ by using Equation (4.21)
7:  end for
```

the user's betweenness centrality is high, then the user can quickly share tweet to the entire *Twitter* community through 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

$$I^I(p_i) = D_c(p_i) + CC(p) + B_C(p_i) + C(p_i) + PR(p_i) \tag{4.23}$$

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

4.4.3 Proposed Top-k Influential Users Algorithm

The proposed top-k influential algorithm (refer Algorithm 4.6) 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, the user influence value (refer Algorithm 4.5) has to be determined. Moreover, the users are ranked based on their influence value. Hence, the ranking value of each user is monitored between two consecutive iterations. The rank distance $d_r(k)$ is measured between the ranking of k^{th}-influential user in two

Algorithm 4.6 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:
1:  $UI$={}, K={}
2:  for ($i = 1; i \leq max; i++$) do
3:      for each participant $p_i \in P$ do
4:          s= User_Influence_score($p_i$)
5:          $UI = UI \cup s$
6:      end for
7:      Rank the users $R$ based on their influential value $UI$
8:      Obtain the top-k influential users $R_i$ at $i^{th}$ iteration
9:      Compute the rank distance $d_r(k)$ between $R_i$ and $R_{i-1}$ by using Eq. (4.24)
10:     if $d_r(k) \leq T$ then
11:         break
12:     else
13:         K=Update(K,R)
14:     end if
15: end for
16: return K
```

consecutive (i.e., at i^{th} and $i^{th} - 1$) iterations and it is defined as

$$d_r(k) = \sum_{i=1}^{K} |R_i(p_i) - R_{i-1}(p_i)| \tag{4.24}$$

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 leads to high accuracy of identifying the most influential users.

4.5 Performance Evaluation

In this section, the experimental results are presented to evaluate the performance of the proposed single and multi-agent *Deep Q-Learning* algorithms by considering real-world datasets collected from the *Twitter* network, such as *The Fake Project* dataset, *Social Honeypot* dataset and *User Popularity Band* dataset (the dataset is partitioned into four groups based on number of followers) [27]. The details of three different *Twitter* datasets are presented in Table 4.4. The proposed algorithms are offline deep reinforcement learning algorithms, in each case a deep Q-network is trained to determine an optimal action for a given state. The proposed algorithms adapted offline process in order to train the deep Q-networks with the collected offline data (i.e., from each dataset) in terms of series of state, action and reward. The proposed algorithms are trained and tested in an offline process where a massive volume of offline data makes deep Q-network highly stable with less number of iterations [211]. Therefore, once the proposed algorithms identify social bots, then an appropriate action can be taken to isolate social bots by the Twitter network. The proposed single agent *Deep Q-Learning* algorithm (with three hidden layers) has been compared with the other existing algorithms, such as feed-forward neural network (*FFNN*) [28], deterministic *Q*-Learning (QL) [29] and regularized deep neural network (*RDNN*) [30]. Further, the proposed multi-agent *P-DQL* has been compared with the proposed adaptive single-agent deep Q-learning (*ADQL*) algorithm and with the existing algorithms, such as *FFNN*, *RDNN*, content-based deep reinforcement learning (*C-DRL*) [212] and social network analysis-based deep reinforcement learning (*SNA-DRL*) [213]. Further, the proposed top k-influential users algorithm has been compared with other existing algorithms, such as degree centrality based radius-neighborhood (*DERND*) [32], suspected infected recovered (*SIR*) diffusion model [33] and true-top [34]. The performance of the proposed single and multi-agent *Deep Q-Learning* algorithms are evaluated in terms of precision, recall (true positive rate) and F-measure (refer section 4.5.1). The following metrics are defined for the performance evaluation of the proposed algorithms:

- *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 Negative (FN)*: the total number of users detected as legitimate users, which are actually social bots,
- *False Positive (FP)*: the total number of users detected as social bots, which are actually legitimate users.
- *True positive rate (or Recall)*: It is defined as $\frac{TP}{TP+FN}$,
- *False positive rate*: It is defined as $\frac{FP}{FP+TN}$,
- *Precision*: It is defined as $\frac{TP}{TP+FP}$.
- *F-measure*: It is defined as $\frac{2\times Precision\times Recall}{Precision+Recall}$
- *G-measure*: It is defined as $\sqrt{Precision \times Recall}$

Table 4.4: Summary of datasets collected from Twitter

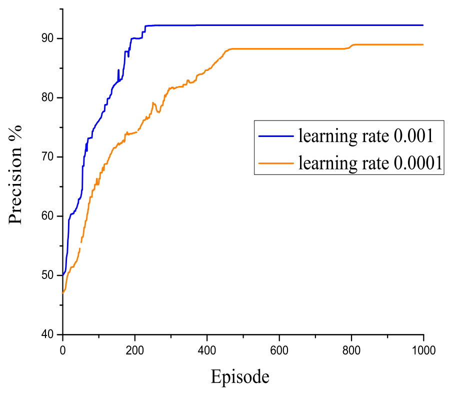
	Dataset Name	Human	Bots	Total Accounts	Tweets
Dataset 1	The Fake Project	3474	991	4465	9,987,698
Dataset 2	Social Honeypot	19,276	22,223	41499	5,613,166
Dataset 3	User Popularity				
	Band 10M	26	24	50	150,336
	Band 1M	450	296	746	303,517
	Band 100K	740	707	1447	230,577
	Band 1K	794	499	1293	37,679

4.5.1 Experimental Results for Single-Agent Deep Q-Learning

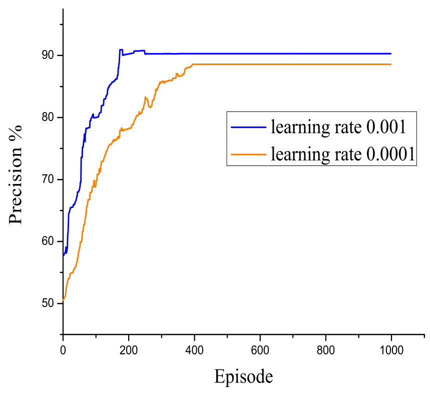
Fig. 4.4(a), Fig. 4.4(b) and Fig. 4.4(c) show the (convergence) performance of the proposed *DQL* algorithm with two different learning rate parameter values i.e., $\epsilon = 0.001$ and $\epsilon = 0.0001$. It can be observed that the proposed algorithm quickly converges with a learning

Table 4.5: List of parameters

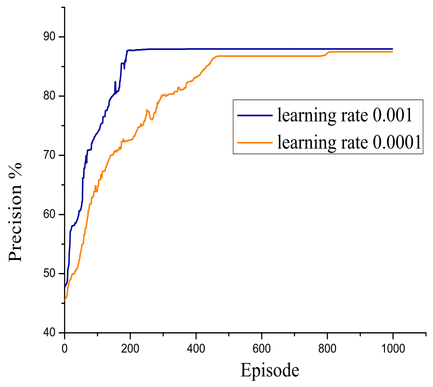
Parameter	Value
Learning rate	0.001
Discount factor	0.99
Mini Batch size	32
Replay memory size	50,000



(a) The Fake Project Dataset



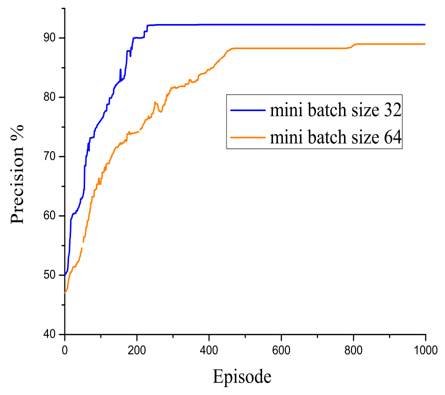
(b) Social Honeypot Dataset



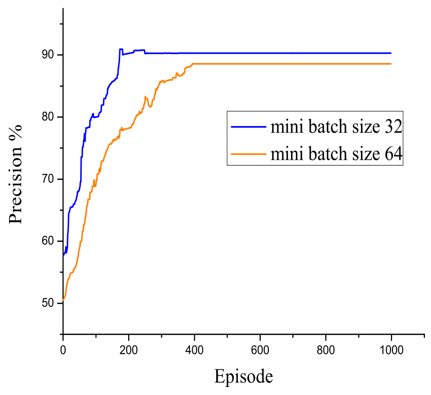
(c) User Popularity Band Dataset

Figure 4.4: Comparison of precision value with different learning rate parameter values

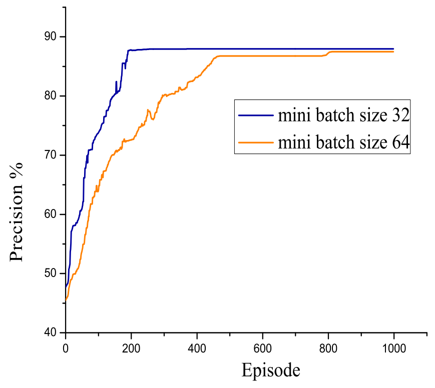
rate $\epsilon = 0.001$ when compared to $\epsilon = 0.0001$. Moreover, a higher learning rate leads to a local optimum in order to obtain higher precision value. From Fig. 4.4, it can be observed



(a) The Fake Project Dataset



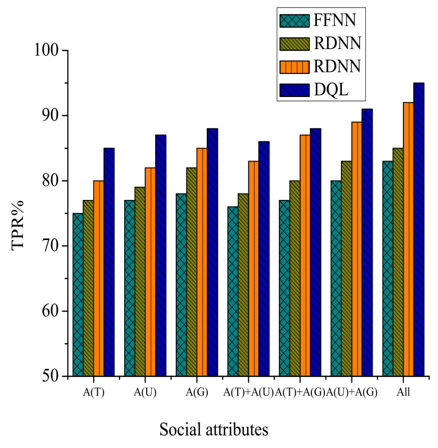
(b) Social Honeypot Dataset



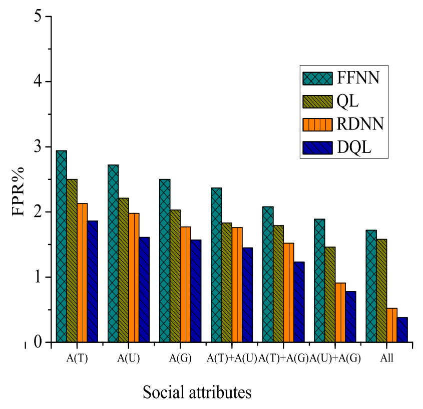
(c) User Popularity Band Dataset

Figure 4.5: Comparison of precision value with different mini batch sizes

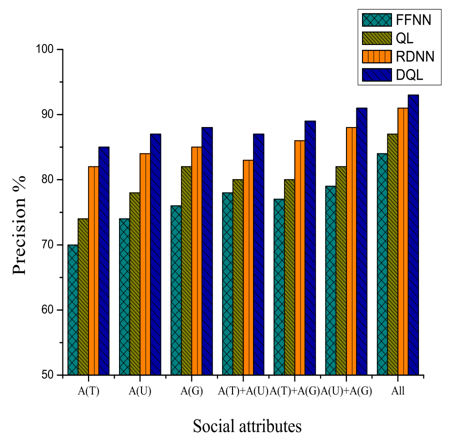
that for learning rate $\epsilon = 0.001$, the precision value is more than 90% (on an average) for social bot detection. Further, lowering learning rate value below 0.0001 will give lower precision. The convergence of target Q-function is also affected by other parameters, such as discount factor and mini batch size. The parameter values that are used for computing the target Q-values are listed in Table 4.5. 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 [210]. Therefore, discount factor $\gamma = 0.99$ has been chosen. Fig. 4.5(a), Fig. 4.5(b) and Fig. 4.5(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 [214]. From Fig. 4.5, it has been observed that the 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) for social bot detection. Further, increasing mini-batch size (i.e., greater than 64) will give lower precision.



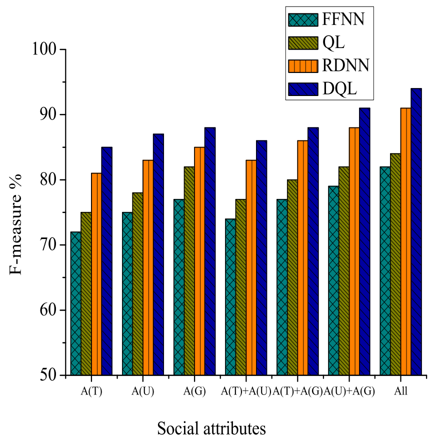
(a) True Positive Rate



(b) False Positive Rate

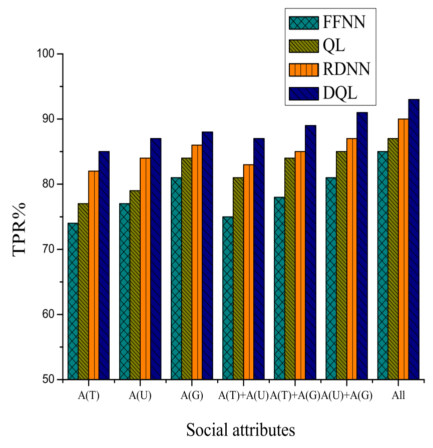


(c) Precision

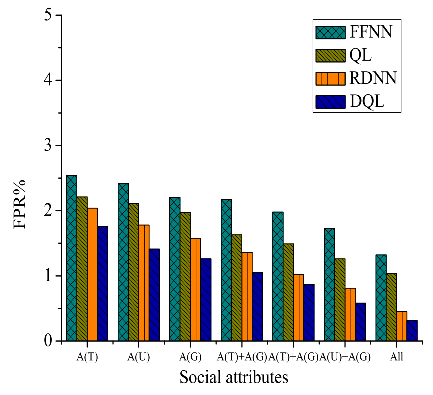


(d) F-measure

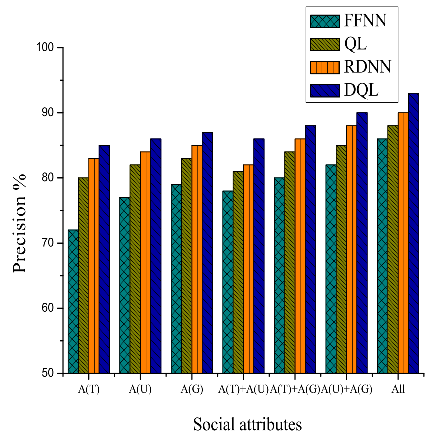
Figure 4.6: Experimental results by considering all possible combinations of social attributes on *The Fake Project* dataset



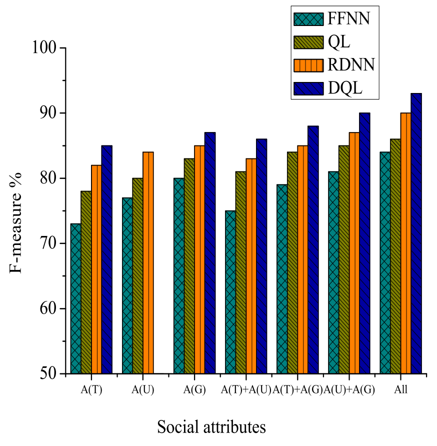
(a) True Positive Rate



(b) False Positive Rate



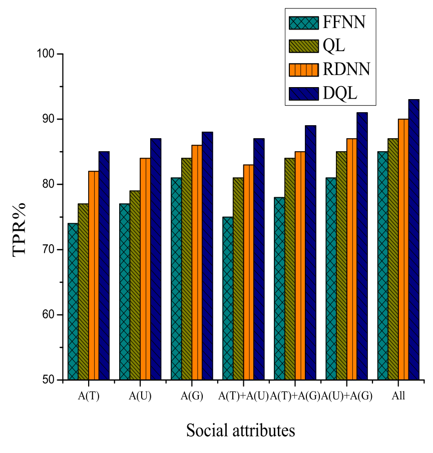
(c) Precision



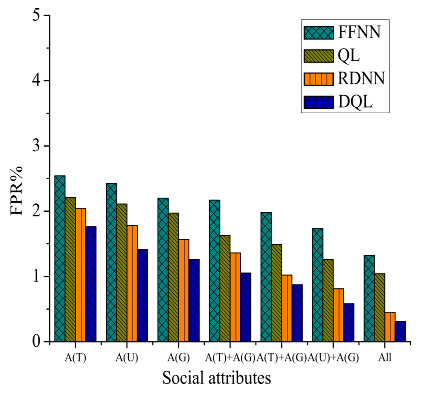
(d) F-measure

Figure 4.7: Experimental results by considering all possible combinations of social attributes on *Social Honeypot* dataset

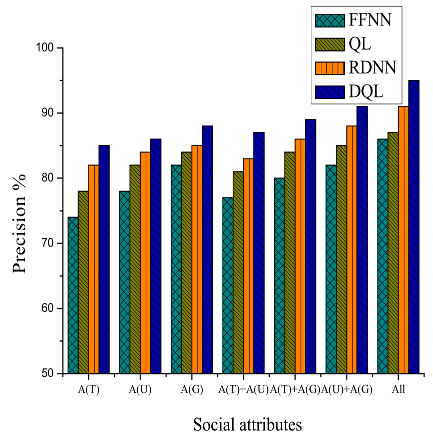
A set of social attributes, such as tweet-based attributes (i.e., from the content of each user tweet), user profile-based attributes (from a series of each user's tweets) and social graph-based attributes (i.e., the user establishes the social relationship with their friends and followers) are considered and they are denoted as $A(T)$, $A(U)$ and $A(G)$ respectively (discussed in Section 4.2.2). The performance of the proposed *Deep Q-Learning (DQL)* algorithm has been compared with other existing algorithms (such as feed-forward neural network (*FFNN*), deterministic Q-Learning (*QL*) and regularized deep neural network (*RDNN*) 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).



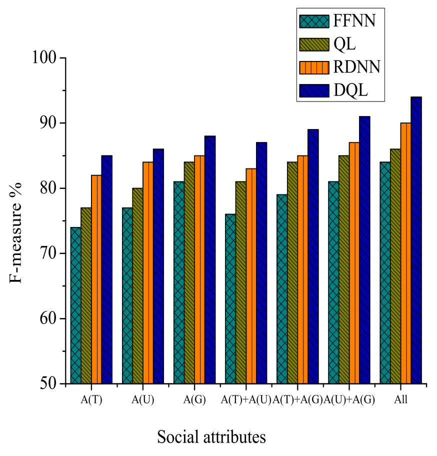
(a) True Positive Rate



(b) False Positive Rate



(c) Precision



(d) F-measure

Figure 4.8: Experimental results by considering all possible combinations of social attributes on *User Popularity Band* dataset

From Fig. 4.6, it has been observed that all the algorithms can obtain the best social bot detection performance by considering all the three different types of social attributes. When only user profile-based attributes are considered, the social bot detection performance of the proposed *DQL* algorithm and *RDNN* has been fallen down from 87% to 84% respectively on precision value. From Fig. 4.6, it can also be observed 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 up to 5-9% on precision value. Therefore, by combining all the social attributes, the social bot detection performance has been improved up to 4-10% on precision value. From Fig. 4.7, it has been observed that by combining the tweet-based attributes with the user profile-based attributes, the social bot detection performance has been improved up to 4-8% on precision value. Therefore, by combining all the social attributes, the social bot detection performance has been improved up to 3-8% on precision value. From Fig. 4.8, it can be observed that by combining the tweet-based attributes with the user profile-based attributes, the social bot detection performance has been improved up to 4-9% on precision value. Therefore, by combining all the social attributes, the social bot detection performance has been improved up to 5-10% on precision value. Table 4.6 shows performance of proposed *DQL* algorithm for 5-fold cross-validation.

Table 4.6: 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 4.7: Average Execution time for the proposed DQL algorithm and the existing QL algorithm

	Execution time in seconds		
Dataset	DQL-1	DQL-2	DQL-3
The Fake Project	2521	2754	2846
Social Honeypot	1521	1676	1707
User Popularity	904	972	1012

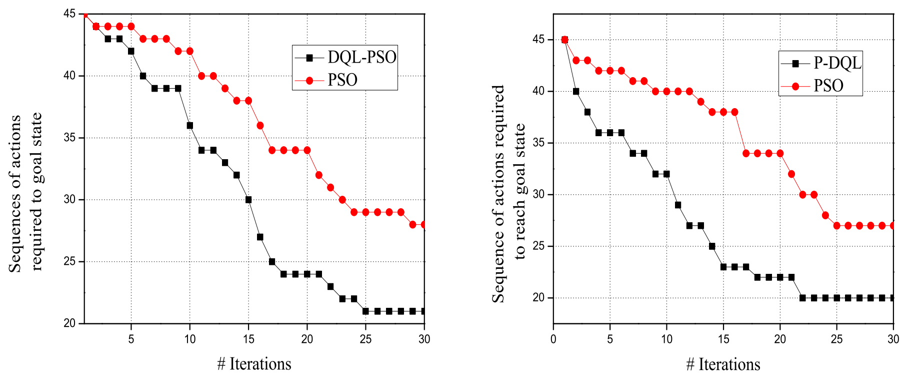
Table 4.7 shows the average execution time for the proposed *Deep Q-Learning* (*DQL*) algorithm. The average execution time for the *DQL* algorithm is computed with one, two and three hidden layers, which are denoted as *DQL-1, DQL-2* and *DQL-3*, respectively. As the number of hidden layers increase, 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.

4.5.2 Experimental Results for Multi-Agent Deep Q-Learning

Table 4.8: *P-DQL* parameters

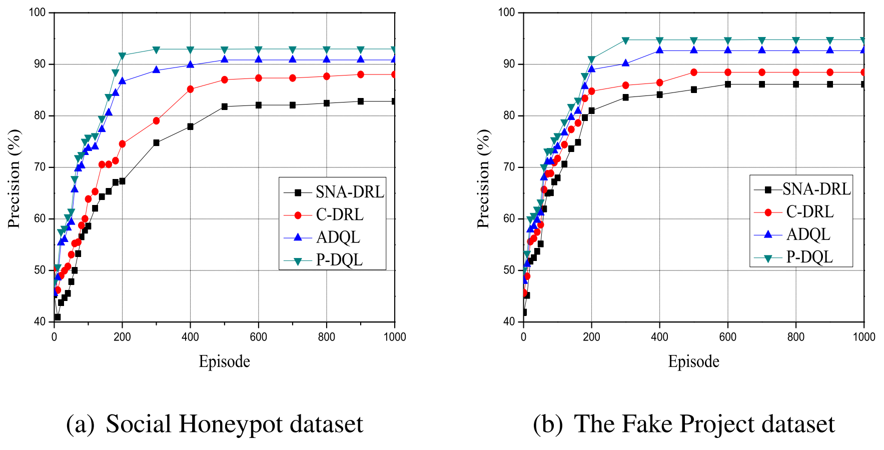
Parameter	Value
learning rate ϵ	0.0001
Mini Batch size	32
DQN discount rateγ	0.99
Number of hidden layers	3



(a) Social Honeypot dataset (b) The Fake Project dataset

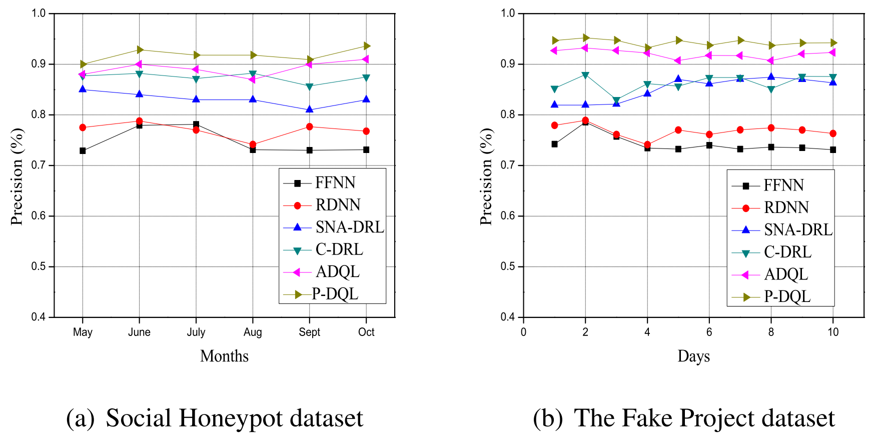
Figure 4.9: Optimal learning action sequences needed to reach terminal state

Fig. 4.9 shows comparison of proposed particle swarm optimization based deep Q-learning (P-DQL) algorithm with existing particle swarm optimization (*PSO*) algorithm [31]. Number of learning actions required to reach terminal (or goal) state versus number of iterations for P-DQL algorithm is depicted in Fig 4.9. From Fig. 4.9, it is observe that less number of actions are required for P-DQL to reach terminal (or goal) state. This is due to the fact that deep Q-network considers the updation strategy of Q-value based on global and local best action sequences which helps to reach terminal state quickly. The proposed



(a) Social Honeypot dataset (b) The Fake Project dataset

Figure 4.10: Convergence Performance of P-DQL and other existing algorithms (SNA-DRL, C-DRL and ADQL)

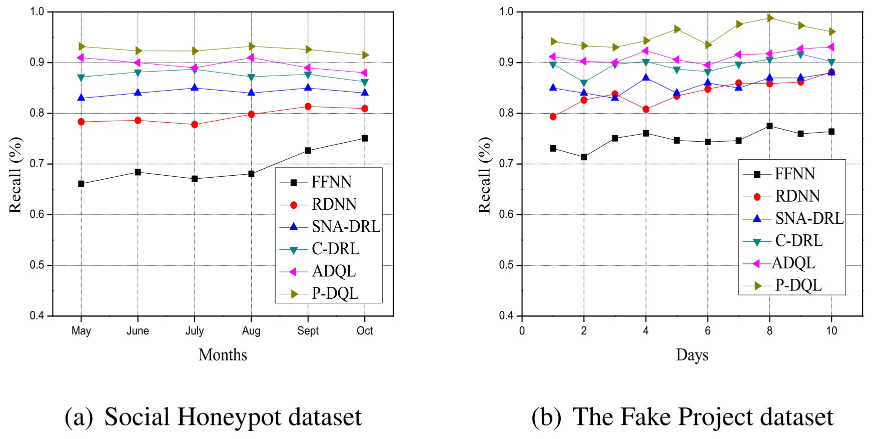


(a) Social Honeypot dataset (b) The Fake Project dataset

Figure 4.11: Precision of P-DQL and other existing algorithms (FFNN, RDNN, SNA-DRL, C-DRL and ADQL)

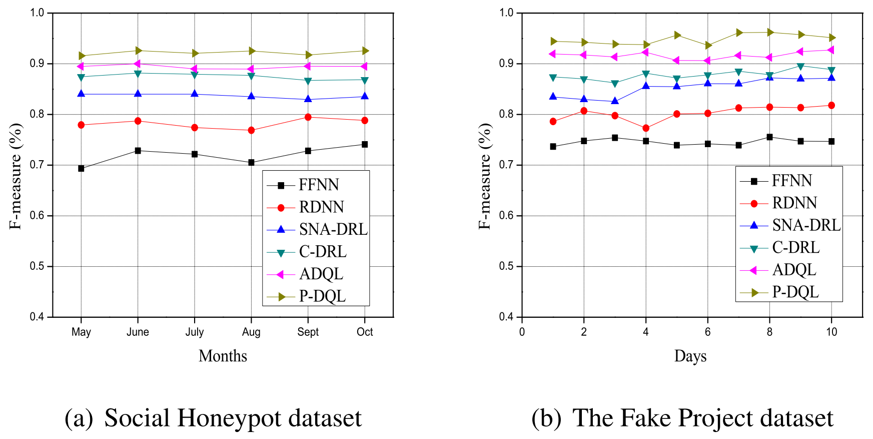
P-DQL algorithm converges quickly with less number of iteration. Hence, this presents that P-DQL is better than *PSO* in terms of faster convergence rate.

