Title: A Novel Framework for Internet of Knowledge Protection in Social Networking Services

Authors: Shailendra Rathore, Arun Kumar Sangaiah, Jong Hyuk Park

PII: S1877-7503(17)31072-4
DOI: https://doi.org/10.1016/j.jocs.2017.12.010
Reference: JOCS 815

To appear in:

Received date: 3-10-2017
Revised date: 14-12-2017
Accepted date: 27-12-2017

A Novel Framework for Internet of Knowledge Protection in Social Networking Services

Shailendra Rathore[1], Arun Kumar Sangaiah[2], Jong Hyuk Park[1,*]

[1]Department of Computer Science and Engineering, Seoul National University of Science and Technology, (SeoulTech) Seoul 01811, Korea

[2]School of Computing Science and Engineering, VIT University, Vellore, India

Email id: rathoreshailendra@seoultech.ac.kr, arunkumarsangaiah@gmail.com, jhpark1@seoultech.ac.kr

Highlights

- Spam activities on Social Networking Services (SNSs) are studied.
- A novel feature set is introduced to the task of spammer detection on SNSs.
- We propose a new Bagging Extreme Learning Machine approach to detect SNS spammers.

***Abstract*:** With the increasing number of users on Social Networking Service (SNS), the Internet of knowledge shared on it is also increasing. Given such enhancement of Internet of knowledge on SNS, the probability of spreading spammers on it is also increasing day by day. Several traditional machine-learning methods, such as support vector machines and naïve Bayes, have been proposed to detect spammers on SNS. Note, however, that these methods are not efficient due to some issues, such as lower generalization performance and higher training time. An Extreme Learning Machine (ELM) is an efficient classification method that can provide good generalization performance at higher training speed. Nonetheless, it suffers from overfitting and ill-posed problem that can degrade its generalization performance. In this paper, we propose a Bagging ELM-based spammer detection framework that identifies spammers in SNSs with the help of multiple ELMs that we combined using the bagging method. We constructed a labeled dataset of the two most prominent SNSs -- Twitter and Facebook -- to evaluate the performance of our framework. The evaluation results show that our framework obtained higher generalization performance rate of 99.01% for the Twitter dataset and 99.02 % for the Facebook datasets, while required a lower training time of 1.17s and 1.10s, respectively.

Keywords- Internet of knowledge; social networking services; machine learning; extreme learning machine; spammer detection; bagging method

*Corresponding Author: Jong Hyuk Park (SeoulTech, Korea)

1. Introduction

In the knowledge economy, knowledge is a primary resource. The internet is considered a power of its connectivity. It is used in the internet of things (IoT) to connect passive objects to exhibit their smart features. Similarly, the internet of knowledge is described as the connection of knowledge points originally spread over diverse places to represent them with high value. With the enhancement of the internet of knowledge, the knowledge is progressively found at the network level, such as SNSs. Recently, SNSs have been converted into an essential and prominent medium of communication and sharing knowledge. Typically, SNS users are liable for sharing knowledge in the network and are basic key elements in the network structure. The communities that consist of families, group of friends, and acquaintances are the next basic element in the network structure. In SNSs, users can share knowledge by posting links to their favorite webpages, files, photos, and videos. Furthermore, the structure of SNS communities generates a network of credibility and trust [1, 2].

Facebook and Twitter are leading SNSs. According to a report from Statista [3], the total number of Facebook and Twitter users stood at 1,968 million and 319 million, respectively, as of April 2017. With the escalating number of users, a huge amount of heterogeneous knowledge is also being produced every day on these two SNSs [4]. Mainly, multimedia knowledge (in the form of text, audio files, videos, and images) is produced, stored, and transferred in a huge amount. The multimedia knowledge posted on these SNSs is mostly accompanied by user likes, comments, tags, hashtags, and so on. Multimedia has become an essential part of SNSs. According to Zephoria Digital Marketing’s report [5], approximately 136,000 photos are shared, 293,000 statuses are updated, and 510,000 comments are posted every 60 seconds on Facebook. This report also reveals that the average content sharing rate on Facebook was 4.75 billion pieces of content per day as of May 2013, which was 94% more than the content sharing rate as of August 2012. The Statista report [6] states that Facebook and Twitter play a significant role in global content sharing activities, and that 57% and 18% of social content sharing activities occurred via Facebook and Twitter, respectively, as of the second quarter of 2016. The statistics above and our survey on SNS security [7] demonstrate that the amount of multimedia knowledge shared on Facebook and Twitter is increasing every single day. With this increase in multimedia, however, these two SNSs have become the desired target of spammers for spreading spam. A statistic from Nexgate’s report showed an estimated 355% escalation in social media spam during the first half of 2013 [8].

Spammers use numerous techniques for posting spam messages on these SNSs. The posted spam messages act as marketing advertisements and scams and help transmit malware via enclosed malicious URLs, or they are used to perform phishing attacks [9]. In addition to this, spammers can follow unknown users and send unwanted messages containing malicious URLs to obtain high exposure. The masquerade URLs can spread threats in the form of drive-by-download (install malware) and infect the host machine, allowing the installed malware to tap into the host’s confidential information [10]. Furthermore, the infected host machine may also participate in wicked botnet activities, such as operating during Distributed Denial of Service (DDOS) attacks, or become a source of email spam. Spammers may use embedded links to organize a phishing attack, wherein they target a legitimate user to collect his or her confidential information. Usually, spammers on Facebook and Twitter pose as legitimate users. Therefore, identifying and differentiating them from legitimate users make for a difficult task. In the past, spammers on these SNSs were typically simple, with a clear appearance that helped in differentiating them from legitimate users. Nevertheless, spammers can still use cheap automated methods for gaining credibility and trust and making themselves difficult to detect in the large population of SNS users.

Spammer detection in SNSs is a classification problem wherein legitimate users are distinguished from spammers based on their respective features. Note, however, that these classification problems are associated with various significant challenges when attempting to detect spammers. These challenges include the high dimensionality of features, biased and limited training sets, high computational classification complexities, and public unavailability of training sets. To overcome these challenges and carry out spammer classification effectively, several traditional machine-learning methods, such as Support Vector Machine [11, 12], Decision Tree [13], Jrip [14], Bayesian Network [15], Random Forest [16], k-Nearest Neighbors [17], etc. have been proposed. Due to the high dimensionality of features, however, finding the optimum parameters for parametric supervised algorithms is very time-consuming and difficult [18]. Therefore, existing models suffer from numerous issues, such as poor generalization performance, longer training time, and higher false positive rates. Moreover, many approaches, such as [19], use biased dataset containing a much smaller number of spam profiles than legitimate ones and provide inaccurate classification results.

Recently, the Extreme Learning Machine (ELM) [20] has been introduced as one of the effective machine learning classifiers. It is considered to be an effective and promising classifier over the other traditional classifiers, such as the Support Vector Machine and Naïve Bayesian, due to the following reasons: 1) it has better computational efficiency; 2) it provides similar or better generalization performance compared with the traditional machine learning algorithms; 3) it does not require adjusting any additional parameters except the predefined network architecture; and 4) it can use all piecewise continuous functions as activation functions, such as radial basis functions, triangular basis functions, sigmoid functions, etc. A significant number of studies from the academe show the competences of ELM in accurate classification at high learning speed [21-23]. Nevertheless, ELM has certain limitations. For instance, the arbitrary selection of input bias and weights in ELM can create the problem of ill-posed wherein the classification returns more than one solution that further results in lower generalization performance [24]. Note, however, that spammer detection in SNSs requires better generalization performance and shorter training time. In particular, shorter training time can be obtained by ELM [18], and generalization performance can be enhanced using ensemble learning [24], wherein a strong learner can be created by combining multiple weak learners [25]. It has been used with many weak learners, such as Decision Tree and Neural Networks, to create a strong learner, and it gives better generalization performance. Thus, similar to other weak learners, an ELM can be used as a weak learner, and its limitations can be overcome by employing the strategy of ensemble learning wherein a strong ELM can be created by combining multiple ELMs.

In this paper, we propose a Bagging ELM-based spammer detection framework wherein multiple ELMs are employed to distinguish spammers from legitimate users on SNSs. Bagging [26] is used as an ensemble learning method to combine multiple ELMs. The main contributions of this paper are as follows:

- We analyze spam activities on the two most popular SNSs of Facebook and Twitter and propose novel feature sets to facilitate spammer detection for both SNSs. Compared with other existing feature sets [11, 14], our proposed feature sets consist of recent and selective features that are responsible for spam on both SNSs.