Fig. 4.10 shows comparison of the proposed *P-DQL* and adaptive single-agent deep Q-learning (*ADQL*) algorithm with other existing algorithms namely, content-based deep reinforcement learning (*C-DRL*) [212] and social network analysis-based deep reinforcement learning (*SNA-DRL*) [213] on two real-time *Twitter* datasets in terms of precision. The proposed *P-DQL* obtains better precision value when compared to other existing algorithms. The proposed *P-DQL* algorithm considers multiple learning agents to find an



(a) Social Honeypot dataset (b) The Fake Project dataset

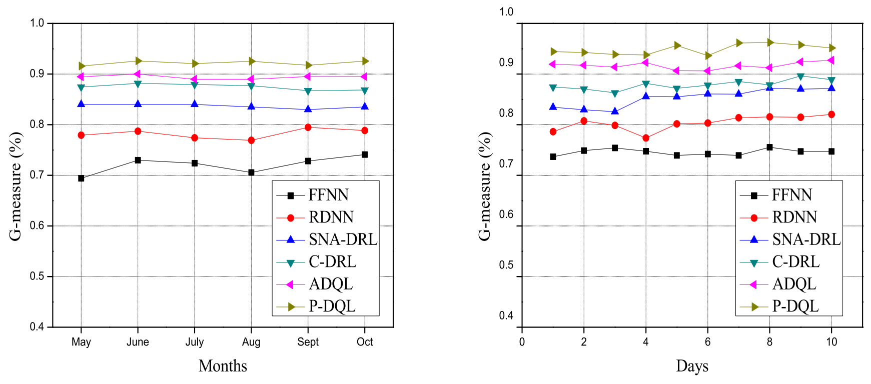
Figure 4.12: Recall of P-DQL and other existing algorithms (FFNN, RDNN, SNA-DRL, C-DRL and ADQL)



(a) Social Honeypot dataset (b) The Fake Project dataset

Figure 4.13: F-measure of P-DQL and other existing algorithms (FFNN, RDNN, SNA-DRL, C-DRL and ADQL)

optimal policy at faster convergence rate with less number of iterations as compared with deep reinforcement learning algorithms such as *ADQL* , *C-DRL* and *SNA-DRL* (as shown in Fig. 4.10). Moreover, the existing deep reinforcement learning algorithms with a single agent selects an appropriate action in a longer learning time (as shown in Fig. 4.10). Moreover, the convergence of the proposed *P-DQL* algorithm is affected by parameters, such as *DQN* discount rate (γ), learning rate (ϵ) and mini-batch size. From Fig. 4.10, it can be observed that for *DQN* discount rate γ 0.99, mini-batch size 32 and learning rate ϵ 0.0001, the *P-DQL* algorithm has achieved highest precision (i.e., on an average more



(a) Social Honeypot dataset (b) The Fake Project dataset

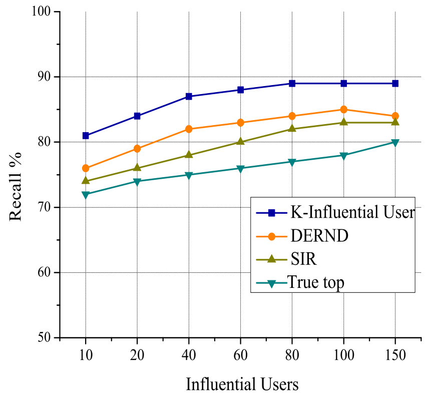
Figure 4.14: G-measure of P-DQL and other existing algorithms (FFNN, RDNN, SNA-DRL, C-DRL and ADQL)

than 90% precision) for social spam bot detection. In *ADQL*, deep Q-network selects a specific action depending on next state and stores all possible state-action pairs. Due to this the convergence of *ADQL* is slow when compared to the proposed *P-DQL* algorithm (as depicted in Fig. 4.10). It is observed that the precision value of proposed *P-DQL* algorithm is (on average) 11% higher than existing *SNA-DRL*.

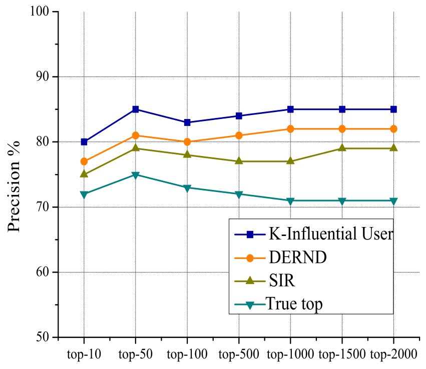
Fig. 4.11, Fig. 4.12, Fig. 4.13 and Fig. 4.14 show comparison of proposed *P-DQL* and adaptive single-agent deep Q-learning (*ADQL*) algorithm with other existing algorithms namely, feed-forward neural network (*FFNN*), regularized deep neural network (*RDNN*), content-based deep reinforcement learning (*C-DRL*) and social network analysis-based deep reinforcement learning (*SNA-DRL*) on two real-time *Twitter* datasets in terms of precision, recall, F-measure and G-measure. From Fig. 4.11, it can be observed that precision value of *FFNN* algorithm is reduced from 78% to 73%. Moreover, the precision of *P-DQL* and *ADQL* is around 94% and 91%, respectively. This is due to the fact that the proposed P-DQL finds an optimal action based on updating Q-values (using PSO) with temporal features and spam content in the tweet because the behavior of spam bot rapidly changes over time as compared with existing algorithms such as FFNN, RDNN, ADQL, C-DRL and SNA-DRL (as shown in Fig. 4.11, Fig. 4.12, Fig. 4.13 and Fig. 4.14). It can be observed that from Fig. 4.12, the recall of *P-DQL* outperforms with other existing algorithms when testing on different months. For *The Fake Project* dataset and *Social Hon-*

eypot dataset, the recall value of *P-DQL* is about 15% and 12% higher than *FFNN* when the data is tested on different days and months, respectively. Fig. 4.12 shows that recall of *P-DQL* is above 90% for all months (and days). Especially, *FFNN* has lowest social spam bot detection which is around 78%. This happens due to the consideration of temporal features which can capture dynamic behavior of spam bots more accurately. For The Fake Project dataset, the performance of proposed algorithm is improved by 7% on F-measure over *C-DRL* algorithm. Therefore, F-measure and G-measure results show that proposed *P-DQL* algorithm outperforms other existing algorithms.

4.5.3 Experimental Results for Top-k Influential Users



(a) Recall



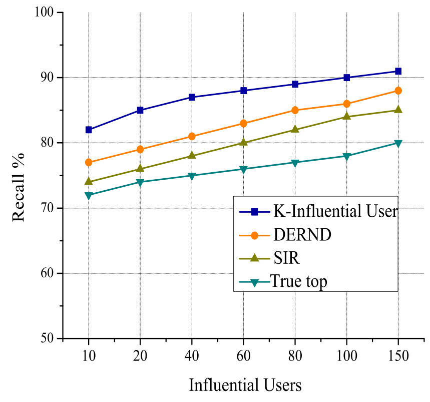
(b) Precision

Figure 4.15: Top-k Influential Users on *The Fake Project* Dataset

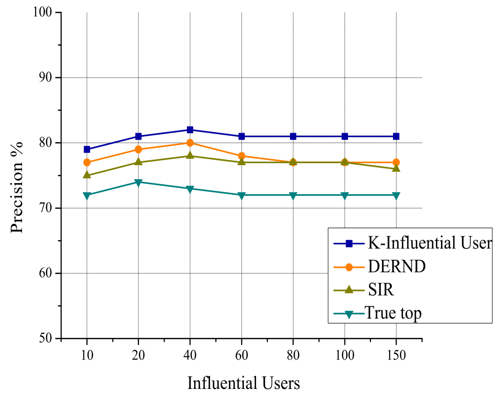
The performance of the proposed top-k influential users algorithm is evaluated by considering the following metrics.

- *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 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 $\frac{|LU_1(k) \cap LU_2(k)|}{|LU_2(k)|}$.

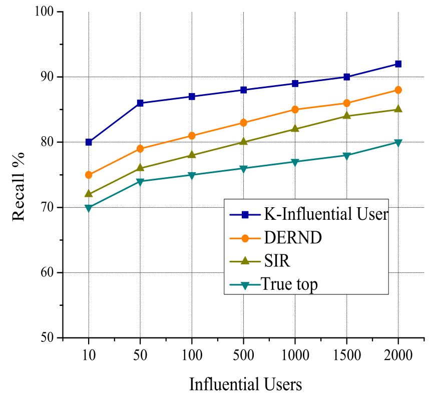


(a) Recall

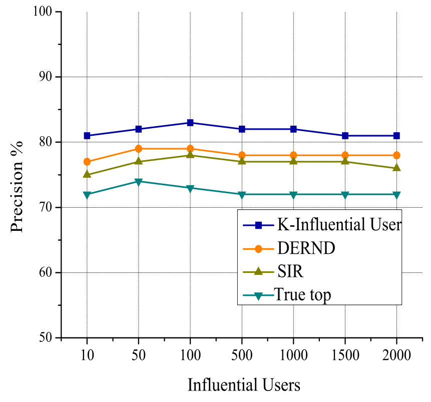


(b) Precision

Figure 4.16: Top-k Influential Users on *Social Honeypot* Dataset



(a) Recall



(b) Precision

Figure 4.17: Top-k Influential Users on *User Popularity Band* Dataset

Fig. 4.15, Fig. 4.16 and Fig. 4.17 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*) [32], suspected infected recovered (*SIR*) diffusion model [33] and true-top [34] (on all three different *Twitter* datasets). It can be observed 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 the proposed algorithm is approximately 80% as shown in Fig. 4.15(b), Fig. 4.16(b) and Fig. 4.17(b). This implies that the proposed algorithm can identify 80% of top-10% influential 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 Fig. 4.15(b), Fig. 4.16(b) and Fig. 4.17(b), it has been observed that *DERND* 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 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. It is observed that the proposed top-k influential users 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 algorithm selects the users which are influenced by social bots so that these users cannot further influence the current users.

4.6 Summary

In this chapter, a deep Q-network architecture has been designed by incorporating a single agent Deep Q-Learning (DQL) model using the social attributes in the Twitter network for detection of malicious social bots. A multi-agent deep Q-Learning algorithm has been proposed by using particle swarm optimization method with users' temporal features in order to detect malicious social bots in Twitter network. Moreover, each social attribute of a user is considered as a state and the learning agent's movement from one state to another state is considered as an action. In the proposed single agent and multi-agent *DQL* algorithms, the learning agent chooses a specific learning action with an optimal Q-value in each state for social bot detection. Further, an algorithm has been proposed to identify top-k influential Twitter network users (which are influenced by the social bots) based on the tweets and the users' interactions. In the next chapter, social botnet and spam influential community detection approaches have been presented.

Chapter 5

Detection of Social Botnet and Spam Influential Communities

A *social botnet* is a group of social bots created and controlled by a botmaster (acting as a leader among social bots) and performs malicious activities, such as creating multiple fake accounts, spreading spam, manipulating online ratings, and so on [8], [215]. To protect against botnet attacks, existing social botnet detection approaches [49], [50], [81] have mostly focused on the tweet content and social interactions among the participants in the Twitter network. In [73], [216], some methods have been proposed to identify fraudsters who sell legitimate online social networking accounts created in the Twitter network. If a botmaster (malicious user) is willing to buy legitimate accounts from fraudsters, then it can compromise a larger number of legitimate participants (OSN accounts) by creating attack edges between them and the social bot. In fact, the botmaster can perform devastating malicious activities, such as spreading social spam content, manipulating online ratings and recommendations [50]. The social bots can also re-tweet the malicious tweets posted by the botmaster [81]. Furthermore, the botmaster may create multiple fake identities and attempt to establish social relationship between a larger number of legitimate participants to avoid detection [81], affecting the quality of experience for Twitter users. Therefore, detecting malicious bots is an important problem.

To protect against botmaster attacks, in this chapter, a weighted signed Twitter network graph is constructed based on the trust values and behavioral similarity between pairs of participants. For evaluating the trust value of each participant, a random walk model [217] has been adopted in which each participant moves to one of its neighboring participants with equal probability. The proposed trust-driven random walk model evaluates the trust value of each participant by considering important features, such as tweet content, URL-based, graph-based, profile-based features and influence value of the neighboring participants. The behavioral similarity of the participants are analyzed by considering tweet-content similarity, shared URL similarity, interest similarity, and social interaction similarity for identifying similar type of behavior (malicious or non-malicious) among the participants in the Twitter network. Next, a *Social Botnet Community Detection (SBCD)* algorithm is proposed by considering the behavioral similarity matrix in order to identify the social botnet communities in the weighted signed Twitter network graph. A *Deep Autoencoder based Social Botnet Community Detection* (DA-SBCD) algorithm is proposed to reconstruct and detect social botnet communities with different types of malicious activities. Further, a *Spam-Influential Users and Influential Community Detection (SIU-ICD)* algorithm has been proposed to identify the spam influential communities $C = \{c_1, ...c_m\}$ in *Twitter* network. Finally, the effectiveness of SBCD, DA-SBCD and SIU-ICD are analyzed experimentally in terms of normalized mutual information (NMI), precision, recall, F-measure and modularity.

The novel contributions of this chapter are summarized as follows:

- Analyze the participants' behavioral features to identify malicious and non-malicious participants in the Twitter network.

- Evaluate the trust value of each participant based on several features and influence values of the neighboring participants.

- Design SBCD algorithm to detect social botnet communities of social bots having higher malicious behavioral similarity.

- Based on deep autoencoder model, develop DA-SBCD algorithm to reconstruct and detect social botnet communities that exhibit better performance.

- Develop a *Spam Influential users and Influential Community Detection (SIU-ICD)* algorithm to minimize the disseminating of spam-content through influential communities in Twitter network.

- Conduct experiments with two Twitter datasets to demonstrate the efficacy of the proposed algorithms in terms of normalized mutual information, precision, recall and F-measure.

This chapter is organized as follows. Section 5.1 presents the problem formulation. Section 5.2 deals with the detection of social botnet communities using deep learning. After analyzing the participants' behavioral features and trust-driven random walk model, this section describes the Social Botnet Community Detection (SBCD) algorithm and and DA-SBCD) algorithm using deep autoencoder. Section 5.3 presents *SIU-ICD* algorithm to detect spam influential communities in Twitter network. Section 5.4 presents the experimental results while the final section offers summary of this work.

5.1 Problem Formulation

Given a Twitter network $G = (P, E)$, where the vertex-set $P = \{p_1, p_2, ..., p_n\}$ represents the set of participants (i.e., OSN accounts) and $E = \{< p_i, p_j >\}$ is the set of directed edges representing the social relationship between pairs of participants $p_i, p_j \in P$. The weight on a directed edge is based on the behavioral similarity features, such as tweet similarity, shared URL similarity, interest similarity, and interaction similarity (see Section 5.2.2). The social trust relationship between two participants p_i and p_j determines the $sign_{i,j}$ of each directed edge $< p_i, p_j >$. Furthermore, a normalized weighted behavioral similarity matrix $S = [w_{ij}]_{n\times n}$, where w_{ij} is the weight of the edge between p_i and p_j and $n = |P|$ is the number of participants. Now S is constructed based on the weighted edges with a goal to partition the participants into different communities (groups), $C = \{c_1, c_2, ..., c_m\}$, where m represents the desired number of communities. The participants belonging to the same community are assumed to have similar type of behavior (malicious or non-malicious) and higher behavioral similarity.

As illustrated in Fig. 5.1(a), the botmaster P_1 usually establishes a strong social re-

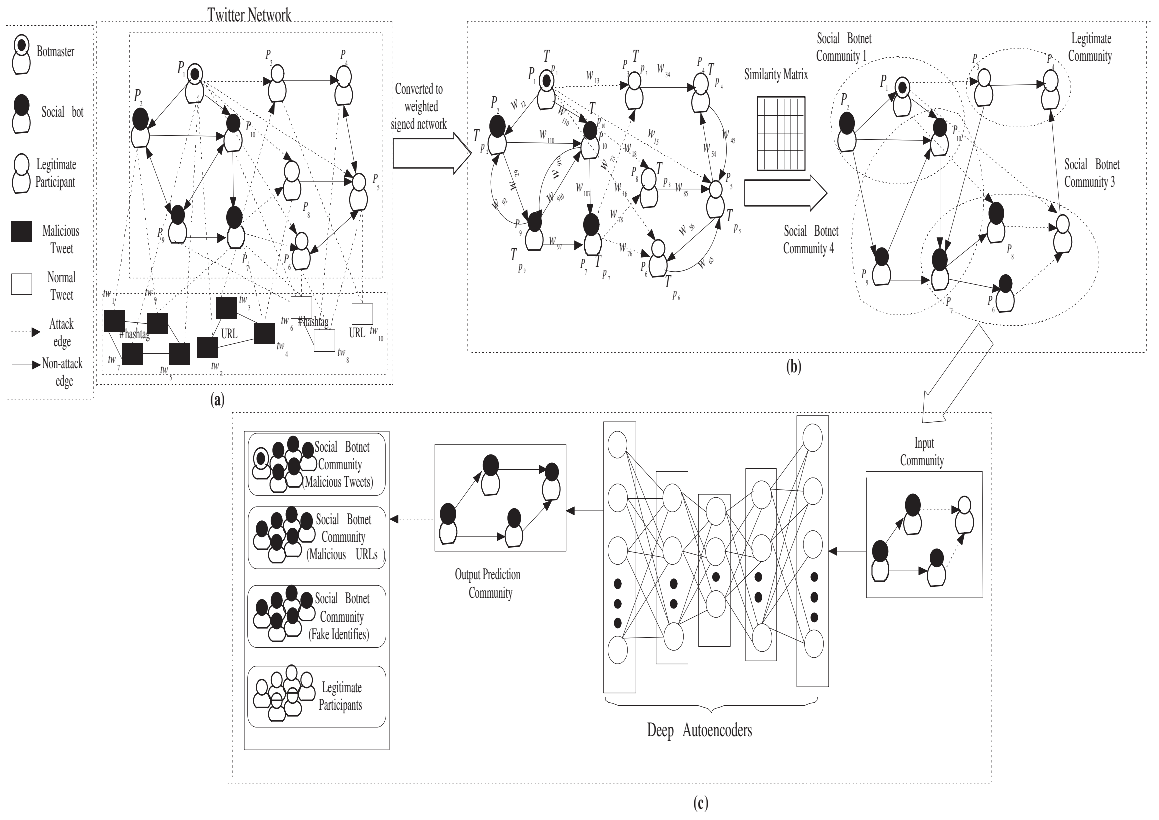

Twitter Network
Botmaster
Social bot
Legitimate Participant
Malicious Tweet
Normal Tweet
Attack edge
Non-attack edge
(a)
Converted to weighted signed network
Similarity Matrix
Social Botnet Community 1
Legitimate Community
Social Botnet Community 3
Social Botnet Community 4
(b)
Input Community
Deep Autoencoders
Output Prediction Community
Social Botnet Community (Malicious Tweets)
Social Botnet Community (Malicious URLs)
Social Botnet Community (Fake Identifies)
Legitimate Participants
(c)

Figure 5.1: Deep learning architecture for social botnet community detection

lationship with other social bots P_2, P_7, P_9 and P_{10}. In this attack model, the botmaster constructs a re-tweeting graph, where each node represents the social bot and each directed edge represents a re-tweeting relationship between two participants (users). The botmaster thus creates malicious tweets (with fake information or malicious URL in the tweet) and the social bots re-tweet them. Additionally, the social bots can spread malicious tweets to other legitimate participants. Indeed, the aim of the botmaster is to spread spam content by creating multiple fake identities. Moreover, in Fig. 5.1, the botmaster P_1 establishes a strong social relationship not only with social bots (P_2, P_7, P_9 and P_{10}) but also with legitimate participants (P_3, P_5 and P_8). If a botmaster is willing to buy the legitimate accounts from fraudsters, then the botmaster can have more legitimate friends by creating attack edges (e.g., between P_1 and P_3). This type of attack will influence legitimate users by affecting users' behavior, opinions and emotions. Many such malicious activities can be performed either by the botmaster or social bots.

Definition 1 (Signed Edge Set): Let $E' \subseteq P \times P \times sign$ represent a set of directed edges between the pairs of participants in P and $sign \in \{1, -1\}$. If $sign_{ij} = 1$, it implies there exists a trusted (non-attack) edge from participant p_i to p_j. On the other hand, if $sign_{ij} = -1$, then there exists an attack (or untrusted) edge between a social bot and a legitimate participant from p_i to p_j.

Definition 2 (Weighted Signed Twitter Network Graph): A weighted signed Twitter network graph $G' = (P, E', T, S)$ is constructed from four components: a participant set P, a signed edge set (with trusted and untrusted edges) E', a trust vector T for all participants, and a weighted behavioral similarity matrix S.

Problem (Social Botnet and Spam Influential Communities Detection): A Twitter network graph $G = (P, E)$ is given with the set of (malicious or non-malicious) activities performed by each participant $p_i \in P$. The objective is to construct a weighted signed Twitter network graph $G' = (P, E', T, S)$ with the set of trusted (non-attack) and untrusted (attack) weighted edges based on the behavioral similarity among the participants with trust values. Here G' is used to identify the social botnet communities with different types of malicious activities. The goal is to determine three functions $f : G \rightarrow G', g : G' \rightarrow C = \{c_1, c_2, ..., c_m\}$ and $h : C = \{c_1, c_2, ..., m\} \rightarrow \tilde{C} = \{\tilde{c_1}, \tilde{c_2}, ..., \tilde{c_m}\}$ for distinguishing legitimate participants among social botnet communities C (with different types of malicious activities) and community structure $\tilde{C}$ is reconstructed for detecting social botnet communities with better accuracy. Further, to minimize influence of spreading spam content, spam influential communities (i.e., participants who are more influenced by spam bots) $C = \{c_1, ..., c_m\}$ have to be identified in Twitter network.

Detection of social botnet communities with different types of malicious activities is a challenging task. In the next section, a deep autoencoder model is applied to reconstruct the community $\tilde{c_i}$ from a community c_i based on trust parameter T and behavioral similarity matrix S. The proposed model is considered accurate when $c_i \approx \tilde{c_i}$, $\forall c_i \in C$, such that the participants belonging to the same community are more likely to have similar type of (malicious or non-malicious) behavioral similarities.