- The novelty of this paper primarily lies in the fact that it offers a framework that uses a new Bagging ELM approach to detecting SNS spammers. This method provides higher generalization performance for spammer detection at shorter training time.
- Since Facebook and Twitter datasets are not publicly available, we constructed a labeled dataset of both SNSs to evaluate the performance of our framework.
- We also provide a comparative analysis of performance of our framework with other existing frameworks in order to validate the effectiveness of our framework in detecting SNS spammers.

The rest of this paper is organized as follows: Section 2 discusses various existing techniques to mitigate the issue of spam in SNSs and the ELM approach for classification. Section 3 describes our proposed framework and its components, including the features set, dataset construction, and bagging ELM. Section 4 provides an experimental evaluation of our proposed framework and our comparison of it with other existing techniques for detecting SNS spammers. Finally, we conclude our paper in Section 5.

2. Related Work

In this section, we discuss various existing machine learning models for detecting spammers in SNSs. Then, we describe our ELM approach for classification.

2.1. Machine learning for spammer detection

With the increasing number of spamming issues in SNSs, various techniques have been proposed to deal with it. A good spammer detection scheme must have capabilities that include effective detection, ability to identify a new spammer, and detection in real time. The machine-learning technique provides all these capabilities, and it has been proven to be effective in detecting spammers in SNSs. In this technique, an appropriate feature set that is responsible for spam behaviors in SNSs is selected, and an operational machine learning method is subsequently employed on the feature set to classify SNS users into two categories: spammers and legitimate users. A number of machine-learning techniques have been proposed.

In 2011, Jin et al. [27] recommended a scalable spam detection framework for SNSs. The framework extracts image content features, social network features, and text features from SNSs. The extracted features are supplied to the GAD clustering algorithm for the real-time detection of spammers. Similarly, in 2012, Gao et al. [28] proposed an online spam filtering system for SNSs that identifies spam campaigns rather than spam messages. Spam campaigns were constructed by applying URL comparison and text singling to multiple spam messages. The system's performance was evaluated by using data collected from Twitter and Facebook. In 2013, Ahmed et al. [14] proposed a generic statistical scheme to detect spammers on Facebook and Twitter. They suggested 14 generic statistical features to differentiate spam profiles from legitimate ones. They evaluated the efficacy of their suggested features on three different classification algorithms: J48, Jrip, and Naïve Bayes. In 2014, Miller et al. [19] attempted spam detection as an anomaly detection problem. They proposed 95 one-gram features of Tweets to facilitate the detection of spammers on Twitter. Two stream clustering algorithms, DenStream and StreamKM++, were modified and applied to classify Twitter users into spammers and legitimate users. In 2015, Zheng et al. [12] proposed a spammer detection model for SNSs that relied on a newly developed data collection mechanism and used Support Vector Machine for the classification task. They validated the effectiveness of their model on the Sina Weibo social network, the largest SNS in China. Zheng et al. [18] applied the ELM algorithm to reduce the detection time in detecting SNS spammers. They compared the performance of the ELM algorithm with Support Vector Machine in terms of training and testing time. In 2016, Liu et al. [16] presented a Latent Dirichlet Allocation (LDA)-based scheme to detect a smart spammer who poses as a legitimate user. The scheme used the Local Outlier Standard Score (LOSS) and Global Outlier Standard Score (GOSS), which represent local and global information, respectively, with regard to the SNS accounts. They used the three traditional machine-learning methods -- Random Forest, Adaboost, and SVM -- to evaluate further the performance of their proposed scheme. In 2017, Sohrabi et al. [29] developed an online spam filtering system that detects spam comments on Facebook. The system used the Particle Swarm Optimization algorithm for appropriate feature selection and the combination of supervised and unsupervised learning algorithms for classification task. Herzallah et al. [17] proposed a feature engineering-based framework wherein they determined the most effective features using feature engineering for spammer detection on Twitter. The features involved content-based features, user behavior features, and graph-based features. The authors used various traditional machine learning classifiers such as Support Vector Machine, Random Forest, J48, and Decision Tree to evaluate the usefulness of features in detecting spammers. Sedhai et al. [30] introduced a semi-supervised learning-based framework to detect spam tweets on Twitter. The framework identified new spam activities and retained good generalization performance. Wu et al. [31] proposed the adaptive spammer detection system that applied the sparse group modeling technique to extract additional content information of users for the identification of spammers on SNS. Some researchers have compared the efficiency of different machine-learning algorithms for spammer detection in SNSs. Recently, we proposed the SpamSpotter framework [15], which compared the eight machine-learning algorithms -- Bayesian Network, Random Forest, Decorate, Decision Tree, Jrip, k-Nearest Neighbors, Logistic Regression, and Support Vector Machine -- and proved that the Bayesian Network classifier achieved higher accuracy for spammer detection on Facebook. Similarly, Amleshwaram et al. [13] selected four machine-learning algorithms -- Decorate, Bayes Network, Random Forest, and Decision Tree -- and compared their ability to detect spammers on Twitter.

2.2. ELM for classification

The Neural Network has been widely used as the classification algorithm for identifying suspicious users and has been proven to be an efficient algorithm for spammer detection in SNSs [32]. The ELM algorithm is derived from the notion of empirical risk minimization, and it uses the theory of a single-layer feedforward network to train the Single Hidden-layer Feedforward Neural Network (SLFN) (as depicted in Figure 1) [20, 33]. A number of classification problems [21-23] have been solved by using ELM because of its excellent performance in efficiently handling non-linearity and imprecision. In this algorithm, the hidden bias and input weights (connecting the input layer to the hidden layer) are selected in an arbitrary manner, and output weights (connecting the hidden layer to the output layer) are measured with the help of the Moore-Penrose (MP) generalized inverse [34]. The ELM learning concept is explained below [33].

The given training dataset with $\mathcal{N}$ different samples and d feature dimension is denoted as $\mathcal{A} = \{a_1, a_2, \dots, a \mid a_\mathrm{i} \in \mathcal{R}^d, \mathrm{i} = 1,2,\dots,\mathcal{N}\}$, and the array of corresponding class variables for all samples in dataset $\mathcal{A}$ is represented as $\mathcal{Z} = \{\mathrm{z}_1, \mathrm{z}_2, \dots, \mathrm{z}_\mathcal{N} \mid \mathrm{z}_\mathrm{i} \in \mathcal{R}\}$. Then, a Standard SLFN with L hidden nodes and activation function $f(a)$ can be mathematically modeled as:

$$\sum_{k=1}^{L} h_k f_k(a_i) = \sum_{k=1}^{L} h_k f(w_k \cdot a_i + b_k) = z_i\,, \qquad i = 1, \dots.. \mathcal{N} \tag{1}$$

where $\mathrm{w_k}$ denotes the weight vector linking the $\mathrm{k^{th}}$ hidden node with the input nodes, h_k stands for the weight vector connecting the $\mathrm{k^{th}}$ hidden node with the output nodes, $\mathrm{z_i}$ refers to the SLFN output, and $\mathrm{b_k}$ denotes the bias value of the $\mathrm{k^{th}}$ hidden node. $f(\mathrm{w_k} \cdot a_\mathrm{i} + \mathrm{b_k})$ is the output of the $\mathrm{k^{th}}$ hidden node with respect to the input sample a_i. This estimation for $\mathcal{N}$ samples (Eq. (1)) is expressed as follows:

$$\mathcal{M} \cdot h = \mathcal{Z} \tag{2}$$

where $\mathcal{M}$ is called the hidden layer output matrix of the SLFNs and h is the output weight matrix, which can be defined as:

$$\mathcal{M} = \begin{pmatrix} f(w_1 a_1 + b_1) & \cdots & f(w_L a_1 + b_L) \\ \vdots & \cdots & \vdots \\ f(w_1 a_\mathcal{N} + b_1) & \cdots & f(w_L a_\mathcal{N} + b_L) \end{pmatrix}_{\mathcal{N} \times L} \tag{3}$$

$$h = \begin{pmatrix} h_1 \\ \vdots \\ h_L \end{pmatrix}_{L \times P} \text{ and } \mathcal{Z} = \begin{pmatrix} z_1 \\ \vdots \\ z_N \end{pmatrix}_{\mathcal{N} \times P} \tag{4}$$

In most real-time applications, including spammer detection in SNSs, the quantity of training samples is much greater than the quantity of hidden nodes ($\mathcal{N} \gg \mathrm{L}$). It denotes that $\mathcal{M}$ is a non-square matrix (non-invertible matrix), and that $\mathrm{w_k}$, $\mathrm{b_k}$, ($\mathrm{k} = 1, \dots.., \mathrm{L}$) may not exist, such that $\mathcal{M} \cdot \hbar = \mathcal{Z}$. Thus, there is an issue in finding the specific $\mathrm{w_k}$, $\mathrm{b_k}$, ($\mathrm{k} = 1, \dots.., \mathrm{L}$) and solving Eq. 2. This issue can be resolved by the research results (theory and simulations) of [33]. These results demonstrate that the parameters of the hidden layers ($\mathrm{w_k}$, $\mathrm{b_k}$, ($\mathrm{k} = 1, \dots.., \mathrm{L}$)) of SLFNs need not be tweaked at all but can be chosen arbitrarily. Training the SLFNs for the arbitrarily chosen parameters is the same as finding the unique norm least-square solution h for linear Eq. 2. The smallest, least square solution for Eq. 2 is:

$$h = \mathcal{M}^+ \cdot \mathcal{Z} \tag{5}$$

where $\mathcal{M}^{+}$ is the Moore–Penrose generalized inverse of matrix $\mathcal{M}$ [34]. As described in [33], the smallest least squares solution defined in Eq. 5 generates the smallest training errors and smallest weights. Thus, this method of training the SLFNs not only achieves the minimum square errors for the training samples but also attains the smallest weights. Therefore, it is sensible to consider that this method provides excellent generalization performance. This simple method for training SLFNs is known as ELM. The solution of the classification problem by using the ELM algorithm is described below.

Input: Training set $\mathcal{A} = \{a_1, a_2, \dots, a_{\mathcal{N}} \mid a_i \in \mathcal{R}^d, i = 1,2, \dots, \mathcal{N}\}$ with an array of class variables $\mathcal{Z} = \{z_1, z_2, \dots, z_{\mathcal{N}} \mid z_i \in \mathcal{R}\}$, activation function $f(a)$, number of hidden nodes L, and unknown or testing instances $\mathcal{A}_u = \{e_1, e_2, \dots, e_j \mid e_j \in \mathcal{R}^d\}$.

Training:

Step 1: Assign the random input weight $\mathcal{W} = [w_1, w_2, \dots., w_{\mathcal{N}}]^T$ and $\mathcal{B} = [b_1, b_2, \dots., b_{\mathcal{N}}]^T$, where T represents the matrix transpose operation.

Step 2: Compute the hidden layer output matrix $\mathcal{M} = f(\mathcal{W} \cdot \mathcal{A} + \mathcal{B})$ as defined in Eq. (3).

Step 3: Compute the output weight $h = \mathcal{M}^{+} \cdot \mathcal{Z}$ as defined in Eq. (5).

Classification:

Step 1: Compute the hidden layer output matrix of new instances of $\mathcal{A}_u$ as follows:

$$\mathcal{M}_u = f(\mathcal{W} \cdot \mathcal{A}_u + \mathcal{B}) \qquad (6)$$

Step 2: Get the class label of new instances of $\mathcal{A}_u$: $\mathcal{Z}_u = \mathcal{M}_u \cdot h$.

3. Proposed Framework

The overview of our framework is shown in Figure 2. Our framework relies on the Bagging ELM for spammer detection on Facebook and Twitter. It is composed of Feature identification, Dataset construction, and Bagging ELM. For each SNS, we identify separate feature sets that are responsible for spamming. Based on the identified feature sets, the two different datasets from Facebook and Twitter were prepared by using the dataset construction component. Each dataset contains a significant number of user profiles and their identified features. Both datasets are supplied to the Bagging ELM, which classifies user profiles into spammers and legitimate users.

3.1. Feature identification

Unlike the legitimate users on Twitter and Facebook, the objective of spammers is typically financial gain, and they degrade the system's reputation. Since legitimate users and spammers have different objectives, they show different behaviors to achieve these objectives (e.g., the two kinds of users have different interaction rate with other users). Generally, legitimate users are more active on these SNSs. They spend more time interacting with other users and doing things like posting status updates, retweeting, and replying. In order to identify this difference and separate spammers from legitimate users on Twitter and Facebook, we analyzed an enormous set of features that reveal the behaviors of users and the characteristics of the content shared by users on both SNSs. Based on this analysis and some of the other existing studies [11, 14] and our work [15], we present a novel set of features for each SNS in order to detect spammers. The feature sets consist of two types of features—account-specific features and object-specific features.

Account-specific features: This type of feature describes the specific characteristics of user behavior in terms of their social interactions, content posting rate, and impact on SNS. We recognized this type of features for Twitter and Facebook, which depict the behavior of a user on his or her account. Each feature with its reference is described in Tables 1 and 2.

Object-specific features: Objects are the unnatural part of the posted content on SNSs. For Twitter, these objects refer to hashtags, URLs, mentions, retweets, and spam words associated with the Tweet. Similarly, for Facebook, objects are tags, reposts, comments, likes, spam words, and hashtags associated with the post. Based on our analysis of the rate at which these objects are shared by spammers and legitimate users on their Twitter and Facebook accounts, we identified a significant number of object-specific features as listed in Tables 1 and 2.

3.2. Dataset construction

In this subsection, we construct a labeled dataset for both Twitter and Facebook. The dataset construction procedures for both SNSs are described below.

Twitter: In order to construct our dataset, we accumulated Twitter data using Twitter API. In this process, we collected 4,000 user profiles and about 1,800K Tweets from Twitter. Then, we extracted all of the features (listed in Table 1) of these 4,000 users by analyzing the content of their profiles and Tweets. Subsequently, we labeled the collected users as spammers if they satisfied one of the following conditions: a) shared at least one phishing or malicious link or URL; b) shared pornography or numerous links to adult websites; or c) shared a large number of advertisements or URLs that promote online shopping websites. For the first condition, we employed Google Safe Browsing to detect malicious or phishing URLs. For testing the next two conditions, we manually scanned the contents of each collected user profile. Finally, we labeled 1,465 users as spammers and 2,535 as legitimate users as shown in Table 3 and used them as a labeled dataset for Twitter in our framework.

Facebook: In order to accumulate data from Facebook, we used Facebook API and collected 4,000 user profiles and around 2,000K posts. Then, for each collected user profile, we extracted all of the features described in Table 2. As a result, we obtained an unlabeled dataset. Note, however, that we also needed a labeled dataset for our framework, so we hired 3 annotators to build this for us. They analyzed each user profile and its features and classified each user as a spammer or a legitimate user. The final decision on the labeling of each user was determined by the annotators by doing a majority voting process. As a result, 2,650 users were labeled as legitimate users, and 1,350, as spammers.

3.3. Bagging ELM

In this subsection, we propose the Bagging ELM approach to classify spammers from legitimate users in SNSs. Our proposed approach is a combination of multiple ELMs and an ensemble method called bagging. We have already described the ELM for classification in Subsection 2.2. In this subsection, we describe the Bagging method and the Bagging ELM approach.

Bagging method: The ensemble method [25] is a very efficient and prominent method that is widely used for the classification task due to its ability to improve the performance of a single classifier. It integrates the knowledge obtained from multiple learning classifiers to generate an efficient classifier. This ensemble of multiple classifiers is often called a committee classifier. There are various ensemble methods that have been proposed by many researchers, such as bagging, boosting, and stacking [25]. Numerous studies on the application of ensemble methods for the classification task, such as [36], have validated that these methods are usually more precise than a single classifier. The most commonly used ensemble methods for generating the committee classifier are boosting [37] and bagging [26], and we used bagging. Bagging uses the concept of the bootstrap sampling technique to produce diverse training subsets from the initial training set $\mathcal{A}$. The generated diverse training subsets are called bootstrap replicates. In Bagging, each bootstrap replicate is trained parallel to each other by using a single learning classifier to make the committee classifier. After building a committee classifier, the output predicted value from each individual classifier is manipulated and aggregated. Various aggregation methods have been investigated and used for manipulating and combining the prediction from more than one classifier in the classification. The most prominent and commonly used methods for aggregating the prediction from multiple classifiers are majority, weight, and average vote combining. We used the weight vote scheme because it aggregates the prediction from multiple classifiers according to the performance of each individual classifier. This means that an excellent classifier will be assigned a high value of weight, and a poorly functioning classifier will be assigned a low value of weight. Thus, better learning performance of each individual classifier will be held efficiently [38].

Bagging ELM approach: Generally, ELM significantly reduces the training time by selecting an arbitrary value of parameters (weights and biases) for hidden nodes. Nevertheless, these parameters might have values that are not optimal [24]. It may further create the ill-posed problem wherein the classification returns more than one solution for unseen (test) data and degrades the generalization performance of the ELM classifier. Therefore, to enhance the generalization performance, we introduce a novel Bagging ELM approach that uses ELM as a weak learner, bagging as an ensemble method, and weight vote as an aggregation method. The overall organization flow of the proposed framework using Bagging ELM is shown in Figure 3, where S different training subsets are produced from the initial training set using bootstrap sampling. The training of each subset is performed parallel to each other on a separate ELM that generates the S trained model of the ELM. Finally, the class variables of the given testing dataset were predicted by using the S trained model, and we applied the weight voting aggregation method for obtaining the final prediction of SNS spammers. The complete stepwise working procedure of the Bagging ELM is provided in Table 4 and explained below.