5.2 Deep Learning based Social Botnet Communities Detection

In this section, firstly a deep learning architecture is presented for social botnet community detection. Then the participants' behavioral features are analyzed from different perspectives to identify similar type of behavior (malicious or not) among the participants. The proposed trust-driven random walk model predicts attack edges based on the participants' behavioral features in the Twitter network. By considering both behavioral similarity measure and trust-driven random walk model, a social botnet community detection (SBCD) algorithm is presented. This is followed by a DA-SBCD (Deep Autoencoder based SBCD) algorithm to detect more accurately social botnet communities with different type of malicious activities.

5.2.1 Deep Learning Architecture for SBCD

Fig. 5.1 shows the proposed deep learning (deep autoencoder) architecture for detecting social botnet community. The architecture consists of two phases – community formation and community reconstruction (which identifies the communities more accurately). In the first phase, the Twitter network graph G is converted into a weighted signed Twitter network graph G' based on the trust parameter and the participants' behavioral similarity features, such as tweet similarity, shared URL similarity, interest similarity and interaction similarity. The weighted signed Twitter network graph G' is used for detecting social botnet communities with different types of malicious activities, such as posting malicious tweets, posting or redirecting to malicious URLs, and creating multiple fake identities). In the second phase, the architecture is integrated with deep autoencoder model consisting of two sub-phases, namely the encoder and decoder. The proposed model encodes an observed input community c_i with the set of trusted and untrusted weighted edges using a function f defined in Equation (5.13). In the decoding sub-phase, a reconstructed community structure $\tilde{c_i}$ is to be determined using the decoding function, i.e., $\tilde{c_i} \approx f(c_i)$, for social botnet community detection with better accuracy.

5.2.2 Participants' Behavioral Features

As mentioned, the behavioral features of the participants include tweet-content similarity, shared URL similarity, social interaction similarity, and interest similarity. These are discussed below.

5.2.2.1 Tweet-Content Similarity

Each tweet is represented as a term frequency inverse document frequency (TFIDF) feature vector $< TFIDF(w_1, tw, T_w), ..., TFIDF(w_l, tw, T_w) >$, where w_l represents (distinct) word in the tweet. The term $TFIDF(w, tw, T_w)$ represents the importance of w within a tweet tw and a set of tweets posted by a participant (denoted as T_w), which is computed as:

$$TFIDF(w, tw, T_w) = TF(w, tw) \times IDF(w, T_w) \tag{5.1}$$

where $TF(w, tw)$ is the ratio of the number of times the word w appears in the tweet tw and the total number of words in the tweet. The term $IDF(w, T_w)$ is computed as:

$$IDF(w, T_w) = \log \frac{|T_w|}{|\{tw \in T_w : w \in tw\}|} \tag{5.2}$$

where $|T_w|$ denotes the total number of tweets posted by a participant and $|\{tw \in T_w : w \in tw\}|$ represents the number of tweets containing the word w.

For any two tweets u and v, let the ***TFIDF*** feature vectors be ***U*** and ***V***, respectively. For example, if a tweet u consists of message "Hi good morning hello hello" and tweet v consists of message "hi good morning". The word "hello" is appearing twice in the tweet u. Thus TF("hello", u) = 2/4 = 0.5, TF("hello", v) = 0/2 = 0 and IDF("hello", T_w) = log(2/1) = 0.301. Finally, $TFIDF$("hello", u, T_w) = TF("hello", u) $\times$ IDF("hello",T_w) = $0.5 \times 0.301 \approx 0.150$ and $TFIDF$("hello", v, T_w) = TF("hello", v) $\times$ IDF("hello",T_w) = $0 \times 0.301 = 0$.

Let *a* denote the total number of tweets posted by p_i, and let *b* denote the total number of tweets posted by p_j. A tweet-content similarity matrix $M = [\hat{x}_{ij}]_{a \times b}$ is calculated using the cosine similarity measure. It is constructed such that each tweet of participant p_i is compared with all tweets of p_j. The cosine similarity measure of two *TFIDF* feature vectors *U* and *V* is defined as

$$x_{ij} = \frac{\sum_{k=1}^{m} U_k V_k}{\sqrt{\sum_{k=1}^{m} U_k^2}\sqrt{\sum_{k=1}^{m} V_k^2}} \tag{5.3}$$

where $U_k \in U$, $V_k \in V$, and m is the length of the feature vector. Considering threshold value ς, the values of x_{ij} are recomputed as

$$\hat{x}_{ij} = \begin{cases} 1, & \text{If } x_{ij} \geq \varsigma \\ 0, & \text{If } x_{ij} < \varsigma \end{cases} \tag{5.4}$$

The tweet-content similarity value TS_{ij} between two participants p_i and p_j is thus defined as:

$$TS_{ij} = \frac{|\{\hat{x}_{ij} \in M | \hat{x}_{ij} = 1\}|}{a \times b} \tag{5.5}$$

where the numerator represents the number of non-zero elements in the tweet-content similarity matrix M of dimension $a \times b$.

5.2.2.2 Shared *URL* Similarity

The shared URL similarity, US_{ij}, is defined as the number of identical shared URLs between two participants p_i and p_j. It is determined by the *Jaccard* coefficient:

$$US_{ij} = \frac{|US_i \cap US_j|}{|US_i \cup US_j|} \tag{5.6}$$

where US_i and US_j are the set of *URLs* which are shared by the participants p_i and p_j, respectively.

5.2.2.3 Social Interaction Similarity

It quantifies the rate at which the participants interact with the neighboring participants. The social interaction similarity, SS_{ij}, between p_i and p_j is determined by the cosine similarity based on their neighboring set:

$$SS_{ij} = \frac{|NB(p_i) \cap NB(p_j)|}{\sqrt{|NB(p_i)| \times |NB(p_j)|}} \tag{5.7}$$

where $NB(p_i)$ and $NB(p_j)$ represent the set of neighboring participants of p_i and p_j, respectively.

5.2.2.4 Interest Similarity

In [218], an associative ripple method has been proposed for representing Twitter data into the most relevant information based on the participant's interests. The associative ripple contains several circles clustered by ranking (from interior to exterior) based on their relevance. This implies that the data closest to the center are the most relevant information to a specific topic based on the interests of the participants. Therefore, this method is adopted to determine the interest similarity, IS_{ij}, between two participants p_i and p_j based on their current interests. Thus,

$$IS_{ij} = \frac{\sum_{k=1}^{K} w_{c_k}(|\{l : \text{ if } p_l \in p_{c_k}\}|)}{\sum_{k=1}^{K} w_{c_k}(|p_{c_k}|)} \tag{5.8}$$

where p_{c_k} represents the set of participants having similar type of topic-based interest and clustered on the k^{th} circle and K is the total number of circles. The term w_{c_k} represents the weight of each circle (which represents the ranking of relevant information from the center to the k^{th} circle). Therefore, the interest similarity value determines two participants' interactions based on the current interests among a group of participants.

Based on the above four behavioral similarity features, the weight w_{ij} of the edge between p_i and p_j is determined as:

$$w_{ij} = \varphi_1 \times TS_{ij} + \varphi_2 \times US_{ij} + \varphi_3 \times SS_{ij} + \varphi_4 \times IS_{ij} \tag{5.9}$$

where φ_1, φ_2, φ_3 and φ_4 are (positive) weighted shares of the corresponding features such that φ_1+φ_2+φ_3+$\varphi_4 = 1$.

The weighted behavioral similarity matrix is given by $S = [w_{ij}]_{n \times n}$, where n is number of participants. The behavioral similarity features help identify multiple fake accounts created by the botmaster attempting to establish social relationship with more legitimate participants via attack edges. The trust-driven random walk model proposed next predicts the attack edges by observing prior behavioral features of each participant in the Twitter network.

Table 5.1: Features to evaluate prior trust of each participant

Category	**Description**
Tweet Content	Tweet content based features capture linguistic factors such as frequency of words in tweets, number of hashtags and positive (or negative) sentimental score [47], [51],
URL	URL-based features deal with URL redirection properties such as frequency of URL appearing in a tweet, URL redirection length and Http-302 status code [23]
Profile	Profile based features include user meta-data features such as profile creation time, number of followers and followees [13]
Graph	Graph based features represent the behavioral pattern of information spreading among users based on the degree centrality, clustering coefficient and betweenness measures [12]

5.2.3 Trust-Driven Random Walk Model

In the random-walk based approaches, each participant moves to one of its neighbors with equal probability [217]. The prior trust value of each participant p_i as T'_{p_i} is initialized based on the *Bayesian* theorem [13]. The prior trust value is determined using Equation (5.11). After the initialization step, p_i equally assigns the same T'_{p_i} value to each of its neighboring participants, p_j. Next, p_i updates its trust value based on its neighbor's influence. The trust value of p_j is distributed to one of its neighbors (since each participant moves to one of its neighbors) with equal probability, $\frac{w_{ij}}{deg(p_j)}$. Here the numerator w_{ij} denotes weight of the edge between p_i and p_j and according to Equation (5.9). The denominator $deg(p_j) = \sum_{p_k \in NB(p_j)} w_{jk}$ is the sum of the weighted edges linked from participant p_j to each of its neighboring participants, p_k, where $NB(p_j)$ denotes the set of neighbors of p_j. The trust value T_{p_i} of p_i is determined as:

$$T_{p_i} = \gamma \; T'_{p_i} + (1-\gamma) \sum_{p_j \in NB(p_i)} T'_{p_j} \frac{w_{ij}}{deg(p_j)} \tag{5.10}$$

where

$$\begin{aligned}
&T'_{p_i} = pr(g = Trustworthy|A) = \\
&\left\{pr(g = Trustworthy) \times \prod_{i=1}^{h} pr(a_i|g = Trustworthy)\right\}/ \\
&\left\{pr(g = Trustworthy) \times \prod_{i=1}^{h} pr(a_i|g = Trustworthy) + \right. \\
&\left. pr(g = Untrustworthy) \times \prod_{i=1}^{h} pr(a_i|g = Untrustworthy)\right\}
\end{aligned} \tag{5.11}$$

Here $A = \{a_1, .., a_i, ...a_h\}$ represents a set of attributes (features listed in Table 5.1) and $\gamma \in [0, 1]$ represents the probability of random walk. The term $pr(g = Trustworthy|A)$ denotes the probability that an attribute set A belongs to the trustworthy group g. (Note that the participant may be in trustworthy or untrustworthy group.) If $T_{p_i} \geq T_f$, (a threshold value), then it implies that p_i is more likely to be a legitimate participant. On the other hand, $T_{p_i} < T_f$ implies that p_i is more likely to be a social bot.

For participant p_i, random variable $w_{p_i} \in \{1, -1\}$ is defined, where $w_{p_i} = 1$ implies that p_i is a legitimate participant and $w_{p_i} = -1$ implies that p_i is a social bot. The sign of the weighted edge $sign_{ij}$ is defined as:

$$sign_{ij} = \begin{cases} 1, & \text{if } w_{p_i} w_{p_j} = 1 \\ -1, & \text{if } w_{p_i} w_{p_j} = -1 \end{cases} \tag{5.12}$$

The trust model captures the attack edges between any two participants based on the feature set and behavioral similarity features as described above. In the next section, the weighted eigenvector centrality measure and friendship-characteristics of communities will be considered to detect the presence of a botmaster and social botnet communities, respectively.

5.2.4 Social Botnet Community Detection (SBCD) Algorithm

Algorithm 5.1 describes the SBCD algorithm. For each participant $p_i \in P$, the trust value T_{p_i} is computed (Lines 1-3), leading to the trust value $T =< T_{p_1}, T_{p_2}, ..., T_{p_n} >$ for all participants. Each element (w_{ij}) of the weighted behavioral similarity matrix S

is determined using Equation (5.9). For each directed edge, the sign of a weighted edge is determined by Equation (5.12) which helps to detect the signed weighted edge set E' with a set of attack and non-attack edges. Then the weighted signed Twitter network graph $G' = (P, E', T, S)$ is constructed (Lines 4-8). In each iteration, the signed edge $< p_i, p_j > \in E'$ with the lowest weighted behavioral similarity value is removed, each disconnected component is considered as a community, and matrix S is recomputed. This process is repeated until the desired number of communities are obtained (Lines 9-17). In Line 18, the disconnected components in G' are shown as $C = \{c_1, ..., c_m\}$. Line 20 executes Intra_Community_Reformation (Algorithm 5.2) to determine the updated communities for better accuracy. Algorithm 5.2 removes legitimate participants from a social botnet community and adds similar type of social bots to a social botnet community based on the sign of the weighted edges.

Algorithm 5.1 Social Botnet Community Detection (SBCD)

```
    Input: $G = (P, E)$: Twitter graph, $\varsigma$: Similarity threshold
    Output: Legitimate and social botnet communities
 1: for each participant $p_i \in P$ do
 2:     Compute trust value $T_{p_i}$ using Equation (5.10)
 3: end for
 4: for $< p_i, p_j > \in E$ do
 5:     $S[i][j] \leftarrow$ Compute weighted behavioral similarity $w_{ij}$ using Equation (5.9)
 6:     Compute sign of weighted edge using Equation (5.12)
 7: end for
 8: Obtain $G' = (P, E', T, S)$
 9: // Finding Primary Communities
10: for $1 \leq i \leq m$ do // $m$: desired number of communities
11:     for $< p_i, p_j > \in E'$ do
12:         if $S[i][j] < \varsigma$ then
13:             Delete the edge $(p_i, p_j)$ with the lowest weighted behavioral similarity value
14:             Recompute weighted behavioral similarity for all participants after removal
    of the edge.
15:         end if
16:     end for
17: end for
18: $C = \{c_1, ..., c_m\} \leftarrow$ disconnected components in $G'$
19: // Intra-Community Reformation
20: $C = \{c_1, ..., c_m\} \leftarrow$ Intra_Community_Reformation
```

Algorithm 5.2 determines which neighbors in the set B (i.e., the set of participants

Algorithm 5.2 Intra_Community_Reformation

```
    Input: $C = \{c_1, ..., c_m\}$ communities
    Output: $C = \{c_1, ..., c_m\}$ updated communities (i.e., $c_i$ contains updated list of participants)
 1: repeat
 2: $C = \{c_1, ..., c_m\} \leftarrow$ disconnected components in $G'$
 3: for $1 \leq i \leq m$ do
 4:     for $p_i \in c_i$ do //Pruning legitimate participants
 5:         if $\upsilon(p_i) < 0$ then
 6:             $c_i = c_i - \{p_i\}$
 7:         end if
 8:         Compute eigen vector centrality of $p_i$
 9:     end for
10:     $BM \leftarrow p_i$ with highest eigen vector centrality.
11:     for $p_i \in B$ do // $B$: set of participants adjacent to $c_i$
12:         if $\upsilon(p_i) > 0$ and $S[bm][i] \geq \varsigma$ then
13:             // Adding similar type of social bot to $c_i$
14:             $c_i = c_i \cup \{p_i\}$
15:         end if
16:     end for
17: end for
18: until communities are similar after two consecutive iterations
```

p_i adjacent to c_i in terms of at least one incoming or outgoing edge) should be added to the detected social botnet c_i. A higher value of $\upsilon_{intra}(p_i) = \sum_{p_j \in c_i} sign(p_i, p_j)$ signifies whether $p_i \in c_i$ is more likely to be within the social botnet community c_i. Similarly, a higher value of $\upsilon_{inter}(p_i) = \sum_{p_j \notin c_i} sign(p_i, p_j)$ signifies whether a participant p_i is more likely to have higher (malicious or non-malicious) behavioral similarity with participants that are outside the social botnet community. An improved way to detect a social botnet community is to increase the number of non-attack edges (between any two social bots) with higher behavioral similarity and decrease the number of attack edges (between the social bots and legitimate participants).

If $\upsilon(p_i) = (\upsilon_{intra}(p_i) - \upsilon_{inter}(p_i)) < 0$, then the participant p_i from c_i is removed. For each social botnet community, the weighted eigenvector centrality measure [219] is determined to identify the botmaster BM, the leader among the social bots. The weighted eigenvector centrality quantifies the influence of a participant based on the (intra community) strength and the number of social interactions with neighboring participants. The weighted eigenvector centrality of $p_i \in c_i$, denoted as $e_c(p_i)$, is based on its neighbors' eigenvector centrality and computed as $e_c(p_i) = \frac{1}{\lambda}\sum_{j=1}^{n} w_{ji}e_c(p_j)$, where λ is a constant and w_{ji} is behavioral similarity value between participants p_j and p_i (Lines 1-9, Algorithm 5.2). The participant with the highest weighted eigenvector centrality measure is selected as the botmaster BM (Line 10, Algorithm 5.2). If there exists higher behavioral similarity between the botmaster BM and participant p_i, and $\upsilon(p_i) \geq 0$, then the social bot p_i is added to c_i. This process is repeated until the social botnet communities remain unchanged over two consecutive iterations. The SBCD algorithm returns $C = \{c_1, ..., c_m\}$ as the detected social botnet communities and legitimate communities (Lines 11-18, Algorithm 5.2).

Thus, the community formation phase (SBCD algorithm) detects social botnet communities with different types of malicious behavior. In the next section, the output of each SBCD community is given as input to the deep autoencoder model to detect different types of social botnet communities more accurately.

5.2.5 Deep Autoencoder based SBCD Algorithm

A deep neural network autoencoder is used and it is one of the deep learning techniques [220], that is trained to reconstruct the social botnet communities more accurately. The deep autoencoder based SBCD algorithm (DA-SBCD) is presented in Algorithm 5.3, for which the inputs are the community set $C = \{c_1, ..., c_m\}$ and the number κ of hidden layers.

For each community c_i, the weighted behavioral similarity matrix S_{c_i} is constructed in such a way that each element w_{ij} in S_{c_i} is computed with the help of Equation (5.9), i.e., $w_{ij} \in \mathbb{R}^{U \times U}$, where S_{c_i} is a similarity matrix in the form $\mathbb{R}^{U \times U}$ and U represents the number of participants in the community c_i. Further, the trust matrix for i^{th} community is represented as $T_{c_i} = [T_{p1}, T_{p2}....]^T$, where $T_{p_i} \in \mathbb{R}^{U \times 1}$ and T_{c_i} is a trust matrix in the form $\mathbb{R}^{U \times 1}$. Equation (5.10) evaluates the trust value T_{p_i} of each participant $p_i \in c_i$ (Lines 1-6, Algorithm 5.3). By concatenating the similarity matrix S_{c_i} with trust matrix T_{c_i}, the aggregated matrix Z (i.e., $\mathbb{R}^{U \times V}$ where $V = U + 1$) will be constructed. Now Z along with the weighted similarity matrix and trust matrix are given as input to the deep autoencoder

Algorithm 5.3 Deep Autoencoder based SBCD

```
    Input: $C = \{c_1, ..., c_m\}$ communities, $\kappa$: number of hidden layers
    Output: $\tilde{C} = \{\tilde{c_1}, ...., \tilde{c_m}\}$ reconstructed communities
 1: $\tilde{C} = \phi$
 2: for each community $c_i \in C$ do
 3:     Compute normalized weighted behavioral similarity matrix $S_{c_i}$ using Equation (5.9)
 4:     for each participant $p_i \in c_i$ do
 5:         Compute trust value of $p_i$ (i.e., $T_{p_i}$) using Equation (5.10)
 6:     end for
 7:     Z ← concat($S_{c_i}, T_{c_i}$)
 8:     $\tilde{Z}$ ← Train the autoencoder model with $Z$ and $\kappa$ using Equations (5.13) and (5.14)
 9:     Execute hierarchical cluster on the rows of $\tilde{Z}$ to obtain $\tilde{c_i}$
10:     $\tilde{C} = \tilde{C} \cup \tilde{c_i}$
11: end for
```

model, which is trained to reconstruct the social botnet communities more accurately (Lines 7-8, Algorithm 5.3). The *Deep Autoencoder* consists of two phases, namely encoding $f(Z) : \mathbb{R}^{U \times V} \rightarrow \mathbb{R}^{U \times D}$ and decoding $g(Y) : \mathbb{R}^{U \times D} \rightarrow \mathbb{R}^{U \times V}$. In the encoding phase, the aggregated matrix Z is mapped to D-dimensional hidden layers to obtain Y, computed as

$$Y = f(Z) = sig(W_{h2} \cdot sig(W_{h1}Z + b_{h1}) + b_{h2}) \tag{5.13}$$

where W_{h1}, W_{h2}, b_{h1} and b_{h2} are respectively the weights and biases of the encoding phase in the deep autoencoder. Here, $sig(\sigma) = \frac{1}{1+e^{-\sigma}}$, where σ is an argument, represents a mapping function for both encoding and decoding phases. In the decoding phase, Y is mapped to obtain the reconstructed aggregated matrix $\tilde{Z}$ as follows:

$$\tilde{Z} = g(Y) = sig(W_{h4} \cdot sig(W_{h3}Y + b_{h3}) + b_{h4}) \tag{5.14}$$

where W_{h3}, W_{h4}, b_{h3} and b_{h4} are respectively the weights and biases of the decoding phase in the deep autoencoder. Thus, the deep autoencoder model is defined as

$$\tilde{Z} = g(f(Z)) \tag{5.15}$$

Line 9 of Algorithm 5.3 implements the hierarchical clustering [221] on the rows of the trained autoencoder model $\tilde{Z}$ to obtain the reconstructed community structure $\tilde{c_i}$. The entire process is repeated for each community in order to obtain the reconstructed community set $\tilde{C} = \{\tilde{c_1}, ..., \tilde{c_m}\}$ for accurate social bonet community detection.

5.3 Spam Influential Users and Influential Community Detection

Bots may influence some legitimate participants by frequently interacting with them. Thus, identifying the influential participants (which are influenced by social spam bots) from *Twitter* network helps to minimize influence of spreading spam content. In this section, spam influence minimization model is presented in order to identify the influential participant set in *Twitter* network.

Given a *Twitter* network $G = (P', E)$ with influence propagation model I_p, where G holds the following properties:

$$\begin{cases} P' = P \cup B \\ P \cap B = \phi \\ \exists \text{ attack edge } e'_{ij} \in E, \text{if } p_i \in P \wedge p_j \in B \\ \exists \text{ non-attack edge } e_{ij} \in E, \text{if } p_i, p_j \in P \text{ or } B \end{cases} \tag{5.16}$$

where P and B represents set of legitimate participants and social spam bots, respectively.

5.3.1 Spam Influence Minimization

The aim of spam influence minimization model is to find a influential set I with k-participants in order to minimize influence of spam content. Moreover, the spam influence minimization model captures the amount of spam propagation in the presence of bot. The probability of a participant p_i influencing any of its outgoing neighbors along a path z_k is defined as

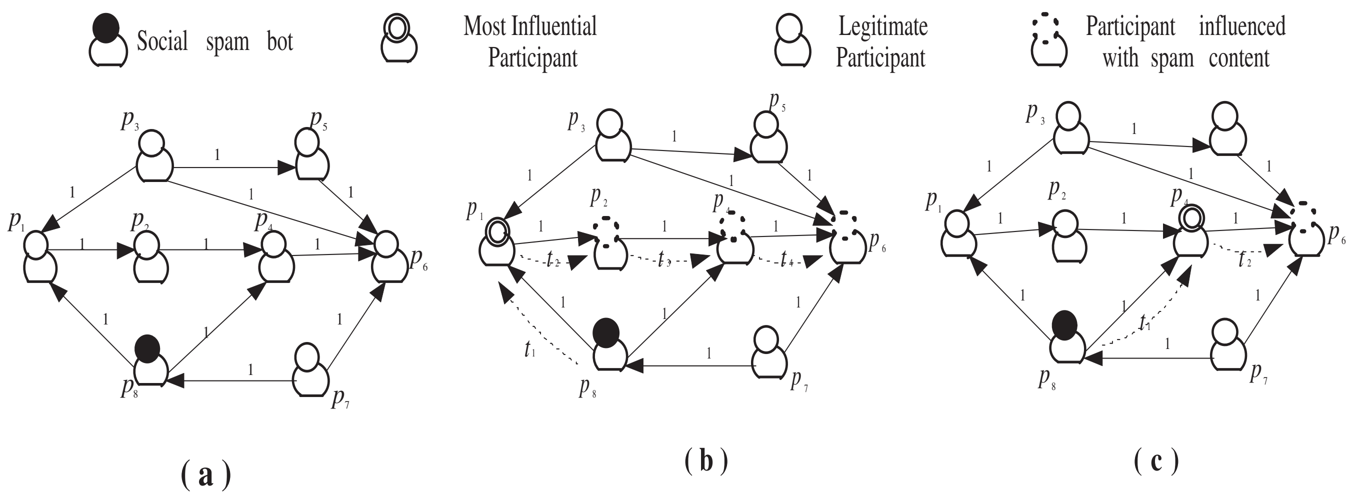


Figure 5.2: An example to illustrate spam influence minimization model

(in [222])

$$\vartheta(z_k) = 1 - \prod_{(p_i,p_j)\in succeed(z_k)} (1 - pr_{ij}) \tag{5.17}$$

where pr_{ij} represents the interaction probability from i^{th} participant to j^{th} participant. In a network, the probability of a participant p_i influencing any of its outgoing neighbors along multi-paths $Z(p_i) = \{z_1, z_2, ..., z_k, ..\}$ (in the presence of bot b) (i.e., which is denoted as $Pr(b, p_i)$)) is defined as

$$Pr(b, p_i) = 1 - \prod_{z_k\in Z(p_i)} (1 - \vartheta(z_k)) \tag{5.18}$$

Fig. 5.2 shows the method of identifying most influential participant by considering one participant as social spam bot. Fig. 5.2(a) shows a *Twitter* network with one participant as social spam bot (i.e., p_8). The most influential participant can be identified either: (i) by choosing the participant p_1 as most influential participant (as shown in Fig. 5.2(b)) or (ii) by choosing the participant p_4 as most influential participant (as shown in Fig. 5.2(c)). In the first case, when p_1 is chosen as most influential participant. At different time slots, p_1 may influence p_2 with probability value 0.2 and p_2 spreads the information to p_4. Finally, p_4 spreads to p_6. Therefore, the probability of participant p_1 influencing any of its outgoing neighbors along a path (i.e., $p_1 \rightarrow p_2 \rightarrow p_4 \rightarrow p_6$) $Pr(p_8, p_1) = 0.488$. In the second

Algorithm 5.4 Spam_Influential_Users (SIU) Detection

```
Input:
    $G = (P', E), B$- set of social spam bots and $NB^{out}(b)$- outgoing neighbors of spam bot $b$
Output:
    I, set of influential participants
1: $I = \phi$; Visited={}; the set $D_p = \phi$
2: for each social spam bot $b \in B$ do
3:     for each $p_i \in NB^{out}(b)$ do
4:         Compute $Pr(b, p_i)$ using Equation (5.18)
5:         $D_p = D_p$.append $Pr(b, p_i)$
6:     end for
7:     $k^* = \max_{p_i \in NB^{out}(b)} D_p$
8:     $I = I \cup k^*$
9: end for
10: return $I$
11: Influential_Community_Detection()
```

case, when p_4 is chosen as influential participant. At time slot t_1, p_8 may influence p_4 with probability value 0.2. Similarly, at time slot t_2, p_4 may influence p_6 with probability value 0.2. Therefore, the participant p_4 influencing any of its outgoing neighbors along a path (i.e., $p_4 \rightarrow p_6$) $Pr(p_8, p_4) = 0.2$. Because of p_1, more participants (i.e., p_2, p_4 and p_6) are influenced by spam content. Hence, it is more likely to choose participant p_1 as most influential participant instead of choosing participant p_4 for spam influence minimization problem.

A *Spam Influential Users (SIU)* detection algorithm (Algorithm 5.4) has been proposed to determine influential participants and to minimize spam influence in Twitter network. In *SIU* algorithm, firstly an empty influential participant set is initialized (i.e., $I = \phi$). For each social spam bot $b \in B$, the participant k^* (who has the maximum dissemination probability) will be added into influential participant set I. Further, Algorithm 5.5 presents influential community detection (ICD) in *Twitter* network. For all participants which are not assigned to a influential community c_i, the influence value of each participant p_i will be determined using Equation (5.18). Later, all one-hop neighbors of p_i (whose influence value with p_i is greater than threshold value T_f) will be identified. If neighbors are not assigned to an influential community c_i then p_i and its neighbors belong to same influential

Algorithm 5.5 Influential_Community_Detection (ICD)

```
    Input:
        $G = (P', E), B, I_p$
    Output:
        $C = \{c_1, c_2, ....c_m\}$ a set of influential communities
 1: Visited ={}
 2: for each participant $p_i \in P'$ do
 3:     if $p_i \notin$ Visited then
 4:         C={}; temp={$p_i$}
 5:         $c_i \leftarrow \{p_i\}$ and $C = C \cup c_i$
 6:         Visited = Visited $\cup \{p_i\}$
 7:         while temp $\neq \phi$ do
 8:             $p \leftarrow$ Randomly select a participant from temp
 9:             Compute influence of $p$ and its neighbors using Equation (5.18)
10:             $N \leftarrow$ Find the neighbors of $p$ whose influence value is greater than thresh-
    old value $T_f$
11:             $N \leftarrow$ Delete the participants from $N$ which belong to Visited
12:             $temp \leftarrow$ Add the participants in $N$ to temp
13:             $c_i = c_i \cup N$
14:             Visited = Visited $\cup$ N
15:         end while
16:     end if
17: end for
18: $C = \{c_1, c_2, ....\}$
19: repeat
20: for each community $c_i \in C$ do
21:     for each participant $p_i \in c_i$ do
22:         NC=Find the neighbors of $p_i$ which doesn't belong to $c_i$
23:         for j=1 to $|NC|$ do
24:             $cc(p_i, NC_j) \leftarrow$ Compute closeness between $p_i$ and $NC_j$
25:             if $cc(p_i, NC_j) < \varsigma$ then
26:                 Delete the edge between $p_i$ and $NC_j$
27:             else
28:                 $c_i = c_i \cup NC_j$
29:             end if
30:         end for
31:     end for
32: end for
33: until influential communities are almost identical for at least two successive iterations
```

community c_i. Moreover, once the initial influential community c_i (which is associated with p_i) is detected, then the *ICD* algorithm starts to identify the influential communities which are associated with other participants (i.e., which are not assigned to c_i). The influential community c_i is extended by considering the neighbors of p_i (where $p_i \in c_i$) which doesn't belong to c_i (and it is denoted as NC). If closeness centrality between p_i and NC_j (i.e., denoted $cc(p, NC_j)$) is greater than threshold value ς, then the participant NC_j is added to c_i. Otherwise, if $cc(p, NC_j)$ is less than ς, then delete the edge between p_i and NC_j from G. The algorithm runs till the influential communities are identical (for at least two successive iterations).

5.4 Performance Evaluation

In this section, the performance of the proposed SBCD and DA-SBCD algorithms are evaluated for detecting social botnet communities, and compare them with two recent methods, namely detecting spam communities (*SpamCom*) [36] and *Botnet Discovery* [37]. The SpamCom identifies spammers (or social spam bots) based on the user behavioral features and applies clique to determine strongly connected botnet communities. On the other hand, the Botnet Discovery identifies bots by considering correlation graph and applies modularity based clustering approach for community detection. Further, the proposed spam influential users and influential community detection (*SIU-ICD*) algorithm is compared with two existing algorithms, such as opinion spammer community detection (*OSCD*) [40] algorithm and spammer group detection (*SGD*) [41]. Two datasets, such as *The Fake Project* dataset and *Social Honeypot* dataset are considered for performance evaluation. The proposed SBCD and DA-SBCD algorithms consider two parameters namely similarity value (Section 5.2.2) and trust value (Section 5.2.3) to detect botnet communities.The performance of the proposed algorithms are evaluated in terms of normalized mutual information (NMI), precision, recall, F-measure G-measure and modularity. These metrics are defined as follows:

- *Normalized Mutual Information (NMI)*: There exist two different types of communities, namely detected communities $\tilde{C}$ and ground-truth labeled communities $\vec{C}$. Let q_{ij} represent the number of participants in a detected community $\tilde{c}_i$ with label j. Let q_i and q_j represent the number of participants in each detected community $\tilde{c}_i$ and with labeled data j for a ground-truth labeled community $\vec{c_j}$, respectively. NMI is defined as follows:

$$NMI(\tilde{C},\vec{C}) = \frac{\sum_{\tilde{c_i}\in\tilde{C}}\sum_{\vec{c_j}\in\vec{C}} q_{ij} log \frac{n.q_{ij}}{q_i.q_j}}{\sqrt{(\sum_{\tilde{c_i}\in\tilde{C}} q_i log \frac{q_i}{n})(\sum_{\vec{c_j}\in\vec{C}} q_j log \frac{q_j}{n})}}$$

where n represents the total number of participants. If *NMI* is close to zero, it implies that there exists dissimilarity between the detected and ground-truth labeled communities. If *NMI* is close to one, it implies that there exists high similarity between the detected and ground-truth labeled communities.

- *Modularity*: Modularity $\tilde{Q}$ is a metric which evaluates the quality of detected communities, especially when the ground-truth is unavailable. Modularity is defined as $\tilde{Q} = \frac{1}{2k}\sum_{i,j}(A_{ij} - \frac{deg_i deg_j}{2k})\delta(i,j)$ (in [223]), where k is the number of edges, deg_i and deg_j represent the degree of participant p_i and p_j , respectively and A_{ij} is the element of adjacency matrix. The term $\delta(i,j) = 1$ represents that participant p_i and p_j belong to same group. Otherwise, $\delta(i,j) = 0$. Modularity $\tilde{Q}$ value lies between -1 and 1. Higher modularity $\tilde{Q}$ value represents a best partition of communities in network.

- *Precision* (P): It is defined as $P = \frac{TP}{TP+FP}$, where TP (respectively, FP) represents similar (respectively, dissimilar) type of participants assigned to the same community.

- *Recall* (R): It is defined as $R = \frac{TP}{TP+FN}$, where FN represents similar type of participants that are assigned to different communities.

- *F-measure*: F-measure is defined as $\frac{2\times Precision\times Recall}{Precision+Recall}$

- *G-measure*: G-measure is defined as $\sqrt{Precision \times Recall}$

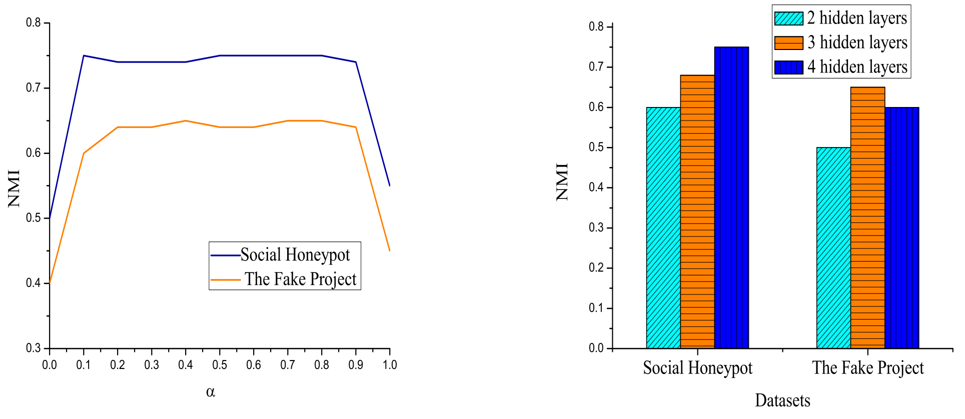
5.4.1 Experimental Results for Social Botnet Community Detection

The proposed DA-SBCD algorithm considers both similarity and trust values which are given as input matrix $[S,T]^T$ to the deep autoencoder model. A parameter α is considered, where $0 < \alpha < 1$, in order to balance the proportion of trust T and similarity values S, i.e., $[(1-\alpha).S, \alpha.T]^T$. Fig. 5.3(a) shows the performance of DA-SBCD algorithm in terms of NMI by varying α values from 0 to 1 on the Social Honeypot and The Fake Project datasets. It is observed that when α is either 0 or 1, the DA-SBCD algorithm achieves low NMI value. However, when $\alpha \in (0, 1)$ in the open interval, the DA-SBCD algorithm provides better NMI value. The is because by considering the trust value of a participant and behavioral similarity between participants, better performance is achieved for social botnet community detection. Fig. 5.3(a) demonstrates that by varying α between 0.1 and 0.9, the NMI values slightly fluctuate because the deep autoencoder adjusts α for better performance.

Fig. 5.3(b) shows the performance of DA-SBCD algorithm for detecting social botnet communities with two, three, and four hidden layers on the two datasets. For *Social Honeypot* dataset, it has been observed that DA-SBCD algorithm with four hidden layers achieves better NMI value when compared to the performance with three hidden layers. This implies that the number of hidden layers in the deep autoencoder plays a vital role. For *The Fake Project* dataset, the NMI value of the proposed DA-SBCD algorithm with four hidden layers has been drastically reduced when compared to the performance with three hidden layers. However, for The Fake Project dataset, the DA-SBCD algorithm with four hidden layers achieves moderate accuracy (in terms of NMI) as compared to the cases with two hidden layers and three hidden layers. This is because, when more hidden layers are considered for low dimensionality of data, the original data will be reduced, leading to lower performance.

Table 5.2: Evaluation metrics of legitimate and other type of social bot attacks for two Twitter datasets

Dataset	**Community**	**P**	**R**	**F**
The Fake Project	Legitimate	89.21	85.58	87.35
	Spam bots	91.16	84.42	87.66
	Fake followers	92.12	87.65	89.82
Social Honeypot	Legitimate	92.16	84.26	88.03
	Content polluters	91.25	87.14	89.14

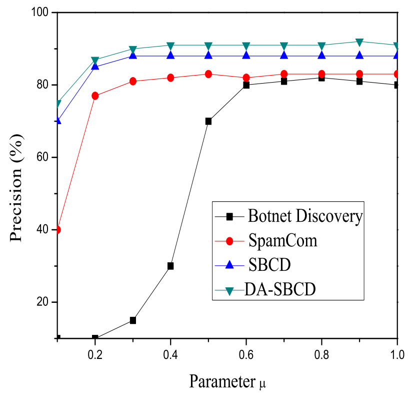


(a) Varying the parameter α (b) Varying the number of hidden layers

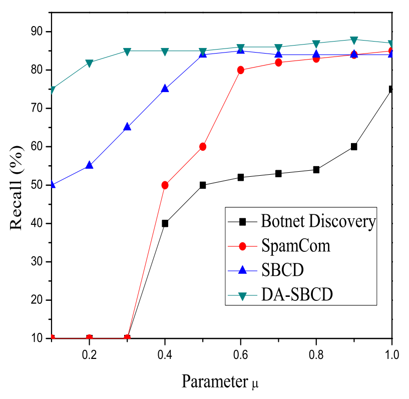
Figure 5.3: Performance of *DA-SBCD* algorithm on two Twitter datasets

Table 5.2 shows the performance of determining overlapping communities using evaluation metrics, such as recall (R), precision (P) and F-measure (F) on two Twitter datasets for different types of communities. For *The Fake Project dataset*, the DA-SBCD algorithm achieves an average precision of $P = 90.86\%$ for classifying legitimate users and other types of social botnet communities. The obtained F-measure values are $F = 87.35\%, 87.66\%$, and 89.82% for legitimate, spam bots, and fake followers communities, respectively.

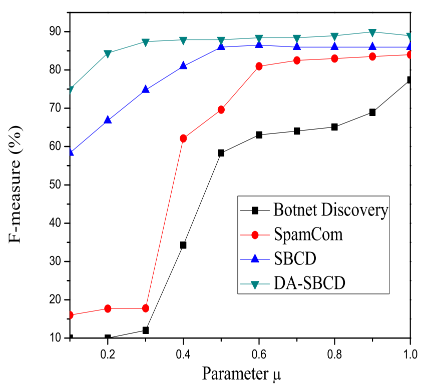
Figures 5.4 and 5.5 compare the performance of the proposed DA-SBCD and SBCD algorithms with existing methods SpamCom and Botnet Discovery in terms of precision, recall, F-measure and G-measure on the two Twitter datasets with a parameter μ ranging from 0 to 1 (where the parameter μ is defined as the ratio of the number of attack edges that a social bot can create at a particular time slot and the total number of attack edges that exist in the network). It has been observed that DA-SBCD outperforms those two existing algorithms on the Twitter datasets. Specifically, for $\mu > 0.3$, DA-SBCD achieves better precision value when compared to SpamCom and Botnet Discovery methods. As μ increases, the performance in terms of precision, recall, F-measure and G-measure of all social botnet community detection algorithms also increase, due to the fact that the social bots attempt to establish social relationships with the legitimate participants. Moreover, the DA-SBCD algorithm provides 2-4% improvement on precision over the SBCD algorithm.



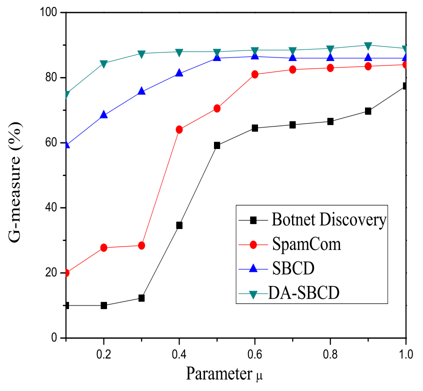
(a) Precision



(b) Recall



(c) F-measure

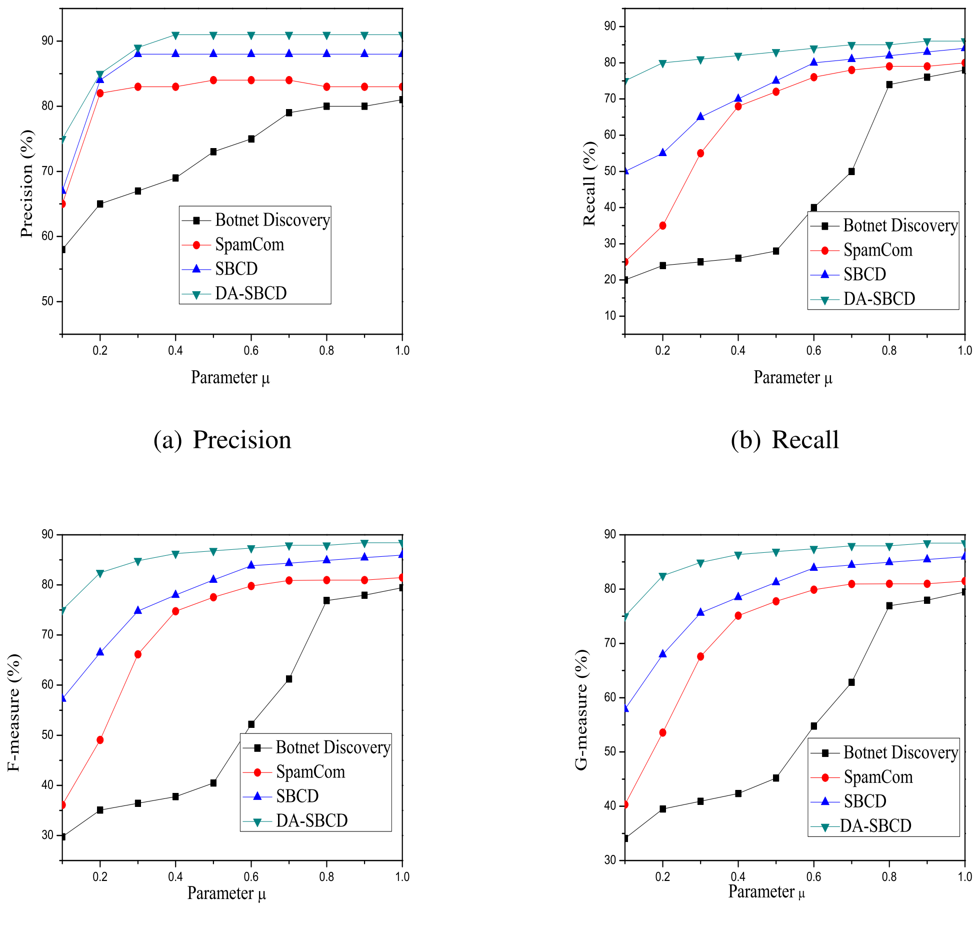


(d) G-measure

Figure 5.4: Comparison of *DA-SBCD* algorithm with other botnet communities detection algorithms on *The Fake Project* dataset

The is because DA-SBCD identifies different types of malicious activities using deep autoencoder with random-walk based trust model and similarity among participants. For *Social Honeypot* dataset, the performance of DA-SBCD algorithm provides an improvement of about 6% on precision, 3% on recall, and 3% on F-measure over the SpamCom algorithm . Therefore, on an average, the performance of the proposed DA-SBCD algorithm is improved around 3% over the existing algorithms. Indeed, the F-measure result demonstrates that the DA-SBCD outperforms other existing methods (such as SpamCom and Botnet Discovery) for social botnet community detection.

Figures 5.6(a) and 5.6(b) compare the DA-SBCD algorithm with the SBCD algorithm,

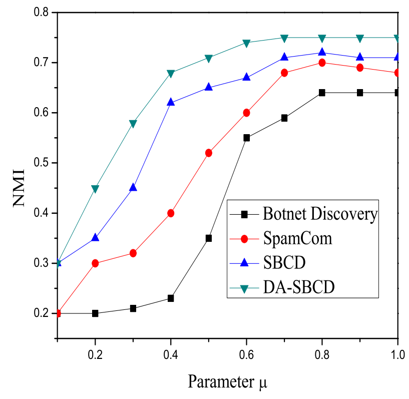


(a) Precision (b) Recall

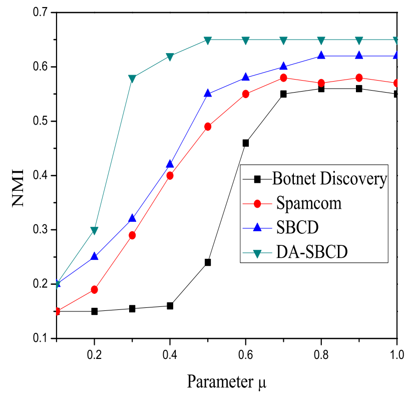
(c) F-meaure (d) G-meaure

Figure 5.5: Comparison of *DA-SBCD* algorithm with other botnet communities detection algorithms on *Social Honeypot* dataset

SpamCom and Botnet Discovery in terms of NMI for different values of μ varying from 0 to 1. It has been observed that DA-SBCD outperforms other existing social botnet detection algorithms on both the Twitter datasets considered. Precisely, DA-SBCD provides 4-8% improvement on the NMI values. For Social Honeypot dataset, DA-SBCD achieved the highest $NMI = 0.75$ when $\mu = 0.7$, which is a significant improvement over Botnet Discovery. Although SpamCom achieves better performance when compared to Botnet Discovery, it is lower than DA-SBCD algorithm. One reason for this is that the existing social botnet community detection algorithms learn only from a single layered representation of data. However, the proposed DA-SBCD algorithm learns from multiple deep layers for



(a) Social Honeypot



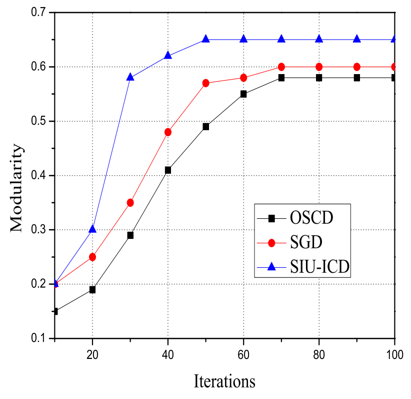
(b) The Fake Project

Figure 5.6: Performance of *DA-SBCD* algorithm on two Twitter datasets in terms of NMI

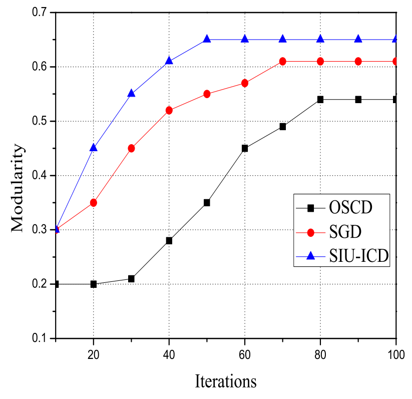
social botnet community detection. Another reason is thatDA-SBCD integrates both trust and similarity values, and learns by adjusting the parameter α for better performance in different types of datasets. Moreover, in the deep autoencoder model, the encoding phase of hidden layer considers only a set of participants having high behavioral similarity edges, instead of considering all participants. This implies that by integrating the deep autoencoder model with trust and similarity values, the botnet detection approach is more accurate.

5.4.2 Experimental Results for Spam Influential Communities

Fig. 5.7(a) and 5.7(b) show comparison of proposed spam influential users and influential community detection (*SIU-ICD*) algorithm with two existing algorithms, such as opinion spammer community detection (*OSCD*) [40] algorithm and spammer group detection (*SGD*) [41] in terms of modularity $\tilde{Q}$ over 100 iterations. From Fig. 5.7, it can be observed that the proposed *SIU-ICD* algorithm holds good community detection performance in terms modularity $\tilde{Q}$ after 50 iterations. Moreover, the proposed *SIU-ICD* algorithm achieves better among other existing spammer community detection algorithms on two *Twitter* datasets and it obtains the highest modularity $\tilde{Q}$ value on the two datasets. The reason is that in the datasets with non-overlapping community structures, the existing community detection algorithms can accurately detect communities with good modularity



(a) Social Honeypot dataset



(b) The Fake Project dataset

Figure 5.7: Modularity of SIU-ICD and other existing algorithms (OSCD and SGD)

value. The proposed *SIU-ICD* algorithm is not much significant for such kind of datasets. The datasets with dynamic community structures (i.e., where user behavior changes over a period of time) having spammer behavior, the proposed *SIU-ICD* algorithm identifies the most influential users and communities based on their malicious behavior (and which are influenced by spam bots).