First, we describe all the inputs required for Bagging ELM. Then, the outer loop labeled as 1 begins the classification procedure to obtain the average testing results. This loop consists of four segments as described below.

Bootstrapping: This segment is responsible for creating bootstrap replicates. To create replicates initially, it divides the given dataset ($\mathcal{A}$) into training dataset ($\mathcal{A}_t$) and testing dataset ($\mathcal{A}_u$). The division is carried out according to the given ratio of $|\mathcal{A}_t|:|\mathcal{A}_u| = n_t:n_u$. It also defines $(\mathcal{A},\mathcal{Z}) = (\mathcal{A}_t,\mathcal{Z}_t) + (\mathcal{A}_u,\mathcal{Z}_u)$, where, $\mathcal{Z}_t$ and $\mathcal{Z}_u$ correspond to the array of class variables of $\mathcal{A}_t$ and $\mathcal{A}_u$, respectively. Later, the Bootstrap sampling method is applied to training dataset $\mathcal{A}_t$ for creating S different samples (training data subsets). Each sample is called a bootstrap replicate.

Training: This segment contains three loops. The first two loops find the best parameters for the single ELM by using 10-fold Cross validation. The best parameters are saved in an array of $param_{best}$. Subsequently, the third loop trains an ELM classification model for each sample $\{\mathcal{A}_{\mathrm{t}}^{(\mathrm{j})} \mid \mathrm{j} = 1,2,\dots,\mathrm{S}\}$ of training set $\mathcal{A}_{\mathrm{t}}$ (created in the Bootstrapping phase) by using $param_{best}$. Finally, this segment provides a trained model $ELM^{(j)}$ for each sample $\mathcal{A}_{\mathrm{t}}^{(\mathrm{j})}$.

Testing: This segment tests each trained model $ELM^{(j)}$ by using testing dataset $\mathcal{A}_{\mathrm{u}}$ and obtains testing result $\mathcal{Z}_{\mathrm{u}}^{(\mathrm{j})}$ for each trained model $ELM^{(j)}$.

Aggregation: This segment uses the weight voting scheme on all testing results $\mathcal{Z}_{\mathrm{u}}^{(\mathrm{j})}$ to aggregate the results and provides result $\overline{\mathcal{Z}_{\mathrm{u}}}$, which is an array of predicted class variables of testing set $\mathcal{A}_{\mathrm{u}}$. The accuracy of the result is calculated by using predicted class variables $\overline{\mathcal{Z}_{\mathrm{u}}}$ and actual class variables $\mathcal{Z}_{\mathrm{u}}$.

All of the abovementioned four segments are repeated $\mathcal{T}$ times (number of satisfied tests), and the accuracy results ($Acc^{(1)}$, $Acc^{(2)}, \dots \dots Acc^{(st)}$) are recorded for each time. Finally, the average accuracy ($\overline{Acc}$) of Bagging ELM is computed by averaging all of the accuracy results ($Acc^{(1)}, Acc^{(2)}, \dots \dots Acc^{(st)}$).

4. Experiments and Evaluation

In this section, we explain how we evaluated the performance of our proposed framework in detecting spammers in SNSs. The Bagging ELM method was employed on both Twitter and Facebook datasets as described in Subsection 3.2 to evaluate the performance of our proposed framework. The ELM, Adaboost ELM [39], and Majority Voting ELM (MV-ELM) [40] methods were also employed on the same dataset to validate the performance of Bagging ELM. The implementation and evaluation of all of the methods were run in a MATLAB 2012B environment on an Intel(R) Core(TM) i3-2328M CPU at 2.20GHz with 4GB RAM. In the evaluation stage, both Twitter and Facebook datasets were divided into two portions -- training and testing datasets at the ratio of $n_s:n_t = 0.5:0.5$ -- and the sigmoidal function $f(a) = 1/(1+e^{-\lambda a})$ was selected as an activation function. The number of hidden neurons (L) is the only parameter required for the training of a single ELM. To find the optimal value of L, we set it to 5 ~ 30 and employed 50 simulations of ELM using 10-fold cross validation on the training dataset for each value of L. Figure 4 shows the average testing accuracy across 50 simulations for each value of L. As clearly shown in Figure 4, the accuracy of ELM increases as the number of hidden neurons increases. At a certain point (L=20), however, the accuracy of ELM reaches its maximum value and starts decreasing with an increasing of number of hidden neurons. This is due to the overfitting problem wherein, with the increasing number of hidden neurons, an ELM performs very well on a training dataset but provides lower generalization performance on new data (testing dataset) and attains lower testing accuracy. ELM achieves higher percentage of accuracy when the value of L is 20 for both Twitter and Facebook. Therefore, we selected 20 as the optimal value of L.

The number of hidden neurons (L), number of samples of training dataset (S), and number of satisfied tests ($\mathcal{T}$) are the parameters in Bagging ELM, Adaboost ELM, and MV-ELM. The value of L was set to 20 in each base ELM classifier, and the value of $\mathcal{T}$ was set to 50. The value of S was set to 5, 10, 15, 25, and 30, and 50 simulations for each value of S were run. Figure 5 shows the average testing accuracy across 50 simulations for each value of S over the training dataset of Twitter. Similarly, Figure 6 depicts the average testing accuracy across 50 simulations for each value of S over the training dataset of Facebook. It can be easily seen from Figures 5 and 6 that the Bagging ELM achieves higher percentage of accuracy than MV-ELM for all values of S. For a small S, the Bagging ELM obtains lower accuracy than Adaboost ELM. Nevertheless, after a certain point (S=20), when the value of S is increased, Bagging ELM provides higher accuracy than an Adaboost ELM. After a certain point (S=20), the Adaboost ELM displays overfitting behavior, and its testing accuracy is decreased. Note, however, that the overfitting problem does not occur in our proposed Bagging ELM method, which demonstrates that Bagging ELM provides better generalization performance on new data, and that it can efficiently classify unknown users on Twitter and Facebook as being either spammers or legitimate users. When the value of S is greater than 20, Bagging ELM achieves higher accuracy. This proves that, when the number of samples is set to 25, the results of our experiments were the best. Therefore, we selected 25 as the optimal value of S and carried out further experiments with this value of S.

For $\mathrm{S} = 25$, the experimental results for all three algorithms in terms of accuracy, training time, False Positive Rate (FPR), and Standard Deviation (SD) of accuracy are summarized in Tables 5 and 6. All of the measures were averaged for the 50 simulations. We also evaluated the performance of the other conventional classifiers, including Support Vector Machine (SVM) [41], Random Forest [42], k-Nearest Neighbors (k-NN) [43], and Naive Bayes (NB) [44], in terms of all measures as shown in Tables 5 and 6. In our experiment setup, the Libsvm method was used to train the SVM classifier wherein the range of the cost parameter was {20-50} and the range of the gamma parameter was {0.1-0.9}. These parameters were optimized by employing a grid search scheme with 10-fold cross validation. Similarly, the k-parameter of the k-NN classifier was optimized using a cross validation parameter selection method.

The following observations can be made from Tables 5 and 6:

- The conventional classifiers, SVM and Random Forest, obtained higher accuracy than a single ELM. Note, however, that the ELM requires training time of 0.016s (in the case of Twitter) and 0.012s (in the case of Facebook), which is significantly shorter than the Random Forest and SVM.
- K-NN and NB generate higher FPR and lower accuracy and require more training time than a single ELM.
- The Bagging ELM has the highest accuracy rate and the lowest FPR and requires less training time than the Random Forest.

- All of the ensemble-based ELMs, including Bagging ELM, Adaboost ELM, and MV-ELM, always achieve higher accuracy, lower SD, and lower FPR than a single ELM. This demonstrates that ensemble-based ELM outperforms a single ELM.
- Bagging ELM has higher accuracy and needs less training time than Adaboost ELM. It reduces the FPR value to some extent. It also encounters a lower SD than Adaboost ELM and MV-ELM, suggesting that the Bagging ELM is the most efficient and consistent algorithm among all the ensemble-based ELM algorithms.

To validate the performance of our framework, we compared its performance results with other existing methods in terms of spammer detection on SNSs. The results were compared with regard to the machine learning algorithm, *dataset, and* obtained performance in terms of various standard evaluation measures such as FPR, accuracy, precision, recall, and F-measure. The comparison results are shown in Table 7, demonstrating that our framework obtains the lowest value of FPR and highest values of accuracy and F-measure in comparison to all other existing methods. The performance results of methods [12] and [18] slightly surpass or somewhat match those of our framework. Note, however, that these methods were evaluated over the Sina Weibo dataset, whereas Facebook and Twitter datasets were used for the evaluation in our framework.

We also compared the existing state-of-the-art approaches based on their open issues. Table 8 summarizes these existing approaches and their open issues and provides a comparison with our framework. In addition, Table 9 provides a summary of the existing approaches based on six factors: platform, biased dataset, detection, traditional machine learning classifier, proposed advanced classification algorithm, and training time. In Table 9, platform represents the SNSs used for evaluation, and biased dataset denotes whether the dataset used by the approach is biased or not.

Table 9 also describes if the approach detects spam message or spammer and uses a traditional machine learning classifier or the proposed advanced classification algorithm. It can be easily seen from Table 9 that majority of the spammer detection approaches do not discuss the training time and use the traditional machine learning classifier. Some approaches use the newly proposed classification algorithm but have issues, such as detecting spam messages rather than spammers and using a biased dataset. Note, however, that our framework relies on a newly proposed Bagging ELM algorithm and uses a baseline dataset to facilitate spammer detection on Twitter and Facebook with better generalization performance at shorter training time.

5. Conclusion

In this paper, we proposed a Bagging ELM-based spammer detection framework for SNSs. Our proposed framework has three major contributions in this area. First, it identifies account- and object-specific features to facilitate spammer detection in SNSs. Second, it constructs a novel dataset of the two most popular SNSs, i.e., Twitter and Facebook. Finally, it introduces a Bagging ELM classifier and applies this classifier to the dataset that we constructed from Twitter and Facebook. Our experiments and comparison with other traditional classifiers show that our framework is able to achieve much better generalization performance than other existing frameworks. Our framework achieved average accuracy rate of 99.01 % for the Twitter dataset and 99.02 % for the Facebook dataset while requiring shorter training time of 1.17s and 1.10s, respectively. Note, however, that the performance result of the framework relies on the labeled dataset, which typically needs considerable labor cost and time. Furthermore, manually labeling the process to obtain the labeled dataset suffers from inaccurate result due to individual bias. To address the issue of labeled dataset, our framework can be enhanced by using semi-supervised learning with ELM. The semi-supervised ELM can support the use of easily acquired unlabeled dataset and provide good generalization performance at higher speed.

Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2014-0-00720-002) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

1. D. H. Lee, Personalizing Information Using Users' Online Social Networks: A Case Study of CiteULike, J. Inf. Process. Syst. 11 (1) (2015) 1-21
2. J. Singh, G. Singh, R. Singh, Optimization of sentiment analysis using machine learning classifiers, Human-centric Computing and Information Sciences. (2017). https://doi.org/10.1186/s13673-017-0116-3
3. Statista, Most famous social network sites worldwide as of April 2017, ranked by number of active users (in millions). https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (accessed 13 September 2017)
4. P. Sharma, S. Rathore, J. H. Park, Multilevel learning based modeling for link prediction and users' consumption preference in Online Social Networks, Future Generation Computer Systems. (2017). https://doi.org/10.1016/j.future.2017.08.031
5. Zephoria Digital Marketing, The Top 20 Valuable Facebook Statistics – Updated May 2016. https://zephoria.com/top-15-valuable-facebook-statistics/ (accessed 13 September 2017)
6. Statista, Distribution of global social content sharing activities as of 2nd quarter 2016, by social network, https://www.statista.com/statistics/283889/content-sharing-primary-social-networks-worldwide/ (accessed 12 September 2017)
7. S. Rathore, P.K. Sharma, V. Loia, Y. S. Jeong, J. H. Park, Social Network Security: Issues, Challenges, Threats, and Solutions. Information Sciences. 421(2017) 43-69
8. Nexgate, Research Report 2013 State of Social Media Spam. http://nexgate.com/wp-content/uploads/2013/09/Nexgate-2013-State-of-Social-Media-Spam-Research-Report.pdf (accessed 12 September 2017)
9. H. Xu, W. Sun, A. Javaid, Efficient spam detection across Online Social Networks, in: International Conference on Big Data Analysis (ICBDA), IEEE, 2016, pp. 1-6
10. S. Rathore, P.K. Sharma, J. H. Park, XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs, J. Inf. Process. Syst. 13 (4) (2017) 1014-1028
11. F. Benevenuto, G. Magno, T. Rodrigues, V. Almeida, Detecting spammers on twitter. in: Collaboration, electronic messaging, anti-abuse and spam conference (CEAS), 2010, pp. 12-21
12. X. Zheng, Z. Zeng, Z. Chen, Y. Yu, C. Rong, Detecting spammers on social networks, Neurocomputing. 159 (2015) 27-34
13. A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu, C Yang, Cats: Characterizing automation of twitter spammers, in: Fifth International Conference on Communication Systems and Networks (COMSNETS), IEEE, 2013, pp. 1-10
14. F. Ahmed, M. Abulaish, A generic statistical approach for spam detection in Online Social Networks, Comput. Commun. 36 (10) (2013) 1120-1129
15. S. Rathore, V. Loia, J. H. Park, SpamSpotter: An Efficient Spammer Detection Framework based on Intelligent Decision Support System on Facebook. Applied Soft Computing. (2017) https://doi.org/10.1016/j.asoc.2017.09.032
16. L. Liu, Y. Lu, Y. Luo, R. Zhang, L. Itti, J. Lu, Detecting" Smart" Spammers On Social Network: A Topic Model Approach, arXiv preprint arXiv:1604.08504 (2016) 1-6
17. W. Herzallah, H. Faris, O. Adwan, Feature engineering for detecting spammers on Twitter: Modelling and analysis, Journal of Information Science. (2017). https://doi.org/10.1177/0165551516684296
18. X. Zheng, X. Zhang, Y. Yu, T. Kechadi, C. Rong, ELM-based spammer detection in social networks, The Journal of Supercomputing 72 (8) (2016) 2991-3005

19. Z. Miller, B. Dickinson, W. Deitrick, W. Hu, A. H. Wang, Twitter spammer detection using data stream clustering. Information Sciences. 260 (2014) 64-73
20. G. B. Huang, Q. Y. Zhu, C. K Siew, Extreme learning machine: theory and applications. Neurocomputing. 70 (2006) 489-501
21. W. Li, C. Chen, H. Su, Q. Du, Local binary patterns and extreme learning machine for hyperspectral imagery classification, IEEE Transactions on Geoscience and Remote Sensing. 53 (7) (2015) 3681-3693
22. J. Luo, C. M. Vong, P.K. Wong, Sparse Bayesian extreme learning machine for multi-classification, IEEE Transactions on Neural Networks and Learning Systems. 25 (4) (2014) 836-843
23. X. Li, W. Mao, W. Jiang, Extreme learning machine based transfer learning for data classification. Neurocomputing, 174 (A) (2016) 203-210
24. J. J. Cao, S. Kwong, R. Wang, K. Li, A weighted voting method using minimum square error based on Extreme Learning Machine, in: International Conference on Machine Learning and Cybernetics (ICMLC), IEEE, 2012, pp. 411-414
25. D. W. Opitz, R. Maclin, Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research. 11 (1999) 169-198
26. L. Breiman, Bagging predictors, Machine learning. 24 (2) (1996) 123-140
27. X. Jin, C. Lin, J. Luo, J. Han, A data mining-based spam detection system for social media networks, in: Proceedings of the VLDB Endowment, Vol. 4, 2011, pp.1458-1461
28. H. Gao, Y. Chen, K. Lee, D. Palsetia, A. N. Choudhary, Towards Online Spam Filtering in Social Networks, in: NDSS, vol. 12, 2012, pp. 1-16
29. M. K. Sohrabi, F. Karimi, A Feature Selection Approach to Detect Spam in the Facebook Social Network, Arabian Journal for Science and Engineering. (2017). https://doi.org/10.1007/s13369-017-2855-x
30. S. Sedhai, A. Sun, Semi-Supervised Spam Detection in Twitter Stream, arXiv preprint arXiv:1702.01032 (2017) 1-9
31. L. Wu, X. Hu, F. Morstatter, H. Liu, Adaptive Spammer Detection with Sparse Group Modeling, in: ICWSM, 2017, pp. 319-326
32. T. Wu, S. Liu, J. Zhang, Y. Xiang, Twitter spam detection based on deep learning, in: Proceedings of the Australasian Computer Science Week Multiconference, ACM, 2017, pp. 3-11
33. G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: International Joint Conference on Neural Networks, IEEE, 2004, vol. 2, pp. 985-990
34. S. Denis, Matrices: Theory and Applications, Springer-Verlag New York, New York , 2002
35. G. Stringhini, C. Kruegel, G. Vigna, Detecting spammers on social networks, in: Proceedings of the 26th Annual Computer Security Applications Conference, ACM, 2010, pp. 1-9
36. W. Budgaga, M. Malensek, S. Pallickara, N. Harvey, F. J. Breidt, S. Pallickara, Predictive analytics using statistical, learning, and ensemble methods to support real-time exploration of discrete event simulations, Future Generation Computer Systems. 56 (2016) 360-374
37. Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: Icml, 1996, vol. 96, pp. 148-156
38. L. I. Kuncheva, J. J. Rodríguez, A weighted voting framework for classifiers ensembles, Knowledge and Information Systems. 38 (2) (2014) 259-275
39. Y. Jiang, Y. Shen, Y. Liu, W. Liu, Multiclass AdaBoost ELM and its application in LBP based face recognition, Mathematical Problems in Engineering, 2015 (2015) 1-9
40. J. Cao, Z. Lin, G. B. Huang, N. Liu, Voting based extreme learning machine, Information Sciences, 185 (1) (2012) 66-77