From Fig. 5.7(b), it can be observed that for *The Fake Project* dataset, the highest modularity $\tilde{Q}$ value obtained by *SIU-ICD* is 0.65. This is due to fact that *SIU-ICD* algorithm considers an influence propagation model where the number of users in the influential neighboring participant set increase iteratively. This makes the proposed algorithm more stable during influential community detection phase. The proposed *SIU-ICD* algorithm achieves 4-9% improvement on modularity $\tilde{Q}$ over existing spammer community detection algorithms. This is due to the fact that the *SIU-ICD* is able to identify the dynamic behavioral changes of spammer community detection. However, identifying the most influential participants (which have higher out-degree) play a significant role in influencing the participants within a community structure. Thus, spreading spam content to such influential participants may have more (negative) impact on such influential community structure. Therefore, identifying such spam influential community structure helps to minimize spam influence in *Twitter* network.

5.5 Summary

This chapter analyzes the behavioral similarity of the participants by considering four different aspects, such as tweet-content similarity, shared URL similarity, interest similarity and social interaction similarity for identifying similar type of behavior (malicious or non-malicious) among participants in the Twitter network. Based on a deep autoencoder model, the proposed algorithm detects social botnet communities with improved precision and recall. Further, an *Influential Community Detection* algorithm has been proposed and this helps in reducing the spread of spam-content through influential communities in *Twitter* network. In the next chapter, a social trust model has been presented with learning automata in order to evaluate trustworthy paths in online social networks for trusted-user recommendations.

Chapter 6

Learning Automata-based Trust Model for Determining Trustworthy Paths in Online Social Networks

Online social networking websites attract a millions of users and provide variety of social services by interacting with participants [224]. In a social network structure, each node is identified as participant and each edge corresponds to the relationship between the participants. Each participant can interact with other participants directly or indirectly [225]. In a service-oriented system, trust plays a major role for selective decision making and requires a methodology to evaluate the trust paths between the participants who are unknown to each other. A service provider may choose trust path selection criteria, such as releasing an upgraded product and interviewing employees for evaluating the trustworthy services for the consumers. The trust value specified between two participants is based on their recommendations and the quality of the products [94]. Moreover, a trustworthy service is based on the social relationships and recommended influence value among the participants. Therefore, finding a trustworthy path by selecting trust parameters is a challenging task [83].

A recommendation based online social network is shown in the Fig. 6.1. Participants A and I are indirectly linked by multiple paths. A is a service provider and I is a consumer (participant) to evaluate a trust path based on direct [226] and indirect trust values

[94] (from A to I). In case of *Big Data* networks [227], multiple recommended trust paths exist between a service provider and the consumers [42]. Moreover, evaluating the trustworthy services for the consumers by considering multiple social paths is a critical issue and time consuming process. Further, finding a best recommended trust path is a challenging problem in social networks. However, for identifying the social trust path, the shortest path based approaches are used [44]. For good recommendations, to purchase services, consumers need to focus mainly on the non-functional requirements, such as service cost, service availability and service delivery time (response time) along with the consumer feedbacks [228]. For accurate recommendations, social network criteria, such as proximity, realization, chunk and betweenness (as depicted in Fig. 6.1) are considered including non-functional requirements (refer Section 6.3).

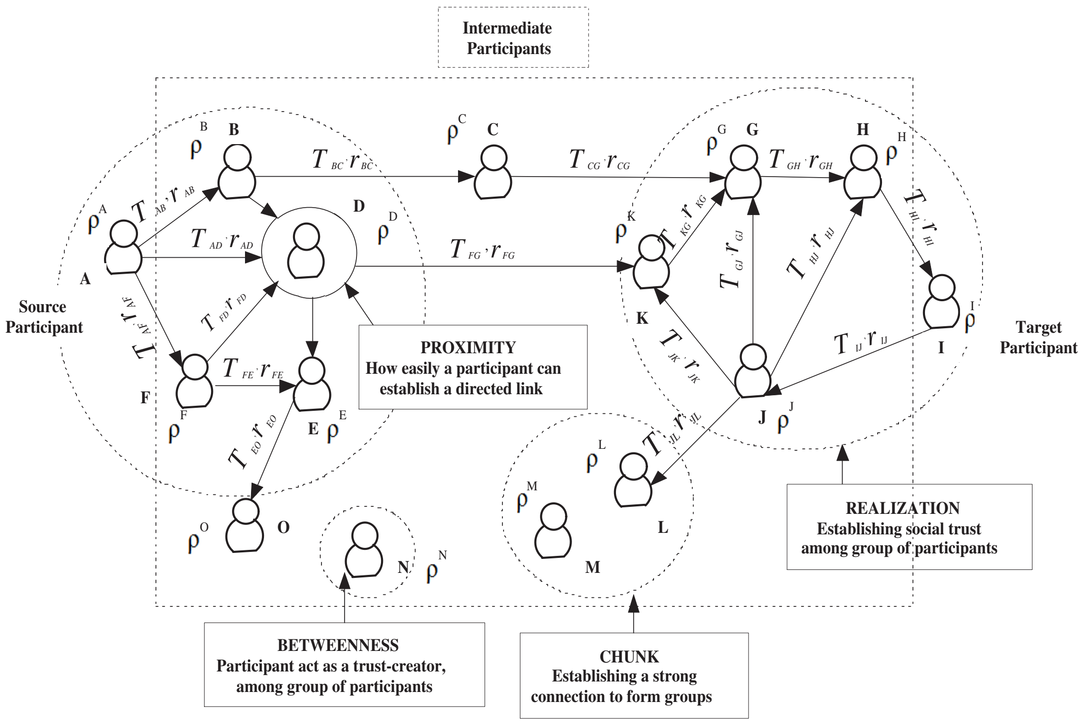


Figure 6.1: A social network structure for finding recommended trust path between a source participant A and a target participant I

A participant (i.e., consumer) may select a recommended trust path with highest trust value. However, the participant may dislike a service as its cost may go beyond the participant's budget or some attributes may not fulfill the participant's preferences. Although, the requirements fulfill the participant's preferences, the services may not be selected by a participant if it is not a recommended trust path [229]. The following are the *Big Data* challenging issues that the existing algorithms fail to address: (i) The potential growth of large volume of non-functional requirements will increase the complexity of social network for selective decision making, (ii) Due to the uncertainty of trust information in a network, evaluating trustworthy services leads to variability in non-functional requirements, (iii) Heterogeneous qualitative or quantitative participant preferences and trust information lead to a variety of non-functional requirements in online social networks [230]. Therefore, a *Big Data* model is developed for finding a recommended trust path to evaluate trustworthy services based on the social trust information.

To address the above challenging issues in online social networks, a novel approach has been proposed for finding a recommended trust path by considering direct trust, indirect trust, social relationships and recommendations. The major contributions are summarized as follows:

- Develop a *High quality of Social trust (HoS)* constrained model for evaluating trustworthy services in online social networks by incorporating attributes, such as trust information (direct trust and indirect trust), social relationships and the participants' recommendations.

- Design a Learning Automata based-Recommended Trust Path Selection (LA-RTPS) algorithm, where multiple recommended trust paths are identified from a source participant to a target participant.

- Experiments are conducted on two real online social network datasets, such as *Slashdot* dataset [231] and *Epinions* dataset [232] to evaluate the efficacy of the proposed *LA-RTPS* algorithm.

This chapter is organized as follows: Section 6.1 presents the motivation. Section 6.2 discusses the basic definitions and overview of learning automata. In Section 6.3, a multiple *HoS* model has been designed for evaluating trustworthy services in online social networks and a *Learning Automata based-Recommended Trust Path Selection* (LA-RTPS) algorithm has been presented. Section 6.4 presents the experimental results based on two datasets to evaluate the performance of the proposed algorithm. Finally, the summary of this work is presented in Section 6.5.

6.1 Motivation

In this era of *Big Data*, it is demanding to extract trust information and finding trustworthy participants in online social networks. The conventional approaches (like content-based recommendation models) consider social relationships based on comments provided in online social networks [233]. Moreover, these types of approaches are not taken into consideration for the establishment of social relationship among multi-hop participants. This has motivated us to propose and design a recommendation based multi-hop trust management system for finding recommended trust paths. Moreover, an attacker may act unethically and gets good reputation. Once an attacker gets high trust value then the attacker may provide untrustworthy recommendations. This motivated us to integrate social trust information along with a good recommendation for finding a best recommended social trust path. In real-time, certain service providers select a few malicious participants to provide faulty decisions to the services (i.e., trust formation and recommendations) of the other participants. Moreover, the service providers provide significant ranking to their own services for selective decision making. Therefore, the social trust information should be taken into consideration for avoiding such malicious comments posted in online social networks. In this work, a *Learning Automata based Recommended Trust Path Selection (LA-RTPS)* algorithm has been proposed where multiple recommended trust paths are identified from a source participant to a target participant. LA-RTPS algorithm aims to overcome the limitations of an existing heuristic based optimal social trust path selection approach [44].

6.2 Basic Definitions and Learning Automata

In this section, some basic terminologies are defined. Later, an overview of learning automata is presented.

6.2.1 Definitions

In complex social networks, trust plays a major role for users' recommendations and social relationships. Many authors have proposed trust definition in different scenarios [83], [96]. In this contribution, the parameters are defined as follows:

Definition 1 (Trust): Trust is a belief on a specific service performed by one participant. Trust is assigned to others based on recommendations and relationships. Let $T_{P_iP_j} \in [0,1]$ represents the trust value between two participants P_i and P_j. If $T_{P_iP_j} = 1$, it indicates that P_i fully trust P_j. Further, if $T_{P_iP_j} = 0$ then it indicates that P_i (fully) distrusts P_j.

Definition 2 (Relevance degree): Relevance degree between two participants P_i and P_j is denoted as $r_{P_iP_j} \in [0,1]$. If $r_{P_iP_j} = 0$, it indicates that no relationship exists between two participants. If $r_{P_iP_j} = 1$, it indicates that there exists a strong social relationship between two participants.

Definition 3 (Recommended influence value): Recommended influence value of a participant P_i is based on recommendations on social trust path. The recommended influence value is denoted as $\rho_{P_i} \in [0,1]$. If $\rho_{P_i} = 1$, it implies that participant P_i will prefer the service. If $\rho_{P_i} = 0$, it implies that participant P_i has no information regarding the service.

Definition 4 (High quality of Social trust): High quality of Social trust (HoS) is the ability of providing a trustworthy service in social trust propagation by considering trust (T), relevance degree (r) and recommended influence value (ρ) as a set of attributes.

Definition 5 (Aggregate Path (AP)): In an online social subnetwork, there exists multiple paths from a source participant v_i to a target participant v_j through an intermediate node v_k. For each path, *HoS* attribute values (refer Section 6.3.2 and Section 6.3.3) are to be computed. An *Aggregate Path* is a social path from v_i to v_j through an intermediate node v_k (i.e., path $AP^{U}_{v_i \to v_j}$) with maximal utility value U (which is determined using Eq.(14)).

In service-oriented computing, participants can define multiple constraints for a given set of *Quality of Service* (QoS) attributes to satisfy non-functional requirements, such as service cost, service availability and service delivery time (response time) [230]. Different non-functional requirements have different constraints for *HoS* attributes.

6.2.2 Learning Automata

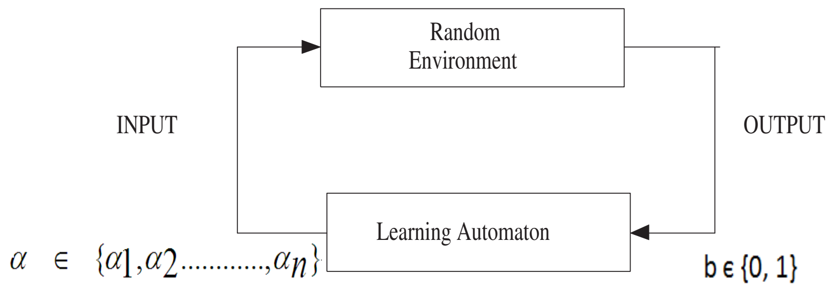


Figure 6.2: Learning Automaton

Fig. 6.2 shows the relationship between learning automaton and a random environment. A learning automata selects the required action from a finite set of feasible actions through repetitive process. The actions are executed on random environment and produces the responses in terms of either a reward or a penalty. Moreover, each input is associated with a specific action and learning automaton updates its action probability value by taking the learning algorithm into consideration [184].

Learning Automaton is defined as a quadruple $< I, \alpha, p_d, R >$, where $I = \{I_1, I_2, ..., I_n\}$ represents learning automaton's input set and $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ represents a finite set of actions such that each automata selects one of the trustworthy participant. Further, $p = \{p_1, p_2, ...p_n\}$ represents the action probability set where p_i denotes the probability of selecting a specific action α_i. The term *R* represents the learning algorithm which updates the reinforcement signal (represents success or failure of a system after performing a finite set of actions) based on the random environment responses. In the random environment, the learning automaton is denoted as $< a, b, c >$, where $a = \{a_1, a_2, ..., a_n\}$ represents the finite input set, $b = \{b_1, b_2, ..., b_n\}$ represents reinforcement signal values and $c = \{c_1, c_2, ..., c_n\}$ represents penalty probability set (where c_i is the corresponding value with each $a_i, 1 \leq i \leq n$). In the proposed algorithm, a reward value is considered. If the learning automaton gets a reward from the random environment then the action probability value *p* is updated. Otherwise, the *p* value remains same. Let $\alpha(m)$ be a specific action chosen by the learning automata at an instant *m*. The action probability value is given by

$$p_j(m+1) = \begin{cases} (1-\varphi)p_j(m), \forall j \neq i, b = 0 \\ p_j(m) + \varphi(1 - p_j(m)), \forall j = i, b = 0 \\ p_j, b = 1 \end{cases} \tag{6.1}$$

where φ is a constant.

6.3 Learning Automata based High Quality of Social Trust Constrained Model

In this section, firstly, a recommendation-based online social network architecture has been designed for analyzing the recommended trust paths based on learning automata. In addition, a *High quality of Social trust (HoS)* model is presented for establishing a strong social connection among a group of participants. *Shannon's* entropy approach is used to compute utility value for each trustworthy service. Further, a *Learning Automata based Recommended Trust Path Selection* (LA-RTPS) algorithm has been proposed to identify multiple recommended trust paths from a given source participant to a target participant.

6.3.1 A Recommendation-based Online Social Network Architecture

As shown in Fig. 6.3, the proposed recommendation-based online social network architecture have the following five measures: (a) Service providers publish their services, such as *social relationships*, *trust value* and *recommended influence value* (which are collected and stored in a service registry), (b) Social trust (influence) among the participants is determined based on proximity, realization, chunk and betweenness. A participant can establish a directed link with others based on proximity (which is determined using a geo-social based distance measure [234]). In real world, a simple way for a participant is to establish a social connection in order to have mutual friends. A participant creates trusted links among a group of participants [235]. The realization and chunk are the two other parameters (in an online social networking structure) which dynamically influence the social trust composition. These two measures are considered as basic idea for establishing a strong so-

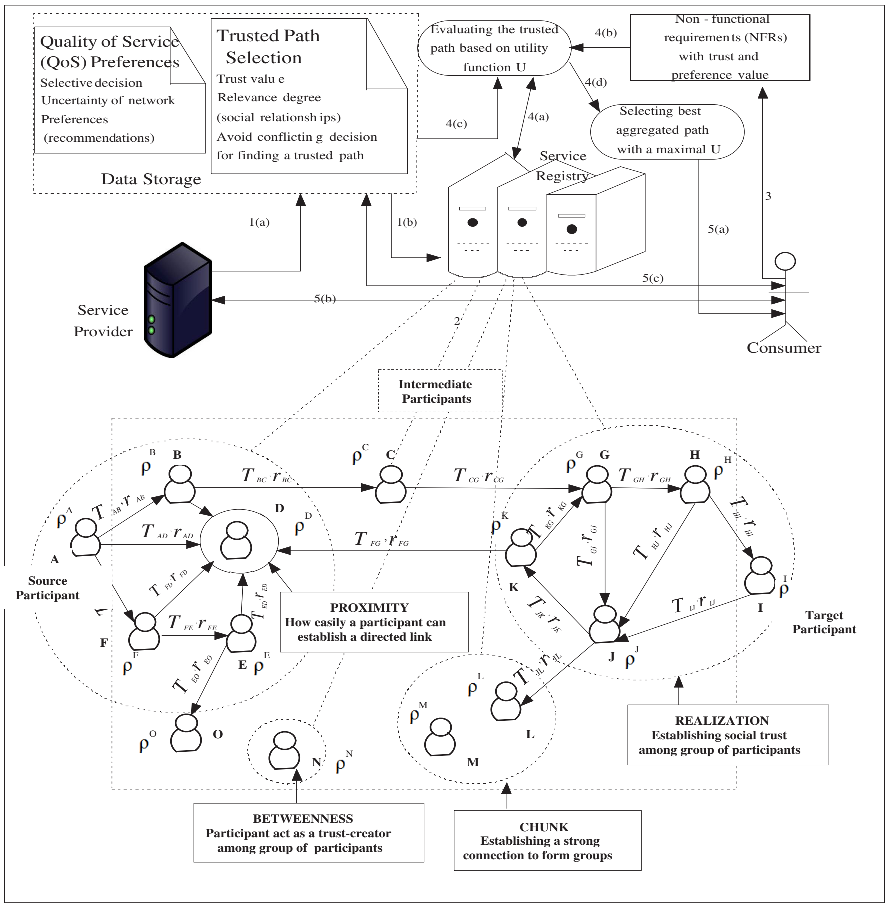

Quality of Service (QoS) Preferences
Selective decision
Uncertainty of network
Preferences
(recommendations)
Trusted Path Selection
Trust value
Relevance degree
(social relationships)
Avoid conflicting decision
for finding a trusted path
Data Storage
Evaluating the trusted path based on utility function U
Non - functional requirements (NFRs) with trust and preference value
Selecting best aggregated path with a maximal U
Service Registry
Service Provider
Consumer
Intermediate Participants
Source Participant
Target Participant
PROXIMITY
How easily a participant can establish a directed link
REALIZATION
Establishing social trust among group of participants
BETWEENNESS
Participant act as a trust-creator among group of participants
CHUNK
Establishing a strong connection to form groups
1(a)
1(b)
2
3
4(a)
4(b)
4(c)
4(d)
5(a)
5(b)
5(c)

Figure 6.3: A recommendation based online social network architecture for finding recommended trust paths

cial connection within the group of participants, (c) When a consumer request for a service, user specifies all his non-functional requirements, such as cost, availability and response time along with the users' feedback and trust value, (d) For a recommended trust path, all services stored in a service registry are collected based on the consumer non-functional requirements. Later, a recommended trust path is evaluated based on utility value 'U' and the best aggregate path is selected with a maximal utility value *'U'*, (e) Finally, after the consumer invokes the recommendations about a service, its feedback will be stored in a service registry for social trust information (as a recommended influence value).

A participant's trust value is evaluated based on either current direct relationships or previous direct relationships. If a participant's trust value is based on recommendations, then it is termed as indirect trust.

6.3.2 HoS Attribute Association

Big Data challenging issues deals with the massive volume and variety of non-functional requirements, such as cost, availability and response time including the consumer feedback and trust. The existing model [44] is customized by considering the parameters, such as direct trust, indirect trust, relevance degree and recommended influence value. In a service oriented system, participants specify constraints for ***HoS*** attributes in order to fulfill the non-functional requirements. As shown in Fig.3, a source participant specifies ***HoS*** constraints (assumed as $H_{AI} = \{T_{AI} > 0.4, r_{AI} > 0.3, \rho_{AI} > 0.4\}$) for the recommended trust path from *A* to ***I***. The ***HoS*** attribute associated model is presented in next section.

6.3.2.1 Trust Association

Trust may be transitive among participants [236]. If there exists *n* participants $\{P_1, P_2, ...P_n\}$ then the recommended trust path is denoted as $T_{P(P_1,P_2,.....P_n)}$. Realization, proximity, chunk and betweenness as the essential characteristics to measure the social trust influence among the group of participants. Trust association model is established between participant P_i and participant P_j at time 't' (which is denoted as $T_{P_iP_j}(t)$) and it is given by

$$T_{P(P_1,P_2,.....P_n)} = \prod_{(P_i,P_j)\in(P(P_1,P_2,.....P_n))} T_{P_iP_j}(t) \tag{6.2}$$

$$T_{P_iP_j}(t) = \sum_{A}^{all} T^{A}_{P_iP_j}(t) \tag{6.3}$$

where 'A' represents a trust attribute which contains realization, proximity, chunk and betweenness. $T^{A}_{P_iP_j}(t)$ denotes the participant $P_i's$ trust (in trust attribute *'A'*) towards the participant P_j at a particular time $(t + \delta t)$ and the term is defined as follows:

$$T^{A}_{P_iP_j}(t + \delta t) = \alpha T^{D,A}_{P_iP_j}(t + \delta t) + (1 - \alpha) T^{ID,A}_{P_iP_j}(t + \delta t) \tag{6.4}$$

where $T^{D,A}_{P_iP_j}(t + \delta t)$ represents a direct trust using direct relationship, $T^{ID,A}_{P_iP_j}(t + \delta t)$ represents indirect trust using recommendations and α lies within a range of 0 and 1. Let $X^{D,A}_{P_i,P_j}(t)$ represents a boolean variable. This implies that the data needed for assessing trust attribute *'A'* is obtained at time δt. The direct trust $T^{D,A}_{P_iP_j}(t+\delta t)$ is represented as follows:

$$T^{D,A}_{P_iP_j}(t+\delta t) = \begin{cases} T^{A}_{P_iP_j}(t+\delta t), & ifX^{D,A}_{P_i,P_j}(t) = true \\ e^{-\theta\delta t} * T^{D,A}_{P_iP_j}(t), & ifX^{D,A}_{P_i,P_j}(t) = false \end{cases} \tag{6.5}$$

The participant P_i will update direct trust $T^{D,A}_{P_iP_j}(t+\delta t)$ towards the participant P_j (for a trust attribute *'A'*) if and only if participant P_i directly interacts with participant P_j at time t and the information required for assessing 'A' is obtained at time δt. Otherwise, the participant simply updates the direct trust $T^{D,A}_{P_iP_j}(t+\delta t)$ with its past interaction $T^{D,A}_{P_iP_j}(t)$ over an exponential time $e^{-\theta\delta t}$ where $0 < \theta < 1$. The following terms are related to social trust measurement for trust evaluation (in this model):

- $T^{realization}_{P_iP_j}(t+\delta t)$: It is computed as the probability of determining whether both the participants P_i and P_j are within same group at a time interval $[t, t+\delta t]$.
- $T^{proximity}_{P_iP_j}(t+\delta t)$: A geo-social based distance measure [234] is used to find the distance between two participants at a time interval $[t, t+\delta t]$. If distance is smaller then less effort is needed to initiate interaction of the participants.
- $T^{betweenness}_{P_iP_j}(t+\delta t)$: Participant P_i creates trusted links between itself and an participant P_j (belonging to a different group) based on identifying family members or close friends over time interval $[t, t+\delta t]$.
- $T^{chunk}_{P_iP_j}(t+\delta t)$: The social trust chunk value is established between participant P_i and participant P_j, if and only if there are more common neighbors between the participants P_i and P_j.

The indirect trust $T^{ID,A}_{P_iP_j}(t+\delta t)$ is computed as follows:

$$T^{ID,A}_{P_iP_j}(t+\delta t) = \begin{cases} e^{-\theta\delta t} * T^{ID,A}_{P_iP_j}(t), & if|S_i| = 0 \\ \frac{\sum_{k\in S_i}(T^{A}_{P_iP_k}(t) * T^{A}_{P_kP_j}(t))}{\sum_{k\in S_i}(T^{A}_{P_iP_k}(t)}, & if|S_i| \geq 1 \end{cases} \tag{6.6}$$

When a participant P_i finds an intermediate participant P_k as a trustworthy participant (based on trust information and social relationships), then the participant P_k provides recommendations to participant P_i for evaluating a participant P_j. Let $|S_i|$ represents the number of elements in a set containing participant $P_i's$ 1-hop neighbors. The indirect trust value should be greater than the social trust threshold value (T_f). A directed link is established between participant P_i and the participant P_j if and only if $T_{P_iP_j}(t) \geq T_f$. This helps for selecting trustworthy next hop neighbors.