41. C. C. Chang, C. J. Lin, Libsvm: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST). 2 (3) (2011) 27
42. A. Liaw, M. Wiener, Classification and regression by randomForest, R news. 2 (3) (2002) 18-22
43. L. E. Peterson, K-nearest neighbor, Scholarpedia, 4 (2) (2009) 1883
44. I. Rish, An empirical study of the naive bayes classier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, IBM New York, 2001, vol. 3, pp. 41-46

Author Biographies

Shailendra Rathore He is a PhD student in the Department of Computer Science at Seoul National University of Science and Technology (SeoulTech.), Seoul, South Korea. Currently, he is working in Ubiquitous Computing Security (UCS) Lab under the supervision of Prof. Jong Hyuk Park. His broadly research interest includes Information and Cyber Security, Artificial Intelligence, Social Networking Service Security, IoT. Previous to joining PhD at SeoulTech, he received his M.E. in Information Security from Thapar University, Patiala, India. He is also reviewer of IEEE System Journal and IEEE Transactions on Industrial Informatics.

Dr.Arun Kumar Sangaiah has received his Master of Engineering (ME) degree in Computer Science and Engineering from the Government College of Engineering, Tirunelveli, Anna University, India. He had received his Doctor of Philosophy (PhD) degree in Computer Science and Engineering from the VIT University, Vellore, India. He is presently working as an Associate Professor in School of Computer Science and Engineering, VIT University, India. His area of interest includes software engineering, computational intelligence, wireless networks, bio-informatics, and embedded systems. He has authored more than 100 publications in different journals and conference of national and international repute. His current research work includes global software development, wireless ad hoc and sensor networks, machine learning, cognitive networks and advances in mobile computing and communications. Also, he was registered a one Indian patent in the area of Computational Intelligence. Besides, Prof. Sangaiah is responsible for Editorial Board Member/Associate Editor of various international journals.

Dr. James J. (Jong Hyuk) Park received Ph.D. degrees in Graduate School of Information Security from Korea University, Korea and Graduate School of Human Sciences from Waseda University, Japan. From December, 2002 to July, 2007, Dr. Park had been a research scientist of R&D Institute, Hanwha S&C Co., Ltd., Korea. From September, 2007 to August, 2009, He had been a professor at the Department of Computer Science and Engineering, Kyungnam University, Korea. He is now a professor at the Department of Computer Science and Engineering and Department of Interdisciplinary Bio IT Materials, Seoul National University of Science and Technology (SeoulTech), Korea. Dr. Park has published about 200 research papers in international journals and conferences. He has been serving as chair, program committee, or organizing committee chair for many international conferences and workshops. He is a steering chair of international conferences – MUE, FutureTech, CSA, CUTE, UCAWSN, World IT Congress-Jeju. He is editor-in-chief of Human-centric Computing and Information Sciences (HCIS) by Springer, The Journal of Information Processing Systems (JIPS) by KIPS, and Journal of Convergence (JoC) by KIPS CSWRG. He is Associate Editor / Editor of 14 international journals including JoS, JNCA, SCN, CJ, and so on. In addition, he has been serving as a Guest Editor for international journals by some publishers: Springer, Elsevier, John Wiley, Oxford Univ. press, Emerald, Inderscience, MDPI. He got the best paper awards from ISA-08 and ITCS-11 conferences and the outstanding leadership awards from IEEE HPCC-09, ICA3PP-10, IEE ISPA-11, PDCAT-11, IEEE AINA-15. Furthermore, he got the outstanding research awards from the SeoulTech, 2014. His research interests include IoT, Human-centric Ubiquitous Computing, Information Security, Digital Forensics, Vehicular Cloud Computing, Multimedia Computing, etc. He is a member of the IEEE, IEEE Computer Society, KIPS, and KMMS.

Author Photos

Shailendra Rathore

Dr.Arun Kumar Sangaiah

Dr. Jong Hyuk Park

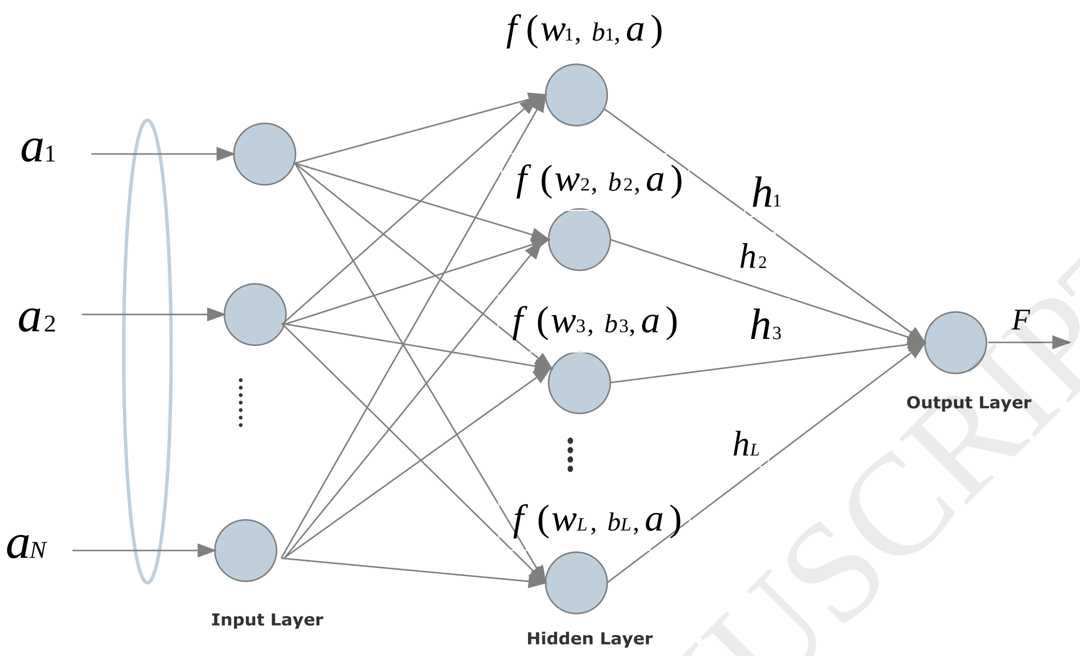

Fig. 1. Structure of a SLFNs

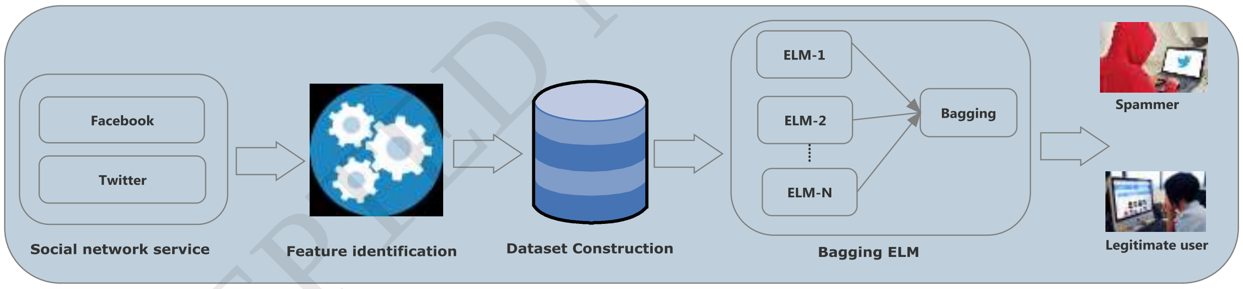

Fig. 2. Architecture of proposed framework

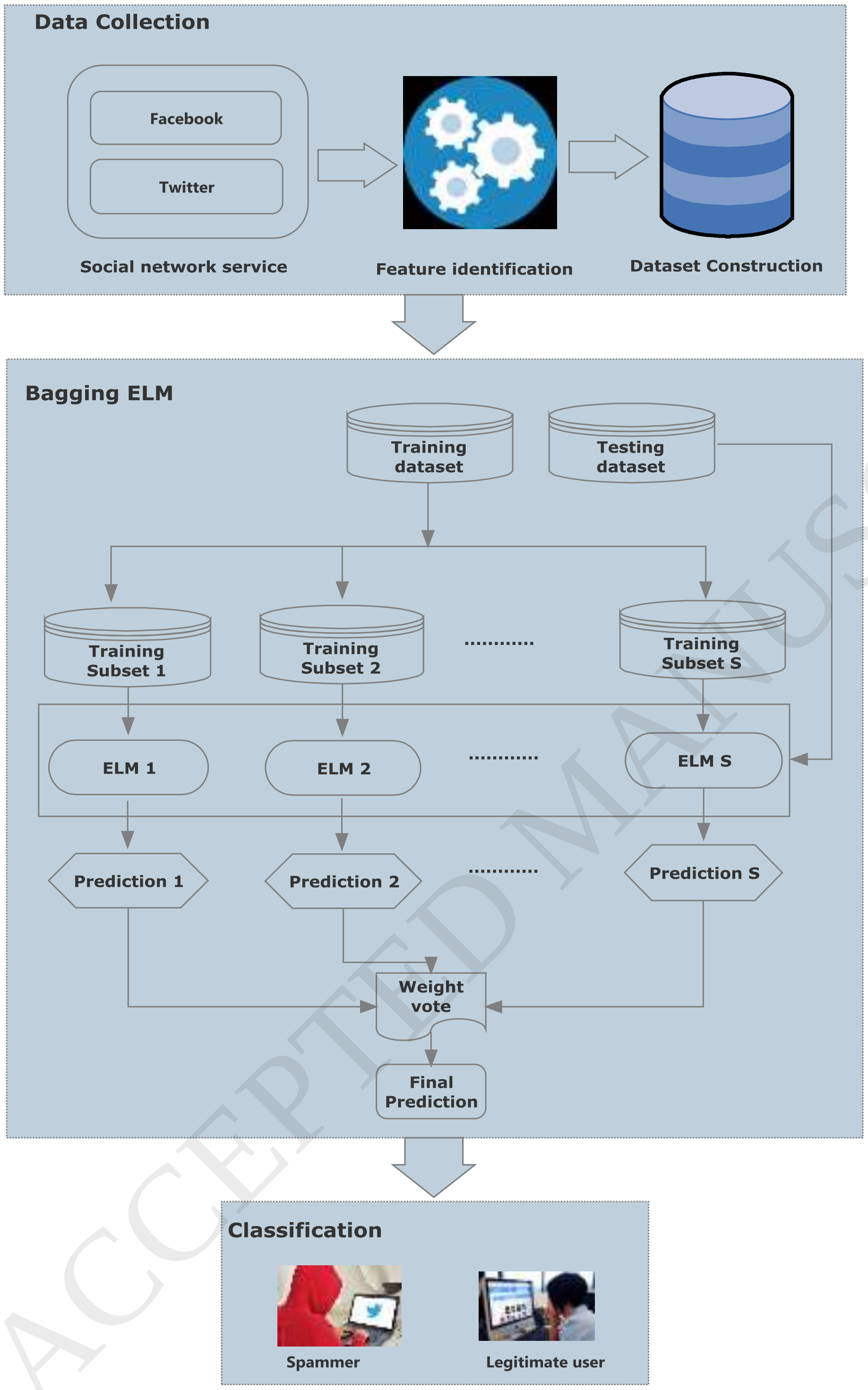

Fig. 3. Overall organization flow of the proposed framework using Bagging ELM

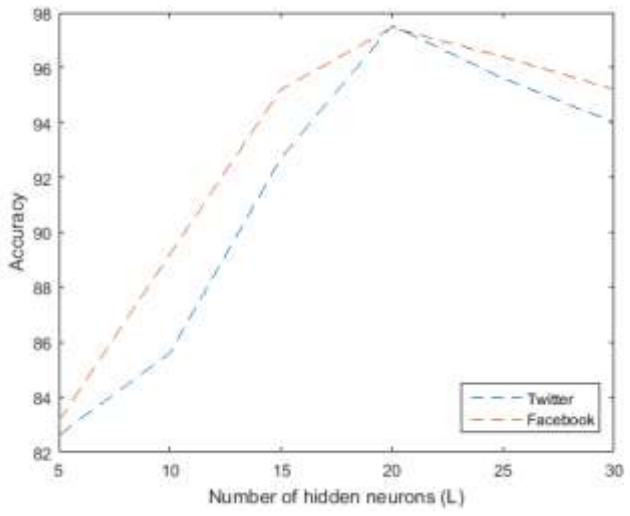

Fig. 4. Accuracy V/s Number of hidden neurons (L)