6.3.2.2 Relevance Degree Association

Relevance degree reduces significantly as the number of transitive hop increases between the intermediate participants in a social trust propagation model. The associated relevance degree value r in a social path $P(P_1, P_2, ...P_n)$ at a time t is defined as follows:

$$r_{P(P_1,P_2,.....P_n)} = \prod_{(P_i,P_j)\in(P(P_1,P_2,.....P_n))} r_{P_iP_j}(t) \tag{6.7}$$

6.3.2.3 Recommended Influence Value Association

The recommended influence value of a participant does not reduce with the increase in the number of transitive hops. The associated recommended influence value ρ in a social path $P(P_1, P_2, ...P_n)$ at a time t is defined as follows:

$$\rho_{P(P_1,P_2,.....P_n)} = \frac{\sum_{i=1}^{n} \rho_{P_i}(t)}{n} \tag{6.8}$$

As shown in Fig.6.3, the *HoS* associated models are determined using the Eq.(2), Eq.(7) and Eq.(8) between a source participant and a target participant. Therefore, *HoS* associated model is considered as basic approach for establishing strong social connections among the group of participants. The *HoS* attribute values are sent to the service registry.

6.3.3 Entropy Based Trust Model

In this section, to deal with variability (i.e., *Big Data* challenges) Shannon's entropy based trust model is proposed by considering trust, relevance degree and recommended influence value. Claude E. Shannon introduced *Shannon's* communication and information theory in 1948 [237]. Shannon entropy is an important measure to quantify uncertainty of information content.

In online social networks, a set of n participants $P = \{P_1, P_2, ...P_n\}$ are considered. The key attributes for providing trustworthy services denoted as $A = \{A_1, A_2, ...A_n\}$. In this work, a sample set *S* is considered and the set contains data tuples that are defined by considering a set of *High quality of Social trust (HoS)* attributes, such as trust (T), relevance degree (r), recommended influence value (ρ) and a class attribute (which has *k* unique values). The class attribute gives the information about a tuple in the sample set *S*. Therefore, there are *k* unique classes C_i (for i = 1,2...k). In this work, there are two classes (C_1 and C_2). The class C_1 belongs to trustworthy link and class C_2 belongs to untrustworthy link. The *HoS* attribute with a highest value of information gain is considered as a most influential *HoS* attribute, while providing *n* services from a participant P_i to a participant P_j. Therefore, this *HoS* attribute reduces the information required in order to classify the data tuples in each subset and leads to impurity in these subsets. According to Shannon entropy [238], the information required in order to classify a data tuple in sample set *S* is defined as:

$$Info(S) = -\sum_{i=1}^{k} \frac{|C_i|}{|S|} log_2 \frac{|C_i|}{|S|} \tag{6.9}$$

where $|S|$ denotes the total number of data tuples in sample set *S* and $|C_i|$ denotes the number of data tuples in class C_i. Info(S) (i.e. entropy) represents the average amount of information required in order to classify the class label information in sample set S with *n* number of services.

Sample set *S* on any *HoS* attribute *'A'* contains *m* unique values, $\{v_1, v_2, ...v_m\}$, from the sample set *S*. Therefore, *HoS* attribute *'A'* divides sample set *S* into *m* subsets, $\{S_1, S_2, ..., S_m\}$, where S_j contains tuples in *S* that have value v_j of *'A'*. In this work, trust attribute contains 3 unique values, such as direct, indirect and past information. Therefore, trust attribute is divided into 3 subsets. However, each subset may likely to be impure (where a subset contain tuples from multiple classes instead from a single class). Moreover, each subset S_j is further divided into a *k* classes C_i (for i = 1,2....k) as $\{S_j(C_1), S_j(C_2), ...S_j(C_k)\}$. For any subset $S_j(C_i)$, the participant P_i provides *n* number of services to the participant P_j in a sample set *S* with a *HoS* attribute *'A'*. Therefore, the expected amount of information needed from *S* by partitioning into subsets is defined as

$$Info_A(S) = -\sum_{j=1}^{m} \frac{|S_j|}{|S|} \sum_{i=1}^{K} \frac{|S_j(C_i)|}{|S_j|} log_2 \frac{|S_j(C_i)|}{|S_j|} \tag{6.10}$$

where $|S_j|$ denotes number of tuples in subset S_j. $|S_j(C_i)|$ denotes number of tuples in subset S_j of class C_i on some ***HoS*** attribute *'A'*. $Info_A(S)$ indicates the expected value of information needed in order to classify the data tuples from S based on partitioning by the most influential ***HoS*** attribute *'A'*. If $Info_A(S)$ is smaller then the impurity in the subsets is low. Therefore, the information gain is defined as follows:

$$Gain(A) = Info(S) - Info_A(S) \tag{6.11}$$

Gain(A) indicates that the amount of impurity in the subsets S_j for a trustworthy selective decision reduces significantly after splitting the ***HoS*** attribute *'A'* in a sample set *S*. Therefore, high value of Gain(A) indicates the most influential ***HoS*** attribute *'A'* for best classification. Accordingly, the weights of each ***HoS*** attribute *'A'* for providing a trustworthy service that is defined as:

$$w_i = \frac{Gain(A_i)}{\sum_{i=1}^{n} Gain(A_i)} \tag{6.12}$$

To adjust a set of attribute weights $w = \{w_1, w_2, ..., w_n\}$, utility function in this model is considered as a measurement for evaluating the trustworthy service in online social networks by incorporating trust, relevance degree and recommended influence value. The utility value U in a recommended trust path $P(P_1, P_2, ...P_n)$ is determined as:

$$U_{P(P_1,P_2,.....P_n)} = \sum_{i=1}^{n} w_i A_i \tag{6.13}$$

where $\sum_{i=1}^{n} w_i = 1$ and $0 < w_i < 1$.

The above Eq. (13) can be rewritten by considering trust (T), relevance degree (r) and recommended influence value (ρ)

$$U(P_{P_1,P_2,.....P_n}) = w_1 T_{P(P_1,P_2,.....P_n)} + w_2 r_{P(P_1,P_2,.....P_n)} + w_3 \rho_{P(P_1,P_2,.....P_n)} \tag{6.14}$$

The main objective of finding a trustworthy selective decision in online social networks is to select a recommended trust path that satisfies *HoS* constraints (as mentioned in Section 6.3.2) and obtains best utility value.

6.3.4 Proposed LA-RTPS Algorithm

A *Learning Automata based Recommended Trust Path Selection* (LA-RTPS) algorithm (refer Algorithm 6.1) has been proposed in online social networks. Let $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ denotes a finite set of paths between source participant v_i and target participant v_j, where α_i is the recommended trust path selected by an intermediate participant v_k at an instant i. Learning automaton uses recommended trust path selection in online social networks depending on parameters namely *direct trust*, *indirect trust* (T), *relevance degree* (r) and *recommended influence value* (ρ) of a participant. The proposed LA-RTPS searches a recommended trust path in online social networks by using *Dijkstra's* single source shortest path algorithm [239]. In the recommended trust path selection procedure from a source participant v_i to a target participant v_j at an intermediate node v_k, there exists either a path $P^U_{v_i \to v_j}$ (which is identified with maximal utility value U using Eq.(14)) or an *Aggregate Path (AP)* $AP^U_{v_i \to v_j}$ (refer Section 6.2.1, Definition 5). The action probability value is given by

$$p_j = \frac{1}{deg(v_k)} \tag{6.15}$$

where $deg(v_k)$ is the degree of an intermediate participant v_k.

The *LA-RTPS* algorithm computes *High quality of Social trust (HoS)* attribute weights using Eq.(12) (Line 2). The *LA-RTPS* algorithm starts on the selection of source participant v_i and learning automata is activated at each intermediate participant v_k (Line 3-4). To identify the recommended trust path ($P_{v_i \to v_j}$) from a source participant v_i to a target participant v_j through an intermediate participant v_k, the *HoS* attributes, such as trust, relevance degree, recommended influence value and utility values are determined. If one of the path satisfies a threshold value *T* then a source participant v_i is inserted into a queue. A neighboring participant of v_i is v_y (which is chosen with maximal utility value of $P_{v_i \to v_y}$). The

Algorithm 6.1 Learning Automata based Recommended Trust Path Selection (LA-RTPS)

```
Input:
    $v_i, v_j, v_k$, Convergence Threshold T
Output:
    $P_{v_i \to v_j}, U(P_{v_i \to v_j})$
Parameters:
    $a_r$: Number of times a specific action is rewarded, $a_p$: Number of times a specific action is penalized
Assumption:
    $\alpha = \{\alpha_1, \alpha_2, ......, \alpha_n\}$ denotes a finite set of paths between source participant $v_i$ and target participant $v_j$, where $\alpha_i$ is the path selected by an intermediate participant $v_k$ at an instant $i$
Procedure:
1: $P_{v_i \to v_j} = \Phi$
2: Compute *HoS* attribute weights based on Shannon's entropy using Eq.(12)
3: Select a source participant
4: Activate learning automata at each intermediate participant $v_k$
5: for i do = 1 to n //n is the total number of intermediate participants
6:     Find the recommended trust path via an intermediate participant $v_k$ as follows
7:     for a given path $P_{v_i \to v_k \to v_j}$ at time $[t + \delta t]$
8:     Compute the trust value using Eq.(2)
9:     Compute the associated relevance degree using Eq.(7)
10:    Compute the associated recommended influence value using Eq.(8)
11:    Compute the utility value U for a given path $P_{v_i \to v_k \to v_j}$ using Eq.(14)
12:    if one of$\{U(P_{v_i \to v_k \to v_j})$ and $U(AP_{v_i \to v_k \to v_j}) \geq T\}$ then
13:        Set Queue= $\Phi$, $P_{v_i \to v_i} = v_i$
14:        Add $v_i$ into Queue
15:        while $Queue \neq \Phi$ do
16:            for $v_y \in Neighbor[v_x]$, where $(v_x \in Queue)$ do
17:                Select $Neighbor[v_x]$ with maximal $\mu$ as $v_y$
18:                Add $v_y$ into Queue and $P_{v_i \to v_j} = P_{v_i \to v_y} + v_y \to v_k \to v_j$
19:            end for
20:            Remove $v_x$ from Queue
21:        end while
22:        Store the utility value U and the associated path are stored in a buffer
23:    else
24:        No feasible solution
25:    end if
26: end for
```

```
27: for j do = 1 to n
28:     if ($j^{th}$ intermediate participant is trustworthy) then
29:         reward the intermediate participant and the associated path
30:         $U = U + \omega$, where $\omega \in [0, 0.1]$
31:         Action probability value is updated using Eq.(1)
32:         $a_r + +$
33:     else
34:         penalize the intermediate participant and the associated path
35:         $U = U - \omega$, where $\omega \in [0, 0.1]$
36:         Action probability value is updated using Eq.(1)
37:         $a_p + +$
38:     end if
39: end for
40: return Maximal utility value U and the associated path
```

Table 6.1: Entropy, Gain and Weights associated with a sample set S

Info(S)=0.7822			
A	$Info_A(S)$	Gain(A)	Wts
$A_1(T)$	0.4310	0.3512	0.4494
$A_2(r)$	0.5217	0.2605	0.3333
$A_3(\rho)$	0.6125	0.1697	0.2171

participant v_y is inserted into the queue. Moreover, the utility value and the associated path are stored in a buffer. This process is repeated for all intermediate participants (Line 5-26). Learning automata produces the response in terms of either penalty or reward. Moreover, if an intermediate participant v_k is considered as a trustworthy participant then the learning automata is rewarded. If learning action is rewarded then the utility value is incremented by ω, where $\omega \in [0, 0.1]$. Otherwise, learning action is penalized and the utility value is decremented by ω (Line 27-39). Therefore, the learning algorithm returns the maximal utility value with best recommended trust path (Line 40) .

6.3.5 Analysis of Proposed Trust Model

In this section, the performance of the proposed model is analyzed by considering Trust

Table 6.2: Proposed Trusted Model Computation Results

Path	Links	A_1(T)	A_2(r)	$A_3(\rho)$	Utility
$P_{v_i \to v_3}$	$v_i \to v_1 \to v_2 \to v_3$	0.6	0.5	0.6	0.57
$P_{v_3 \to v_j}$	$v_3 \to v_4 \to v_j$	0.5	0.6	0.5	0.53
$P_{v_2 \to v_j}$	$v_2 \to v_3 \to v_4 \to v_j$	0.7	0.8	0.7	0.73

T, relevance degree r and recommended influence value ρ as *HoS* attribute set. For each attribute, entropy, gain and weights are listed in Table 6.1. Fig. 6.4 illustrates a social network structure between a source participant v_i and a target participant v_j with six intermediate participants. For a given path, *HoS* attribute values are computed (Table 6.2) based on *HoS* attribute associated model (refer Section 6.3.2).

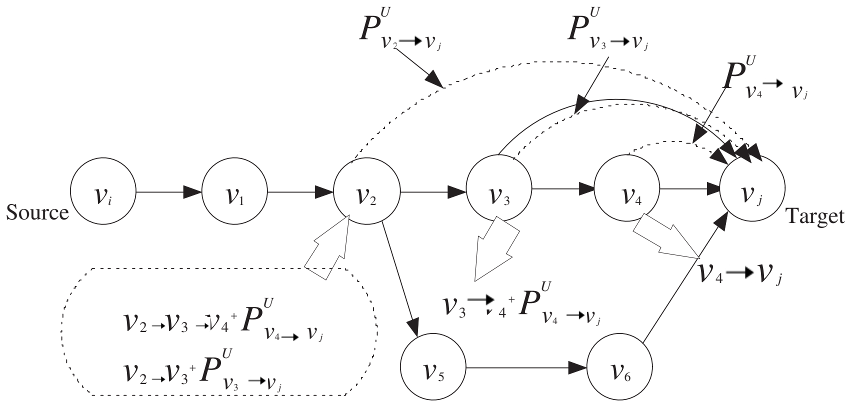


Figure 6.4: Multiple paths are aggregated for the recommended trust path selection

For a recommended trust path, a path ($P^U_{v_i \to v_j}$) has to be identified from v_i to v_j through an intermediate node v_k with maximal utility value U (which is determined using Eq. (14)) by satisfying *HoS* constraints. A given path $P^U_{v_i \to v_j}$ is considered as a recommended trust path if and only if utility value U is greater than a threshold value T. LA-RTPS identifies *m* paths (from v_i to v_j) which are to be aggregated in order to find a best recommended trust path (i.e., path $AP^U_{v_i \to v_j}$). For example, in Fig. 6.4 from v_3 to v_j, there exists only one path through an intermediate node v_4. Similarly, at v_2 there are two intermediate nodes (v_3 and v_4) between v_2 to v_j. LA-RTPS identifies two paths. The first path is $AP^U_{v_2 \to v_j}$ $(i.e., v_2 \to v_3 \to v_4 \to v_j)$. The second path is $AP^U_{v_2 \to v_j}$ $(i.e., v_2 \to v_3 \to v_j)$. The paths will be aggregated with maximal utility value U in order to find a best recommended trust

Table 6.3: Weights associated with T, r and ρ values

W_ID	wt_T	wt_r	wt_ρ
1	0.5	0.3	0.2
2	0.25	0.25	0.5
3	0.5	0.25	0.25

Table 6.4: Features of online social subnetworks with a sub-network *ID* varied from 5 to 8 hops

2*	Slashdot Dataset			Epinions Dataset		
Hops	SID	Nodes	Links	SID	Nodes	Links
5	1	423	1523	20	535	1626
6	1	674	2875	20	784	2985
7	1	1400	6823	20	1282	6698
8	1	935	5723	20	868	4585

path. Table 6.2 shows the results of the proposed trust model which are computed based on the weights specified in Table 6.1.

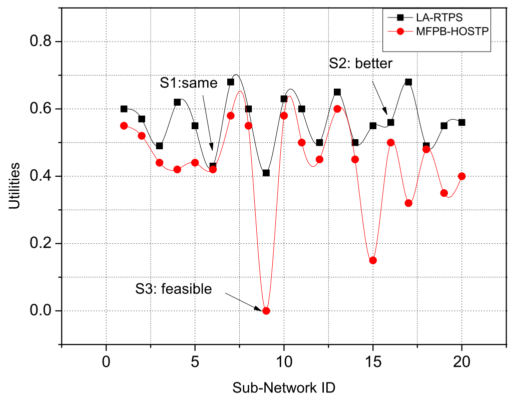
6.4 Performance Evaluation

In this section, experimental results are presented to evaluate the performance of *LA-RTPS* algorithm by considering two datasets, such as *Slashdot* dataset and *Epinions* dataset [43] and comparing with *MFPB-HOSTP* [44]. The two datasets captures real-world characteristics of online social networks and these datasets are extracted from the *Slashdot* [231] and *Epinions* [232] websites published by the *Stanford Network Analysis Project* [240].

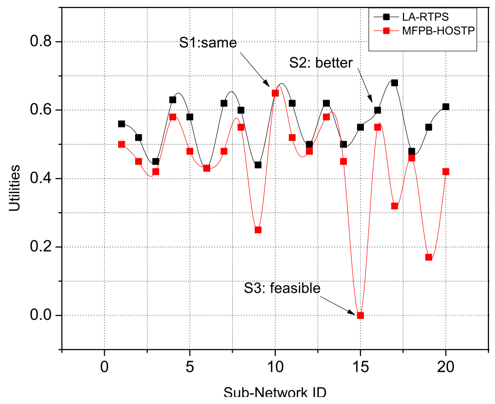
HoS attribute values such as trust (T), relevance degree (r) and recommended influence value (ρ) are considered and they are randomly generated (because different *Big Data* applications may have different values). *HoS* constraints are set by a source participant as $H_{v_i,v_j} = \{T_{v_i,v_j} > 0.4, r_{v_i,v_j} > 0.3, \rho_{v_i,v_j} > 0.4\}$. Moreover, different weights of *HoS* attributes are taken from Table 6.3. Arbitrarily 100 subnetworks are chosen from both *Slashdot* and *Epinions* dataset. Further, the maximum length of the recommended trust path is varied from 5 to 7 hops based on the real-world characteristics. These subnetworks are categorized by number of hops and sub-network *IDs* varying from 1 to 20. Table 6.4 lists the features of online social subnetworks with a sub-network *ID*.

6.4.1 Experimental Results

In this section, we analyze the performance of the proposed algorithm in terms of recommended trust path utilities and execution time.

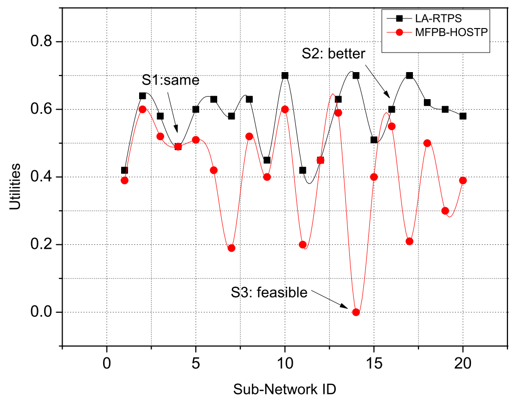


(a) Slashdot Dataset

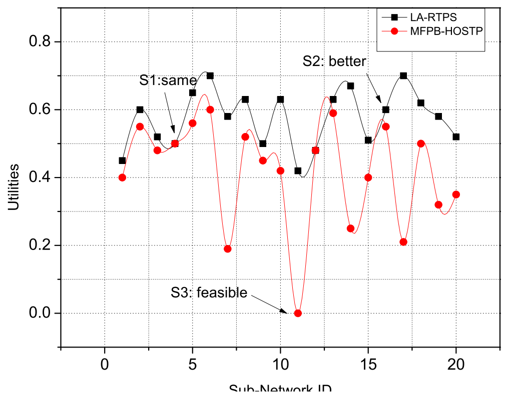


(b) Epinions Dataset

Figure 6.5: Comparison of recommended trust path utilities of online social subnetworks with 5 hops.



(a) Slashdot Dataset

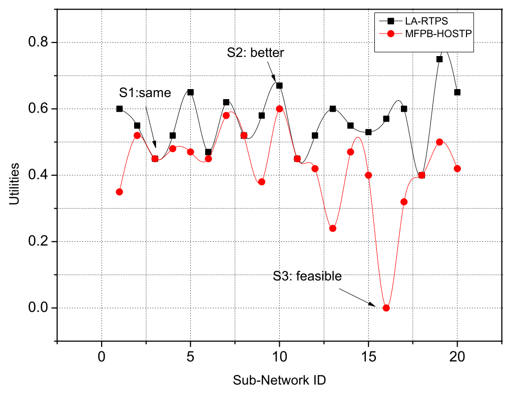


(b) Epinions Dataset

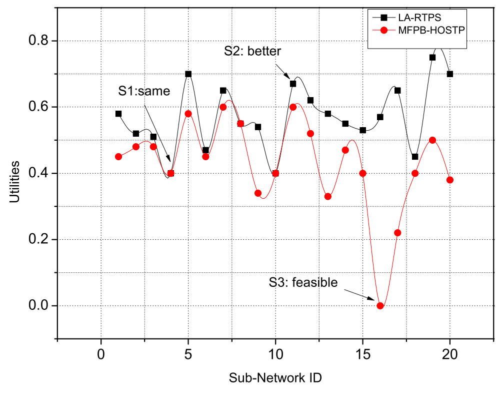
Figure 6.6: Comparison of recommended trust path utilities of online social subnetworks with 6 hops.

6.4.1.1 Recommended Trust Path Utilities

Fig. 6.5, Fig. 6.6 and Fig. 6.7 shows the utilities of the recommended trust paths in online social subnetworks that are grouped based on the number of hops with different weights

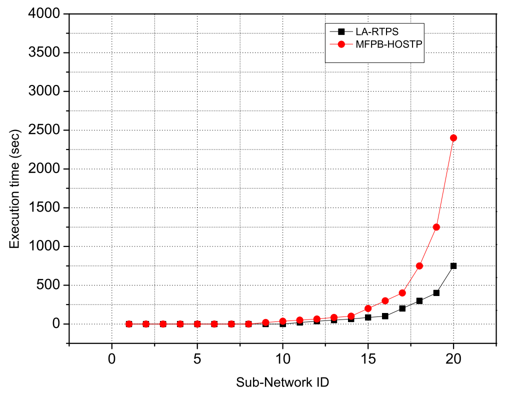


(a) Slashdot Dataset

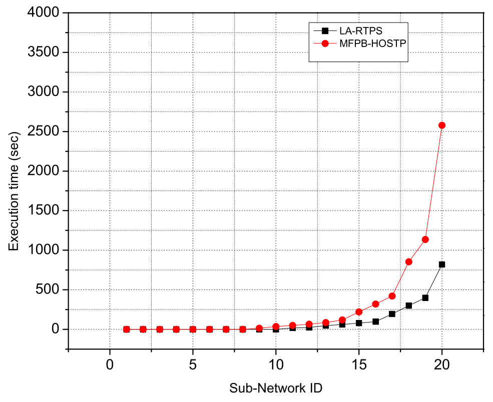


(b) Epinions Dataset

Figure 6.7: Comparison of recommended trust path utilities of online social subnetworks with 7 hops.



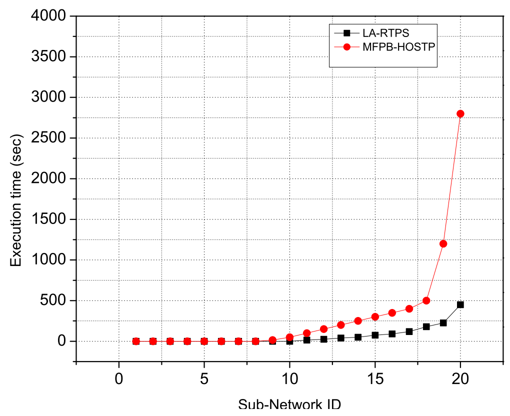
(a) Slashdot Dataset



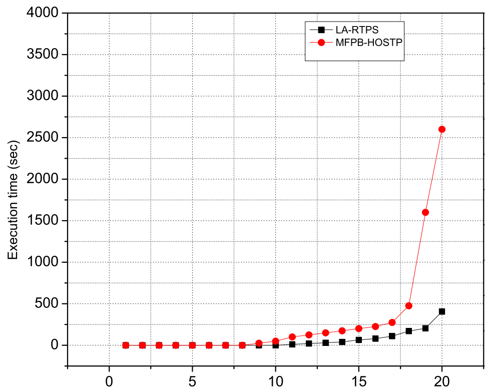
(b) Epinions Dataset

Figure 6.8: Comparison of total execution time of online social subnetworks with 5 hops.

of *HoS* constraints. It has been observed from Fig. 6.5 to Fig. 6.7 that the proposed algorithm obtains the utilities that outperforms *MFPB-HOSTP* algorithm [44] (e.g., case S1 and S2 in Fig. 6.5, Fig. 6.6 and Fig. 6.7). The reason is that, in *LA-RTPS*, the utility values fluctuate due to use of random functions for generating *HoS* attributes. LA-RTPS identifies multiple paths which are aggregated and selects a best recommended trust path with maximal utility value by satisfying *HoS* constraints. However, when there exists a path $P_{v_i \to v_j}$ with minimal utility value then *MFPB-HOSTP* algorithm stops identifying the recommended trust path (e.g., case S3 in Fig. 6.5, Fig. 6.6 and Fig. 6.7). The reason is that the existing algorithm uses trust value without good recommendations in order to

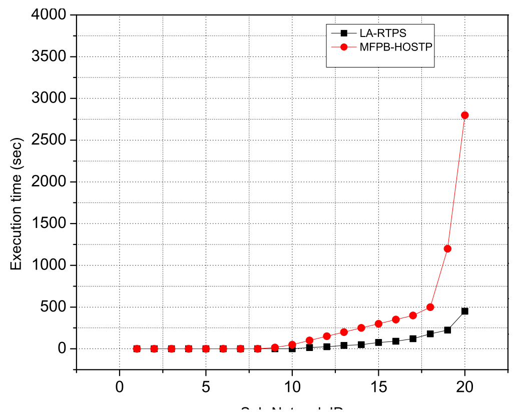


(a) Slashdot Dataset

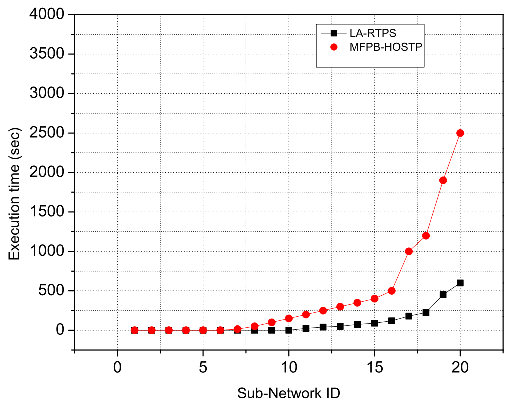


(b) Epinions Dataset

Figure 6.9: Comparison of total execution time of online social subnetworks with 6 hops.



(a) Slashdot Dataset



(b) Epinions Dataset

Figure 6.10: Comparison of total execution time of online social subnetworks with 7 hops.

identify the social trust paths. Due to this a best recommended trust path may not be found in all possible cases by satisfying *HoS* constraints. Therefore, *LA-RTPS* always find a best recommended trust path that outperforms existing algorithm. For *Slashdot* dataset, it can be observed that average value of utilities of *LA-RTPS* is 26.78% better in Fig. 6.5(a), 32.75% better in Fig. 6.6(a) and 35.38% better in Fig. 6.7(a) in comparison to *MFPB-HOSTP* [44]. For *Epinions* dataset, it has been observed that average value of utilities of *LA-RTPS* is 31.14% better in Fig. 6.5(b), 33.69% better in Fig. 6.6(b) and 29.14% better in Fig. 6.7(b) in comparison to *MFPB-HOSTP* [44].

6.4.1.2 Execution Time of LA-RTPS

Fig. 6.8, Fig. 6.9 and Fig. 6.10 shows the execution time of a recommended trust path in online social subnetworks based on the number of hops. From Fig. 6.8 to Fig. 6.10, it can be observed that when the subnetwork ID is small, the performance of both *LA-RTPS* and *MFPB-HOSTP* are similar in terms of execution time. Moreover, when the number of subnetwork ID increases, it has been observed that *LA-RTPS* outperforms existing algorithm in execution time. The reason is that, to identify the recommended trust path, any *HoS* attribute in the selected trust path from v_i to v_k does not satisfy *HoS* constraints. The node v_k is not selected for the next searching process. The existing algorithm uses forward and backward search approach in order to identify the local paths which take more execution time. For *Slashdot* dataset, the average execution time of *LA-RTPS* is 30.11% less in Fig. 6.8(a), 35.24% in Fig. 6.9(a) and 34.28% in Fig. 6.10(a) in comparison to *MFPB-HOSTP* [44]. For *Epinions* dataset, the average execution time of *LA-RTPS* is 35.39% less in Fig. 6.8(b), 36.14% less in Fig. 6.9(b) and 35.33% less in Fig. 6.10(b) in comparison to *MFPB-HOSTP*.

6.5 Summary

In this chapter, a trust model has been presented for user recommendations based on trust information (such as direct trust and indirect trust), relevance degree and recommended influence values in online social networks. For selecting the recommended trust paths with *HoS* attributes in online social networks, firstly, a recommendation-based online social network architecture has been designed. In addition, a *Learning Automata based Recommended Trust Path Selection* (LA-RTPS) algorithm has been proposed, where multiple recommended trust paths are identified from a source participant to the target participant. The proposed model determines utility values for evaluating trustworthy services based on *Shannon* entropy. The multiple recommended trust paths are aggregated and a best recommended trust path is chosen with maximal utility value.

Chapter 7

Conclusions and Future Directions

This thesis investigates the detection of social bots which provides trusted and efficient sharing and accessing information in online social networks. Different trust computational models are presented by taking social attributes (or features) into consideration. The proposed algorithms achieve better performance in terms of the accuracy, precision, recall and F-measure. The proposed algorithms are implemented and experimented on real-time online social network datasets. A comparative study of the proposed algorithms and the experimental results demonstrate that there is a necessity of the proposed algorithms which effectively analyze and detect the participants (i.e., online social networking user accounts) based on their behavioral patterns in online social networks.

In this thesis, the main limitations related to social bot detection, such as reducing trust value of the legitimate participants by sending fake and untrustworthy information, generating multiple fake identities and performing phishing attacks through URL redirection chains have been addressed. Thus, it is important to detect the social bots from legitimate users in online social networks. In this thesis, the important features (such as tweet-content, user profile, URL, graph and behavioral similarity based features) are taken into consideration to analyze the behavior of a participant. This thesis has made contributions by considering the above-mentioned features in order to evaluate the trust value of a participant and trustworthy paths in online social networks for providing trusted user recommendations. Contributions of this thesis are identification of social bots and botnet communities, and analyze the influence of social bots in online social networks.

7.1 The Major Contributions of the Thesis

A learning automata based malicious social bot detection approach has been proposed by integrating a trust computational model with a set of URL-based features for malicious social bot detection. This approach is presented in Chapter 3. The proposed trust computation model contains two parameters namely, direct trust and indirect trust. Moreover, the direct trust is derived from *Bayes'* theorem and indirect trust is derived from *Dempster-Shafer Theory* to evaluate the trustworthiness of tweets (posted by each participant).

In Chapter 4, a single agent deep Q-network based architecture has been designed 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). A multi-agent particle swarm optimization based deep Q-learning algorithm has been proposed using the updation strategy of Q-value based on determining local and global best action sequences in order to detect social bots more accurately. Further, a top-k influential user algorithm has been proposed to identify the most influential users based on tweets and the user's interactions.

Based on the behavioral similarity, Chapter 5 presents a novel two-phase model for detecting social botnet communities. The first phase, called the community formation phase, uses the social botnet community detection algorithm to distinguish legitimate participants among social botnet communities. The second phase, called the community reconstruction phase, involves deep autoencoder based social botnet community detection algorithm in order to classify different types of social botnet communities with improved precision. Further, an influential community detection approach has been developed to minimize the disseminating of spam-content through influential communities in Twitter network.

In Chapter 6, a learning automata based recommended trust path selection algorithm has been designed to determine the trusted user recommendations. In the proposed algorithm, a trust model for user recommendations are taken into consideration based on trust information (such as direct trust and indirect trust), relevance degree and recommended influence value in online social networks. Moreover, the proposed model determines trust path utility values (in terms of trust, relevance degree and recommended influence value) for evaluating trustworthy services based on Shannon entropy.

7.2 Future Directions

Although the proposed social bot detection algorithms show performance improvement over other existing approaches available in the literature. Moreover, there are other aspects and scenarios which can be taken into consideration. The extensions of the research work presented in this thesis are presented as follows:

Online rating and recommended systems play a vital role in affecting consumers' opinion for a selective decision making. However, providing an efficient method for identification and classification of malicious comments or fake information posted by social bots in online rating and reviews is a research challenging task in OSNs. Research can be extended by considering two assumptions that malicious social bots are less likely to have strong social relationship with legitimate participants, and the social interaction between two participants and messages can be modeled as a social network graph structure for detecting malicious comments posted in online recommendation systems. Further, the influence of social botnet communities on the online rating and recommended systems can be investigated with different types of attack models.

Identification of emotional malicious social bots from tweets play an important role to analyze user's behavior in OSNs. Research can be extended by considering bag-of-words models and linguistic based features which may help in identifying different types of emotional bots more accurately. In Chapter 3, each feature is assumed to be conditionally independent. However, there may be dependency among the features. Research may be extended to investigate the dependency among the features and its impact on malicious social bot detection in Twitter network. In Chapter 4, single and multi-agent offline deep reinforcement learning models have been presented. These approaches can be further explored in the presence of an interactive environment with online experiments in Twitter network. As a future research scope, the problem addressed in Chapter 5 can be further investigated by analyzing similar type of malicious social botnet communities across multiple social networks.

Bibliography

[1] M. Al-Qurishi, M. Al-Rakhami, A. Alamri, M. Alrubaian, S. M. M. Rahman, and M. S. Hossain, “Sybil defense techniques in online social networks: a survey,” *IEEE Access*, vol. 5, pp. 1200–1219, 2017.

[2] O. Loyola-González, R. Monroy, J. Rodríguez, A. López-Cuevas, and J. I. Mata-Sánchez, “Contrast pattern-based classification for bot detection on twitter,” *IEEE Access*, vol. 7, pp. 45 800–45 817, 2019.

[3] G. Liu, F. Zhu, K. Zheng, A. Liu, Z. Li, L. Zhao, and X. Zhou, “Tosi: A trust-oriented social influence evaluation method in contextual social networks,” *Neurocomputing*, vol. 210, pp. 130–140, 2016.

[4] E. Van Der Walt and J. Eloff, “Using machine learning to detect fake identities: bots vs humans,” *IEEE Access*, vol. 6, pp. 6540–6549, 2018.

[5] R. D. Perera, S. Anand, K. Subbalakshmi, and R. Chandramouli, “Twitter analytics: Architecture, tools and analysis,” in *Military Communications Conference, 2010-MILCOM 2010*. IEEE, 2010, pp. 2186–2191.

[6] F. Daniel, C. Cappiello, and B. Benatallah, “Bots acting like humans: Understanding and preventing harm,” *IEEE Internet Computing*, 2019.

[7] S. B. Jr, G. F. Campos, G. M. Tavares, R. A. Igawa, M. L. P. Jr, and R. C. Guido, “Detection of human, legitimate bot, and malicious bot in online social networks based on wavelets,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1s, pp. 1–17, 2018.

[8] P. Shi, Z. Zhang, and K.-K. R. Choo, “Detecting malicious social bots based on clickstream sequences,” *IEEE Access*, vol. 7, pp. 28 855–28 862, 2019.

[9] M. Sirivianos, K. Kim, and X. Yang, “Socialfilter: Introducing social trust to collaborative spam mitigation,” in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2300–2308.

[10] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, “Botnet detection using graph-based feature clustering,” *Journal of Big Data*, vol. 4, no. 1, p. 14, 2017.

[11] S. Lee and J. Kim, “Fluxing botnet command and control channels with url shortening services,” *Computer Communications*, vol. 36, no. 3, pp. 320–332, 2013.

[12] G. Yan, “Peri-watchdog: Hunting for hidden botnets in the periphery of online social networks,” *Computer Networks*, vol. 57, no. 2, pp. 540–555, 2013.

[13] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting automation of twitter accounts: Are you a human, bot, or cyborg?” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 811–824, 2012.

[14] S. Kudugunta and E. Ferrara, “Deep neural networks for bot detection,” *Information Sciences*, vol. 467, pp. 312–322, 2018.

[15] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 561–576, 2017.

[16] D. Choi, J. Han, S. Chun, E. Rappos, S. Robert, and T. T. Kwon, “Bit. ly/practice: Uncovering content publishing and sharing through url shortening services,” *Telematics and Informatics*, vol. 35, no. 5, pp. 1310–1323, 2018.

[17] F. Klien and M. Strohmaier, “Short links under attack: geographical analysis of spam in a url shortener network,” in *Proceedings of the 23rd ACM conference on Hypertext and social media*. ACM, 2012, pp. 83–88.

[18] C.-M. Chen, D. Guan, and Q.-K. Su, “Feature set identification for detecting suspicious urls using bayesian classification in social networks,” *Information Sciences*, vol. 289, pp. 133–147, 2014.

[19] T. M. Chen and V. Venkataramanan, “Dempster-shafer theory for intrusion detection in ad hoc networks,” *IEEE Internet Computing*, vol. 9, no. 6, pp. 35–41, 2005.

[20] K. Lee, B. D. Eoff, and J. Caverlee, “Seven months with the devils: A long-term study of content polluters on twitter.” in *ICWSM*, 2011.

[21] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,” in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 963–972.

[22] C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, “Statistical features-based real-time detection of drifted twitter spam,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 914–925, 2017.

[23] K. Hans, L. Ahuja, and S. Muttoo, “Detecting redirection spam using multilayer perceptron neural network,” *Soft Computing*, vol. 21, no. 13, pp. 3803–3814, 2017.

[24] C. Yu, M. Zhang, F. Ren, and G. Tan, "Emotional multiagent reinforcement learning in spatial social dilemmas," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 12, pp. 3083–3096, 2015.

[25] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[26] P. Das, H. Behera, and B. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical q-learning and improved particle swarm optimization with perturbed velocity," *Engineering Science and Technology, an International Journal*, vol. 19, no. 1, pp. 651–669, 2016.

[27] Z. Gilani, L. Wang, J. Crowcroft, M. Almeida, and R. Farahbakhsh, "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, 2016, pp. 37–38.

[28] R. Tang, S. Fong, S. Deb, A. V. Vasilakos, and R. C. Millham, "Dynamic group optimisation algorithm for training feed-forward neural networks," *Neurocomputing*, vol. 314, pp. 1–19, 2018.

[29] Q. Wei, F. L. Lewis, Q. Sun, P. Yan, and R. Song, "Discrete-time deterministic q-learning: A novel convergence analysis," *IEEE transactions on cybernetics*, vol. 47, no. 5, pp. 1224–1237, 2017.

[30] A. Barushka and P. Hajek, "Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks," *Applied Intelligence*, pp. 1–19, 2018.

[31] W.-B. Du, W. Ying, G. Yan, Y.-B. Zhu, and X.-B. Cao, "Heterogeneous strategy particle swarm optimization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 467–471, 2016.

[32] M. Alshahrani, F. Zhu, L. Zheng, S. Mekouar, and S. Huang, "Selection of top-k influential users based on radius-neighborhood degree, multi-hops distance and selection threshold," *Journal of Big Data*, vol. 5, no. 1, p. 28, 2018.

[33] A. Sheikhahmadi, M. A. Nematbakhsh, and A. Zareie, "Identification of influential users by neighbors in online social networks," *Physica A: Statistical Mechanics and its Applications*, vol. 486, pp. 517–534, 2017.

[34] J. Zhang, R. Zhang, J. Sun, Y. Zhang, and C. Zhang, "Truetop: A sybil-resilient system for user influence measurement on twitter," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2834–2846, 2016.

[35] B. Wang, J. Jia, L. Zhang, and N. Z. Gong, "Structure-based sybil detection in social networks via local rule-based propagation," *IEEE Transactions on Network Science and Engineering*, 2018.

[36] P. Bindu, R. Mishra, and P. S. Thilagam, "Discovering spammer communities in twitter," *Journal of Intelligent Information Systems*, vol. 51, no. 3, pp. 503–527, 2018.

[37] J. Wang and I. C. Paschalidis, "Botnet detection based on anomaly and community detection," *IEEE Transactions on Control of Network Systems*, 2016.

[38] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 665–674.

[39] S. Wen, M. S. Haghighi, C. Chen, Y. Xiang, W. Zhou, and W. Jia, "A sword with two edges: Propagation studies on both positive and negative information in online social networks," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 640–653, 2014.

[40] E. Choo, T. Yu, and M. Chi, "Detecting opinion spammer groups and spam targets through community discovery and sentiment analysis," *Journal of Computer Security*, vol. 25, no. 3, pp. 283–318, 2017.

[41] Q. Dang, Y. Zhou, F. Gao, and Q. Sun, "Detecting cooperative and organized spammer groups in micro-blogging community," *Data mining and knowledge discovery*, vol. 31, no. 3, pp. 573–605, 2017.

[42] W. Tan, M. B. Blake, I. Saleh, and S. Dustdar, "Social-network-sourced big data analytics," *IEEE Internet Computing*, vol. 17, no. 5, pp. 62–69, 2013.

[43] K. Xu, K. Zou, Y. Huang, X. Yu, and X. Zhang, "Mining community and inferring friendship in mobile social networks," *Neurocomputing*, vol. 174, pp. 605–616, 2016.

[44] G. Liu, Y. Wang, M. A. Orgun, and E.-P. Lim, "Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 152–167, 2013.

[45] M. Fire, R. Goldschmidt, and Y. Elovici, "Online social networks: threats and solutions," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2019–2036, 2014.

[46] F. Ullah and S. Lee, "Community clustering based on trust modeling weighted by user interests in online social networks," *Chaos, Solitons & Fractals*, vol. 103, pp. 194–204, 2017.

[47] A.-Z. Ala'M, H. Faris, J. Alqatawna, and M. A. Hassonah, "Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts," *Knowledge-Based Systems*, vol. 153, pp. 91–104, 2018.

[48] H. Lin, J. Jia, J. Qiu, Y. Zhang, G. Shen, L. Xie, J. Tang, L. Feng, and T.-S. Chua, "Detecting stress based on social interactions in social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1820–1833, 2017.

[49] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Information Sciences*, vol. 467, pp. 312 – 322, 2018.

[50] M. Al-Qurishi, M. S. Hossain, M. Alrubaian, S. M. M. Rahman, and A. Alamri, "Leveraging analysis of user behavior to identify malicious activities in large-scale social networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 799–813, 2017.

[51] V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer, "The darpa twitter bot challenge," *Computer*, vol. 49, no. 6, pp. 38–46, 2016.

[52] S. Lee and J. Kim, "Warningbird: A near real-time detection system for suspicious urls in twitter stream," *IEEE transactions on dependable and secure computing*, vol. 10, no. 3, pp. 183–195, 2013.

[53] N. Venkatachalam and R. Anitha, "A multi-feature approach to detect stegobot: a covert multimedia social network botnet," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 6079–6096, 2017.

[54] G. Liang, W. He, C. Xu, L. Chen, and J. Zeng, "Rumor identification in microblogging systems based on users' behavior," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 3, pp. 99–108, 2015.

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

[56] A. Davoudi and M. Chatterjee, "Social trust model for rating prediction in recommender systems: effects of similarity, centrality, and social ties," *Online Social Networks and Media*, vol. 7, pp. 1–11, 2018.

[57] F. Zhao, Y. Zhu, H. Jin, and L. T. Yang, "A personalized hashtag recommendation approach using lda-based topic model in microblog environment," *Future Generation Computer Systems*, vol. 65, pp. 196–206, 2016.

[58] D. Irani, S. Webb, K. Li, and C. Pu, "Modeling unintended personal-information leakage from multiple online social networks," *IEEE Internet Computing*, vol. 15, no. 3, pp. 13–19, 2011.

[59] V. V. H. Pham, S. Yu, K. Sood, and L. Cui, "Privacy issues in social networks and analysis: a comprehensive survey," *IET networks*, vol. 7, no. 2, pp. 74–84, 2017.

[60] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proceedings of the 27th annual computer security applications conference*. ACM, 2011, pp. 93–102.

[61] L. Zhao, T. Hua, C.-T. Lu, and R. Chen, "A topic-focused trust model for twitter," *Computer Communications*, vol. 76, pp. 1–11, 2016.

[62] X. Niu, G. Liu, and Q. Yang, "Trustworthy website detection based on social hyperlink network analysis," *IEEE Transactions on Network Science and Engineering*, 2018.

[63] J. Baltazar, J. Costoya, and R. Flores, "The real face of koobface: The largest web 2.0 botnet explained," *Trend Micro Research*, vol. 5, no. 9, p. 10, 2009.

[64] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Compa: Detecting compromised accounts on social networks." in *NDSS*, 2013.

[65] ——, "Towards detecting compromised accounts on social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 4, pp. 447–460, 2015.

[66] S. Ji, P. Mittal, and R. Beyah, "Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1305–1326, 2016.

[67] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. Beyah, "Seed-based de-anonymizability quantification of social networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 7, pp. 1398–1411, 2016.

[68] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Entity matching in online social networks," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 339–344.

[69] S. Torabi and K. Beznosov, "Privacy aspects of health related information sharing in online social networks," in *Presented as part of the 2013 {USENIX} Workshop on Health Information Technologies*, 2013.

[70] H. Li, H. Zhu, S. Du, X. Liang, and X. S. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 646–660, 2016.

[71] Q. Li, A. Ye, and L. Xu, "A defense mechanism against location cheating attack in social network," in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2017, pp. 549–553.

[72] Y. Ji, Y. He, X. Jiang, J. Cao, and Q. Li, "Combating the evasion mechanisms of social bots," *computers & security*, vol. 58, pp. 230–249, 2016.

[73] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna, "Poultry markets: on the underground economy of twitter followers," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 527–532, 2012.

[74] T. Amin, O. Okhiria, J. Lu, and J. An, "Facebook: A comprehensive analysis of phishing on a social system," 2010.

[75] S. Madisetty and M. S. Desarkar, “A neural network-based ensemble approach for spam detection in twitter,” *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 973–984, 2018.

[76] H. B. Kazemian and S. Ahmed, “Comparisons of machine learning techniques for detecting malicious webpages,” *Expert Systems with Applications*, vol. 42, no. 3, pp. 1166–1177, 2015.

[77] H. Gupta, M. S. Jamal, S. Madisetty, and M. S. Desarkar, “A framework for real-time spam detection in twitter,” in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2018, pp. 380–383.

[78] T. Wu, S. Liu, J. Zhang, and Y. Xiang, “Twitter spam detection based on deep learning,” in *Proceedings of the Australasian Computer Science Week Multiconference*. ACM, 2017, p. 3.

[79] 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.

[80] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, and H. Huang, “Evading anti-malware engines with deep reinforcement learning,” *IEEE Access*, vol. 7, pp. 48 867–48 879, 2019.

[81] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, “The rise of social botnets: Attacks and countermeasures,” *IEEE Transactions on Dependable and Secure Computing*, 2016.

[82] W. Wei, F. Xu, C. C. Tan, and Q. Li, “Sybildefender: A defense mechanism for sybil attacks in large social networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2492–2502, 2013.

[83] J. Golbeck and J. Hendler, “Inferring binary trust relationships in web-based social networks,” *ACM Transactions on Internet Technology (TOIT)*, vol. 6, no. 4, pp. 497–529, 2006.

[84] M. Alrubaian, M. Al-Qurishi, M. M. Hassan, and A. Alamri, “A credibility analysis system for assessing information on twitter,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 661–674, 2016.

[85] F. E. Walter, S. Battiston, and F. Schweitzer, “A model of a trust-based recommendation system on a social network,” *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 1, pp. 57–74, 2008.

[86] A. B. Waluyo, D. Taniar, W. Rahayu, A. Aikebaier, M. Takizawa, and B. Srinivasan, “Trustworthy-based efficient data broadcast model for p2p interaction in resource-constrained wireless environments,” *Journal of Computer and System Sciences*, vol. 78, no. 6, pp. 1716–1736, 2012.

[87] Z. Gong, H. Wang, W. Guo, Z. Gong, and G. Wei, "Measuring trust in social networks based on linear uncertainty theory," *Information Sciences*, vol. 508, pp. 154–172, 2020.

[88] N. Hamzelou and M. Ashtiani, "A mitigation strategy for the prevention of cascading trust failures in social networks," *Future Generation Computer Systems*, vol. 94, pp. 564–586, 2019.

[89] J. Wu, F. Chiclana, H. Fujita, and E. Herrera-Viedma, "A visual interaction consensus model for social network group decision making with trust propagation," *Knowledge-Based Systems*, vol. 122, pp. 39–50, 2017.

[90] B. Liu, Q. Zhou, R.-X. Ding, I. Palomares, and F. Herrera, "Large-scale group decision making model based on social network analysis: Trust relationship-based conflict detection and elimination," *European Journal of Operational Research*, vol. 275, no. 2, pp. 737–754, 2019.

[91] S. Tan, Y. Liu, X. Li, and Q. Dong, "A similarity-based indirect trust model with anti-spoofing capability," *Security and Communication Networks*, vol. 9, no. 18, pp. 5868–5881, 2016.

[92] Y. Cheng, J. Liu, and X. Yu, "Online social trust reinforced personalized recommendation," *Personal and Ubiquitous Computing*, vol. 20, no. 3, pp. 457–467, 2016.

[93] P. Zhou, Y. Zhou, D. Wu, and H. Jin, "Differentially private online learning for cloud-based video recommendation with multimedia big data in social networks," *IEEE transactions on multimedia*, vol. 18, no. 6, pp. 1217–1229, 2016.

[94] M. Jamali and M. Ester, "Trustwalker: a random walk model for combining trust-based and item-based recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 397–406.

[95] C.-Y. Lin, N. Cao, S. X. Liu, S. Papadimitriou, J. Sun, and X. Yan, "Smallblue: Social network analysis for expertise search and collective intelligence," in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 1483–1486.

[96] C.-W. Hang, Y. Wang, and M. P. Singh, "Operators for propagating trust and their evaluation in social networks," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 1025–1032.

[97] M. Eirinaki, M. D. Louta, and I. Varlamis, "A trust-aware system for personalized user recommendations in social networks," *IEEE transactions on systems, man, and cybernetics: systems*, vol. 44, no. 4, pp. 409–421, 2013.

[98] N. Iltaf, A. Ghafoor, and U. Zia, "A mechanism for detecting dishonest recommendation in indirect trust computation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 189, 2013.

[99] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, "Tweetcred: Real-time credibility assessment of content on twitter," in *International Conference on Social Informatics*. Springer, 2014, pp. 228–243.

[100] Z. Li, X. Zhang, H. Shen, W. Liang, and Z. He, "A semi-supervised framework for social spammer detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 177–188.

[101] W. Wang, G. Zeng, and D. Tang, "Using evidence based content trust model for spam detection," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5599–5606, 2010.

[102] B. Kang, J. O'Donovan, and T. Höllerer, "Modeling topic specific credibility on twitter," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. ACM, 2012, pp. 179–188.

[103] T. Bodnar, C. Tucker, K. Hopkinson, and S. G. Bilén, "Increasing the veracity of event detection on social media networks through user trust modeling," in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 636–643.

[104] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 675–684.

[105] M. Agarwal and B. Zhou, "Using trust model for detecting malicious activities in twitter," in *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*. Springer, 2014, pp. 207–214.

[106] Y. Ikegami, K. Kawai, Y. Namihira, and S. Tsuruta, "Topic and opinion classification based information credibility analysis on twitter," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 4676–4681.

[107] A. Gupta and P. Kumaraguru, "Credibility ranking of tweets during high impact events," in *Proceedings of the 1st workshop on privacy and security in online social media*. Acm, 2012, p. 2.

[108] K. K. Kumar and G. Geethakumari, "Detecting misinformation in online social networks using cognitive psychology," *Human-centric Computing and Information Sciences*, vol. 4, no. 1, p. 14, 2014.

[109] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman, "Sybilguard: defending against sybil attacks via social networks," *IEEE/ACM Transactions on networking*, vol. 16, no. 3, pp. 576–589, 2008.

[110] N. Z. Gong, M. Frank, and P. Mittal, "Sybilbelief: A semi-supervised learning approach for structure-based sybil detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 6, pp. 976–987, 2014.

[111] Z. Yang, J. Xue, X. Yang, X. Wang, and Y. Dai, "Votetrust: Leveraging friend invitation graph to defend against social network sybils," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 488–501, 2016.

[112] L. Shi, S. Yu, W. Lou, and Y. T. Hou, "Sybilshield: An agent-aided social network-based sybil defense among multiple communities," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 1034–1042.

[113] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 15–15.

[114] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, K. Beznosov, and H. Halawa, "Íntegro: Leveraging victim prediction for robust fake account detection in large scale osns," *Computers & Security*, vol. 61, pp. 142–168, 2016.

[115] A. Mislove, A. Post, P. Druschel, and P. K. Gummadi, "Ostra: Leveraging trust to thwart unwanted communication." in *Nsdi*, vol. 8, 2008, pp. 15–30.

[116] N. Tran, J. Li, L. Subramanian, and S. S. Chow, "Optimal sybil-resilient node admission control," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 3218–3226.

[117] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks." in *NDSS*. San Diego, CA, 2009, pp. 1–15.

[118] D. Mulamba, I. Ray, and I. Ray, "Sybilradar: A graph-structure based framework for sybil detection in on-line social networks," in *IFIP International Information Security and Privacy Conference*. Springer, 2016, pp. 179–193.

[119] Q. Cao and X. Yang, "Sybilfence: Improving social-graph-based sybil defenses with user negative feedback," *arXiv preprint arXiv:1304.3819*, 2013.

[120] S. Sedhai and A. Sun, "Semi-supervised spam detection in twitter stream," *IEEE Trans. Comput. Soc. Syst. PP*, vol. 99, 2017.

[121] C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian, "A performance evaluation of machine learning-based streaming spam tweets detection," *IEEE Transactions on Computational social systems*, vol. 2, no. 3, pp. 65–76, 2015.

[122] 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.

[123] H. Shen and Z. Li, "Leveraging social networks for effective spam filtering," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2743–2759, 2013.

[124] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[125] S. Fakhraei, J. Foulds, M. Shashanka, and L. Getoor, "Collective spammer detection in evolving multi-relational social networks," in *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2015, pp. 1769–1778.

[126] F. Wu, J. Shu, Y. Huang, and Z. Yuan, "Co-detecting social spammers and spam messages in microblogging via exploiting social contexts," *Neurocomputing*, vol. 201, pp. 51–65, 2016.

[127] I. Inuwa-Dutse, M. Liptrott, and I. Korkontzelos, "Detection of spam-posting accounts on twitter," *Neurocomputing*, vol. 315, pp. 496–511, 2018.

[128] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 2015–2028, 2019.

[129] M. Al-Janabi, E. d. Quincey, and P. Andras, "Using supervised machine learning algorithms to detect suspicious urls in online social networks," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 1104–1111.

[130] P. Dewan and P. Kumaraguru, "Towards automatic real time identification of malicious posts on facebook," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2015, pp. 85–92.

[131] M. Akiyama, T. Yagi, T. Yada, T. Mori, and Y. Kadobayashi, "Analyzing the ecosystem of malicious url redirection through longitudinal observation from honeypots," *Computers & Security*, vol. 69, pp. 155–173, 2017.

[132] M. T. Suleman and S. M. Awan, "Optimization of url-based phishing websites detection through genetic algorithms," *Automatic Control and Computer Sciences*, vol. 53, no. 4, pp. 333–341, 2019.

[133] A. Kapravelos, M. Cova, C. Kruegel, and G. Vigna, "Escape from monkey island: Evading high-interaction honeyclients," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2011, pp. 124–143.

[134] J. Cao, Q. Li, Y. Ji, Y. He, and D. Guo, "Detection of forwarding-based malicious urls in online social networks," *International Journal of Parallel Programming*, vol. 44, no. 1, pp. 163–180, 2016.

[135] L. Vu, P. Nguyen, and D. Turaga, “Firstfilter: a cost-sensitive approach to malicious url detection in large-scale enterprise networks,” *IBM Journal of Research and Development*, vol. 60, no. 4, pp. 4–1, 2016.

[136] W. Yang, W. Zuo, and B. Cui, “Detecting malicious urls via a keyword-based convolutional gated-recurrent-unit neural network,” *IEEE Access*, vol. 7, pp. 29 891–29 900, 2019.

[137] J. K. Rout, A. Dalmia, K.-K. R. Choo, S. Bakshi, and S. K. Jena, “Revisiting semi-supervised learning for online deceptive review detection,” *IEEE Access*, vol. 5, pp. 1319–1327, 2017.

[138] M. Jiang, P. Cui, and C. Faloutsos, “Suspicious behavior detection: Current trends and future directions,” *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 31–39, 2016.

[139] D. H. Fusilier, M. Montes-y Gómez, P. Rosso, and R. G. Cabrera, “Detection of opinion spam with character n-grams,” in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2015, pp. 285–294.

[140] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, “What yelp fake review filter might be doing?” in *Seventh international AAAI conference on weblogs and social media*, 2013.

[141] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, “Spotting opinion spammers using behavioral footprints,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 632–640.

[142] S. Shehnepoor, M. Salehi, R. Farahbakhsh, and N. Crespi, “Netspam: A network-based spam detection framework for reviews in online social media,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1585–1595, 2017.

[143] M. I. Ahsan, T. Nahian, A. A. Kafi, M. I. Hossain, and F. M. Shah, “Review spam detection using active learning,” in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2016, pp. 1–7.

[144] J. G. Thanikkal, M. Danish, J. G. Thanikkal, and M. Danish, “A novel approach to improve spam detection using sds algorithm,” *International Journal*, vol. 1, pp. 306–310, 2015.

[145] Y. Shao, M. Trovati, Q. Shi, O. Angelopoulou, E. Asimakopoulou, and N. Bessis, “A hybrid spam detection method based on unstructured datasets,” *Soft Computing*, vol. 21, no. 1, pp. 233–243, 2017.

[146] J. Cao, D. Jin, L. Yang, and J. Dang, “Incorporating network structure with node contents for community detection on large networks using deep learning,” *Neurocomputing*, vol. 297, pp. 71–81, 2018.

[147] J. Wu, G. Zhang, and Y. Ren, "A balanced modularity maximization link prediction model in social networks," *Information Processing & Management*, vol. 53, no. 1, pp. 295–307, 2017.

[148] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, "Scalable clustering of signed networks using balance normalized cut," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 615–624.

[149] M. Ghavipour and M. R. Meybodi, "Trust propagation algorithm based on learning automata for inferring local trust in online social networks," *Knowledge-Based Systems*, vol. 143, pp. 307–316, 2018.

[150] S. S. Singh, A. Kumar, K. Singh, and B. Biswas, "Lapso-im: A learning-based influence maximization approach for social networks," *Applied Soft Computing*, p. 105554, 2019.

[151] S. Jaradat, N. Dokoohaki, M. Matskin, and E. Ferrari, "Learning what to share in online social networks using deep reinforcement learning," in *Machine Learning Techniques for Online Social Networks*. Springer, 2018, pp. 115–133.

[152] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots," in *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 273–274.

[153] C. Freitas, F. Benevenuto, A. Veloso, and S. Ghosh, "An empirical study of socialbot infiltration strategies in the twitter social network," *Social Network Analysis and Mining*, vol. 6, no. 1, p. 23, 2016.

[154] A. B. Ashfaq, Z. Abaid, M. Ismail, M. U. Aslam, A. A. Syed, and S. A. Khayam, "Diagnosing bot infections using bayesian inference," *Journal of Computer Virology and Hacking Techniques*, pp. 1–18, 2016.

[155] N. Chavoshi, H. Hamooni, and A. Mueen, "Identifying correlated bots in twitter," in *International Conference on Social Informatics*. Springer, 2016, pp. 14–21.

[156] A. Halfaker and J. Riedl, "Bots and cyborgs: Wikipedia's immune system," *Computer*, vol. 45, no. 3, pp. 79–82, 2012.

[157] A. Mehrotra, M. Sarreddy, and S. Singh, "Detection of fake twitter followers using graph centrality measures," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2016, pp. 499–504.

[158] A. Soliman and S. Girdzijauskas, "Adagraph: adaptive graph-based algorithms for spam detection in social networks," in *International Conference on Networked Systems*. Springer, 2017, pp. 338–354.

[159] A. Alarifi, M. Alsaleh, and A. Al-Salman, "Twitter turing test: Identifying social machines," *Information Sciences*, vol. 372, pp. 332–346, 2016.

[160] C. Besel, J. Echeverria, and S. Zhou, “Full cycle analysis of a large-scale botnet attack on twitter,” in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 170–177.

[161] J. Echeverria and S. Zhou, “Discovery, retrieval, and analysis of the’star wars’ botnet in twitter,” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 1–8.

[162] A. Dorri, M. Abadi, and M. Dadfarnia, “Socialbothunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification,” in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 496–503.

[163] Q. Ma and J. Ma, “A robust method to discover influential users in social networks,” *Soft Computing*, pp. 1–13, 2017.

[164] S. M. Albladi and G. R. Weir, “User characteristics that influence judgment of social engineering attacks in social networks,” *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 5, 2018.

[165] J. Wu, Y. Sha, R. Li, Q. Liang, B. Jiang, J. Tan, and B. Wang, “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, 2017, pp. 477–489.

[166] D. Zhuang and J. M. Chang, “Enhanced peerhunter: Detecting peer-to-peer botnets through network-flow level community behavior analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1485–1500, 2018.

[167] L. Zhang, Z. Wu, and J. Cao, “Detecting spammer groups from product reviews: A partially supervised learning model,” *IEEE Access*, vol. 6, pp. 2559–2568, 2017.

[168] Z. Wang, S. Gu, X. Zhao, and X. Xu, “Graph-based review spammer group detection,” *Knowledge and Information Systems*, vol. 55, no. 3, pp. 571–597, 2018.

[169] S. Khanchi, A. Vahdat, M. I. Heywood, and A. N. Zincir-Heywood, “On botnet detection with genetic programming under streaming data label budgets and class imbalance,” *Swarm and evolutionary computation*, vol. 39, pp. 123–140, 2018.

[170] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “Key challenges in defending against malicious socialbots,” in *Presented as part of the 5th {USENIX} Workshop on Large-Scale Exploits and Emergent Threats*, 2012.

[171] D. Canali, M. Cova, G. Vigna, and C. Kruegel, “Prophiler: a fast filter for the large-scale detection of malicious web pages,” in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 197–206.

[172] C. Chen, J. Zhang, X. Chen, Y. Xiang, and W. Zhou, “6 million spam tweets: A large ground truth for timely twitter spam detection,” in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 7065–7070.

[173] P. Laskov *et al.*, “Practical evasion of a learning-based classifier: A case study,” in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 197–211.

[174] A. Yazidi, O.-C. Granmo, and B. J. Oommen, “Learning-automaton-based online discovery and tracking of spatiotemporal event patterns,” *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1118–1130, 2012.

[175] M. R. Khojasteh and M. R. Meybodi, “Evaluating learning automata as a model for cooperation in complex multi-agent domains,” in *Robot Soccer World Cup*. Springer, 2006, pp. 410–417.

[176] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, “Is the sample good enough? comparing data from twitter’s streaming api with twitter’s firehose.” in *ICWSM*, 2013.

[177] A. Neumann, J. Barnickel, and U. Meyer, “Security and privacy implications of url shortening services,” in *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2010.

[178] Online, WHOIS database, Available: http://whois.domaintools.com// (last accessed dated 24.11.2018).

[179] X. Yang, Y. Guo, and Y. Liu, “Bayesian-inference-based recommendation in online social networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 642–651, 2013.

[180] N. Abokhodair, D. Yoo, and D. W. McDonald, “Dissecting a social botnet: Growth, content and influence in twitter,” in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 839–851.

[181] W. Li and H. Song, “Art: An attack-resistant trust management scheme for securing vehicular ad hoc networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 960–969, 2015.

[182] Z. Wei, H. Tang, F. R. Yu, M. Wang, and P. Mason, “Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning,” *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4647–4658, 2014.

[183] A. Moayedikia, K.-L. Ong, Y. L. Boo, and W. G. Yeoh, “Task assignment in microtask crowdsourcing platforms using learning automata,” *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 212–225, 2018.

[184] A. Rezvanian, M. Rahmati, and M. R. Meybodi, “Sampling from complex networks using distributed learning automata,” *Physica A: Statistical Mechanics and its Applications*, vol. 396, pp. 224–234, 2014.

[185] H. Huang, X. Wei, and Y. Zhou, "Twin support vector machines: A survey," *Neurocomputing*, vol. 300, pp. 34–43, 2018.

[186] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, "An efficient hybrid multilayer perceptron neural network with grasshopper optimization," *Soft Computing*, pp. 1–18, 2018.

[187] M. C. Simmonds and J. P. Higgins, "A general framework for the use of logistic regression models in meta-analysis," *Statistical methods in medical research*, vol. 25, no. 6, pp. 2858–2877, 2016.

[188] Y. Zhou and G. Qiu, "Random forest for label ranking," *Expert Systems with Applications*, vol. 112, pp. 99–109, 2018.

[189] C. Cai, L. Li, and D. Zengi, "Behavior enhanced deep bot detection in social media," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI).* IEEE, 2017, pp. 128–130.

[190] W. Liu, L. Zhang, D. Tao, and J. Cheng, "Reinforcement online learning for emotion prediction by using physiological signals," *Pattern Recognition Letters*, 2017.

[191] T.-H. Teng, A.-H. Tan, and J. M. Zurada, "Self-organizing neural networks integrating domain knowledge and reinforcement learning," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 889–902, 2015.

[192] M. Kusy and R. Zajdel, "Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 9, pp. 2163–2175, 2015.

[193] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 464–473.

[194] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[195] M. Kaufmann and J. Kalita, "Syntactic normalization of twitter messages," in *International conference on natural language processing, Kharagpur, India*, 2010.

[196] Z. Cheng, J. Caverlee, and K. Lee, "A content-driven framework for geolocating microblog users," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 1, p. 2, 2013.

[197] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[198] V. S. Subrahmanian and D. Reforgiato, "Ava: Adjective-verb-adverb combinations for sentiment analysis," *IEEE Intelligent Systems*, vol. 23, no. 4, pp. 43–50, 2008.

[199] C. Cesarano, B. Dorr, A. Picariello, D. Reforgiato, A. Sagoff, and V. Subrahmanian, "Oasys: An opinion analysis system," in *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs (CAAW 2006)*, 2004, pp. 21–26.

[200] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System design perspective for human-level agents using deep reinforcement learning: A survey," *IEEE Access*, vol. 5, pp. 27 091–27 102, 2017.

[201] S. Alam, G. Dobbie, Y. S. Koh, and P. Riddle, "Web bots detection using particle swarm optimization based clustering," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 2955–2962.

[202] S.-H. Li, Y.-C. Kao, Z.-C. Zhang, Y.-P. Chuang, and D. C. Yen, "A network behavior-based botnet detection mechanism using pso and k-means," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 1, p. 3, 2015.

[203] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[204] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[205] H. Samma, C. P. Lim, and J. M. Saleh, "A new reinforcement learning-based memetic particle swarm optimizer," *Applied Soft Computing*, vol. 43, pp. 276–297, 2016.

[206] M. Shen and J. P. How, "Active perception in adversarial scenarios using maximum entropy deep reinforcement learning," *arXiv preprint arXiv:1902.05644*, 2019.

[207] J. S. Rosenthal, "Nash equilibria for voter models with randomly perceived positions," *Stochastic Models*, vol. 34, no. 1, pp. 98–114, 2018.

[208] Q. He, X. Wang, B. Yi, F. Mao, Y. Cai, and M. Huang, "Opinion maximization through unknown influence power in social networks under weighted voter model," *IEEE Systems Journal*, 2019.

[209] J. Jiao, K. Venkat, Y. Han, and T. Weissman, "Maximum likelihood estimation of functionals of discrete distributions," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6774–6798, 2017.

[210] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," *arXiv preprint arXiv:1512.02011*, 2015.

[211] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE transactions on cybernetics*, 2019.

[212] N. Zhou, J. Du, X. Yao, W. Cui, Z. Xue, and M. Liang, "A content search method for security topics in microblog based on deep reinforcement learning," *World Wide Web*, vol. 23, no. 1, pp. 75–101, 2020.

[213] M. Lim, A. Abdullah, and N. Jhanjhi, "Performance optimization of criminal network hidden link prediction model with deep reinforcement learning," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[214] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," *arXiv preprint arXiv:1708.04133*, 2017.

[215] D. Yuan, Y. Miao, N. Z. Gong, Z. Yang, Q. Li, D. Song, Q. Wang, and X. Liang, "Detecting fake accounts in online social networks at the time of registrations," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1423–1438.

[216] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao, "Follow the green: growth and dynamics in twitter follower markets," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 163–176.

[217] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*. IEEE, 2017, pp. 273–284.

[218] X. Zhou, N. Y. Yen, Q. Jin, and T. K. Shih, "Enriching user search experience by mining social streams with heuristic stones and associative ripples," *Multimedia Tools and Applications*, vol. 63, no. 1, pp. 129–144, 2013.

[219] F.-A. Parand, H. Rahimi, and M. Gorzin, "Combining fuzzy logic and eigenvector centrality measure in social network analysis," *Physica A: Statistical Mechanics and its Applications*, vol. 459, pp. 24–31, 2016.

[220] S. Lee, P. Yoo, T. Asyhari, Y. Jhi, L. Chermak, C. Yeun, and K. Taha, "Impact: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction," *IEEE Access*, 2020.

[221] B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak, "Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 260–268.

[222] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 137–146.

[223] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[224] H. J. Oh, E. Ozkaya, and R. LaRose, "How does online social networking enhance life satisfaction? the relationships among online supportive interaction, affect, perceived social support, sense of community, and life satisfaction," *Computers in Human Behavior*, vol. 30, pp. 69–78, 2014.

[225] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 135–142.

[226] X. Xie and Y. Li, "Trust management model of cloud computing based on multi-agent," in *Network and Information Systems for Computers (ICNISC), 2015 International Conference on*. IEEE, 2015, pp. 370–372.

[227] S. Yu, X. Lin, J. Misic, and X. S. Shen, *Networking for big data*. Chapman and Hall/CRC, 2015.

[228] C. R. Choi and H. Y. Jeong, "A broker-based quality evaluation system for service selection according to the qos preferences of users," *Information Sciences*, vol. 277, pp. 553–566, 2014.

[229] S. Ding, S. Yang, Y. Zhang, C. Liang, and C. Xia, "Combining qos prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems," *Knowledge-Based Systems*, vol. 56, pp. 216–225, 2014.

[230] H. Wang, C. Yu, L. Wang, and Q. Yu, "Effective bigdata-space service selection over trust and heterogeneous qos preferences," *IEEE Transactions on Services Computing*, 2015.

[231] Slashdot, *[Online].*, available: https://slashdot.org/ (last accessed 12.12.16).

[232] Epinions, *[Online].*, available: http://www.Epinions.com/ (last accessed 24.11.16).

[233] H.-M. Chuang, L.-C. Wang, and C.-C. Pan, "A study on the comparison between content-based and preference-based recommendation systems," in *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*. IEEE, 2008, pp. 477–480.

[234] J. Li, Z. Ning, B. Jedari, F. Xia, I. Lee, and A. Tolba, "Geo-social distance-based data dissemination for socially aware networking," *IEEE Access*, vol. 4, pp. 1444–1453, 2016.

[235] Y.-M. Li, C.-L. Chou, and L.-F. Lin, “A social recommender mechanism for location-based group commerce,” *Information Sciences*, vol. 274, pp. 125–142, 2014.

[236] B. Christianson and W. S. Harbison, “Why isn’t trust transitive?” in *International workshop on security protocols*. Springer, 1996, pp. 171–176.

[237] C. E. Shannon, “A mathematical theory of communication, part i, part ii,” *Bell Syst. Tech. J.*, vol. 27, pp. 623–656, 1948.

[238] Z. Jian-Qi, F. Feng, Y. Ke-Xin, and L. Yan-Heng, “Dynamic entropy based dos attack detection method,” *Computers & Electrical Engineering*, vol. 39, no. 7, pp. 2243–2251, 2013.

[239] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[240] *SNAP Datasets: Stanford Large Network Dataset Collection, [Online].*, available: https://snap.stanford.edu/data (last accessed 24.03.17).

List of Publications

Publications from the thesis

Journal papers:

1. G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, ”Learning automata-based trust model for user recommendations in online social networks,” *Computers & Electrical Engineering*, Elsevier, pp. 174-88, 2018.
2. G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, “Adaptive deep q-learning model for detecting social bots and influential users in online social networks,” *Applied Intelligence*, Springer, pp. 1–18, 2019.
3. R. R. Rout, G. Lingam and D. V. L. N. Somayajulu, ”Detection of Malicious Social Bots Using Learning Automata With URL Features in Twitter Network,” *IEEE Transactions on Computational Social Systems*, pp. 1004 - 1018, 2020.
4. G. Lingam, R. R. Rout, D. V. L. N. Somayajulu and S. K. Ghosh, ”Particle Swarm Optimization on Deep Reinforcement Learning for Detecting Social Spam-Bots and Spam-Influential Users in Twitter Network,” *IEEE Systems Journal*, Accepted.

Conference papers:

1. G. Lingam, R. R. Rout, D. V. L. N. Somayajulu, S. K. Das, ”Social Botnet Community Detection: A Novel Approach based on Behavioral Similarity in Twitter Network using Deep Learning,” *In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security* 2020, pp. 708-718.
2. G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, ”Social Botnet Community Detection: A Novel Approach based on Behavioral Similarity in Twitter Network using Deep Learning,” *In Proceedings of the 10th International Conference on Computing, Communication and Network Technologies*, 2019.

3. G. Lingam, R. R. Rout and D. V. L. N. Somayajulu, ”Detection of Social Botnet using a Trust Model based on Spam Content in Twitter Network”, *In Proceedings of the IEEE 13th International Conference on Industrial and Information Systems*, 2018, pp. 280-285.