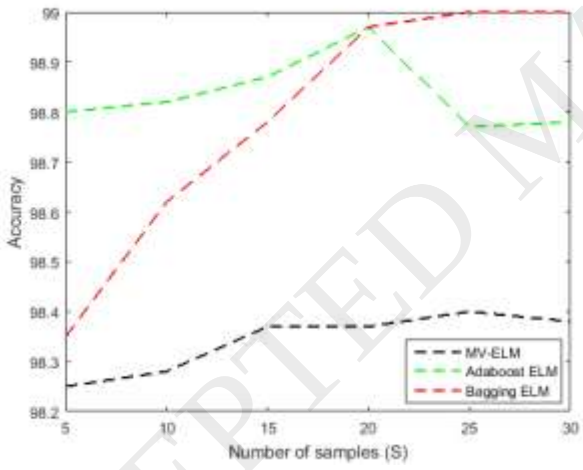

Fig. 5. Accuracy V/s Number of samples (S) for Twitter dataset

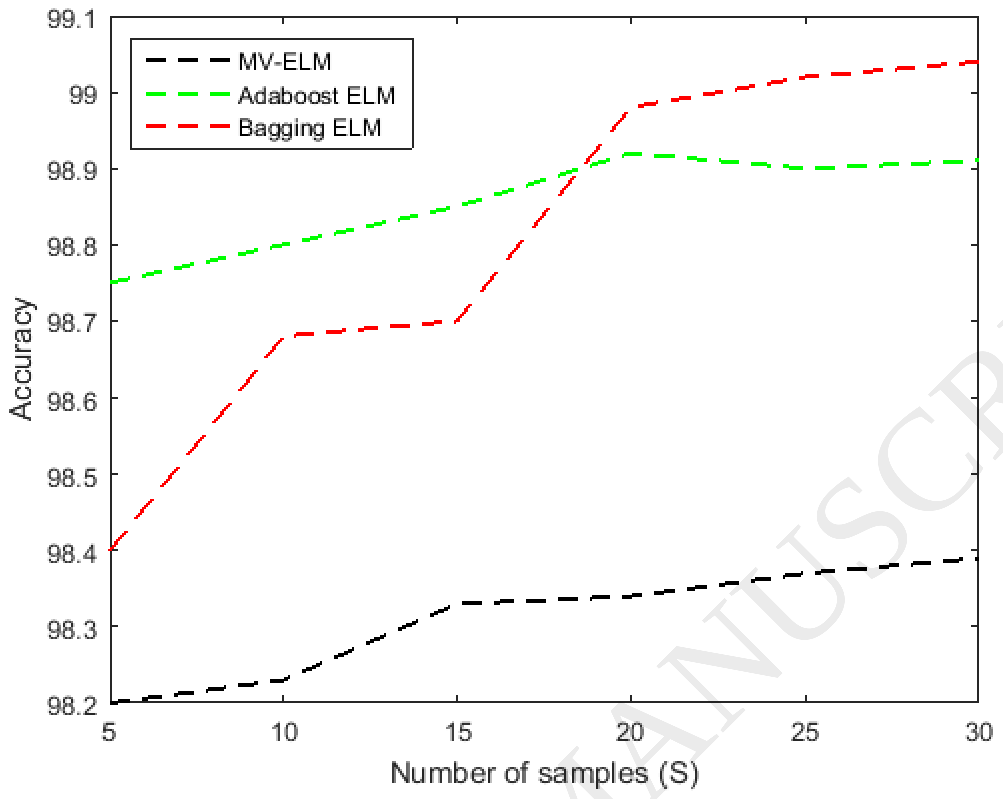

Fig. 6. Accuracy V/s Number of samples (S) for Facebook dataset

Table 1

Set of features for Twitter

No.	Account-specific feature	Reference	Object-specific feature	Reference
1.	Length of the account name: $t_1^{(a)}$	New	Average count of hashtags present in a tweet: $t_1^{(c)}$	[11, 14]
2.	Length (size) of the account description: $t_2^{(a)}$	New	Maximum count of hashtags present in a tweet: $t_2^{(c)}$	[11, 14]
3.	Total count of friends: $t_3^{(a)}$	New	Fraction of tweets with hashtag: $t_3^{(c)}$	New
4.	Total count of followers: $t_4^{(a)}$	[11]	Average count of URLs per tweet: $t_4^{(c)}$	[11]
5.	User reputation: $t_5^{(a)} = t_4^{(a)}/(t_3^{(a)} + t_4^{(a)})$	New	Maximum count of URLs present in a tweet: $t_5^{(c)}$	[11]
6.	Ratio of followers and friends: $t_6^{(a)} = t_4^{(a)}/t_3^{(a)}$	New	Fraction of tweets with URLs: $t_6^{(c)}$	[11]
7.	Life time of the user account (in days): $t_7^{(a)}$	[14]	Average count of mentions per tweet: $t_7^{(c)}$	[11]
8.	Rate of friends: $t_8^{(a)} = t_3^{(a)}/t_7^{(a)}$	New	Maximum count of mentions per tweet: $t_8^{(c)}$	[11, 14]
9.	Rate of followers: $t_9^{(a)} = t_4^{(a)}/t_7^{(a)}$	New	Fraction of tweets with mention: $t_9^{(c)}$	New
10.	Total count of tweets posted: $t_{10}^{(a)}$	[11, 14]	Average count of retweets per tweet: $t_{10}^{(c)}$	[11]
11.	Average count of tweets posted per day: $t_{11}^{(a)} = t_{10}^{(a)}\ /t_7^{(a)}$	[11]	Maximum count of retweets per tweet: $t_{11}^{(c)}$	[11]
12.	Average count of tweets generated per week: $t_{12}^{(a)} = t_{10}^{(a)}\ /7t_7^{(a)}$	New	Average count of favorites per tweet: $t_{12}^{(c)}$	New
13.	Total count of tweets liked/favorited: $t_{13}^{(a)}$	New		
14.	Total count of lists: $t_{14}^{(a)}$	New		

Table 2
Set of features for Facebook

No.	Account-based Features	References	Object specific features	References
1.	Total count of friends: $f_1^{(a)}$	[14, 15, 35]	Average count of hashtags per post: $f_1^{(c)}$	[15]
2.	Total count of followings: $f_2^{(a)}$	[15]	Maximum count of hashtags per post: $f_2^{(c)}$	New
3.	User reputation: $f_3^{(a)} = f_2^{(a)}/(f_1^{(a)} + f_2^{(a)})$	New	Fraction of posts with hashtags: $f_3^{(c)}$	New
4.	Ratio of followings and friends: $f_4^{(a)} = f_2^{(a)}/f_1^{(a)}$	New	Average count of occurrence of URLs per post: $f_4^{(c)}$	New
5.	Life time of the user account (in days): $f_5^{(a)}$	New	Maximum count of URLs present in a post: $f_5^{(c)}$	New
6.	Rate of friends: $f_6^{(a)} = f_1^{(a)}/f_5^{(a)}$	New	Fraction of posts with URLs: $f_6^{(c)}$	[15]
7.	Rate of followings: $f_7^{(a)} = f_2^{(a)}/f_5^{(a)}$	New	Average count of tags per post: $f_7^{(c)}$	[14, 15]
8.	Total count of posts shared: $f_8^{(a)}$	[14, 15, 35]	Maximum count of tags per post: $f_8^{(c)}$	New
9.	Average count of posts shared in a day: $f_9^{(a)} = f_8^{(a)}/f_5^{(a)}$	[15]	Fraction of posts with tags: $f_9^{(c)}$	New
10.	Average count of posts shared per week: $f_{10}^{(a)} = 7f_8^{(a)}/f_5^{(a)}$	New	Average count of reposts per post: $f_{10}^{(c)}$	New
11.	Total count of community in which user have participated: $f_{11}^{(a)}$	[14, 15]	Maximum number of reposts per post: $f_{11}^{(c)}$	New
12.			Maximum count of comments in a post : $f_{12}^{(c)}$	New
13.			Maximum count of likes in a post: $f_{13}^{(c)}$	New
14.			Average number of spam words per post: $f_{14}^{(c)}$	New
15.			Maximum number of spam words per post: $f_{15}^{(c)}$	New

Table 3
Twitter and Facebook dataset

SNS	Label	Number of users
Twitter	Spammer	1465
	Legitimate user	2535
Facebook	Spammer	1350
	Legitimate user	2650

Table 4

Bagging ELM algorithm

Inputs:

$\mathcal{A}$: Dataset,

$\mathcal{Z}$: Array of class variables of $\mathcal{A}$,

$\mathcal{T}$: Number of satisfied testing,

S: Number of different samples,

$param$: Array of parameters of classifier.

Output:

$\overline{Acc}$: Average accuracy of Bagging ELM.

```
(1) for $st = 1$ to $\mathcal{T}$, do
    Arbitrary PARTITION $(\mathcal{A}, \mathcal{Z}) = (\mathcal{A}_t, \mathcal{Z}_t) + (\mathcal{A}_u, \mathcal{Z}_u)$,
    where $|\mathcal{A}_t| : |\mathcal{A}_u| = n_t : n_u$;
    BOOTSTRAP $(\mathcal{A}_t, \mathcal{Z}_t) \Rightarrow (\mathcal{A}_t^{(j)}, \mathcal{Z}_t^{(j)})$, $j = 1, \ldots, S$;        Bootstrapping

    for each parameter in $param$, do
        for each fold in 10-fold Cross validation, do
                $ELM = \boldsymbol{train}(\mathcal{A}_t, \mathcal{Z}_t, parameter)$;
        end
    end
    Save the best-performed parameters in $param_{best}$ ;        Training
    for $j = 1$ to $S$ , do
        $ELM^{(j)} = \boldsymbol{train}(\mathcal{A}_t^{(j)}, \mathcal{Z}_t^{(j)}, param_{best})$
    end

    for $j = 1$ to $S$ , do
                $\mathcal{Z}_u^{(j)} = test(ELM^{(j)}, \mathcal{A}_u, \mathcal{Z}_u)$;        Testing
    end

    $\boldsymbol{Weight\ vote}$ $\mathcal{Z}_u^{(j)} \Rightarrow \overline{\mathcal{Z}_u}$, $j = 1, \ldots, S$;
                                                                  Aggregation
                $Acc^{(st)} = \frac{|\overline{\overline{\mathcal{Z}_u} \oplus \mathcal{Z}_u}|}{|\mathcal{Z}_u|}$ ;
end

$\overline{Acc} = (Acc^{(1)} + Acc^{(2)}, \ldots, Acc^{(st)})/st$ ;
return $\overline{Acc}$;
```

Table 5

Results of different classifiers for Twitter dataset

Classifier	Training time	Accuracy	FPR	SD
Bagging ELM	1.17	99.01	0.010	0.18
Adaboost ELM	1.22	98.75	0.095	0.88
SVM	0.40	94.75	0.135	-
MV-ELM	1.02	98.36	0.067	0.26
Random Forest	2.67	94.40	0.089	0.08
ELM	0.016	93.53	0.096	0.98
k-NN	0.34	93.04	0.138	-
NB	0.073	91.53	0.155	-

Table 6

Results of different classifiers for Facebook dataset

Classifier	Training time	Accuracy	FPR	SD
Bagging ELM	1.10	99.02	0.029	0.10
Adaboost ELM	1.19	98.87	0.066	0.70
SVM	0.50	94.69	0.147	-
MV-ELM	0.99	94.37	0.057	0.51
Random Forest	2.46	94.56	0.076	0.10
ELM	0.012	93.45	0.085	0.87
k-NN	0.41	92.98	0.151	-
NB	0.064	91.01	0.157	-

Table 7

Performance comparison of proposed framework with the existing methods for spammer detection in SNSs

Method	Dataset	Classifier	FPR	Accuracy	Precision	Recall	F-measure
Stringhini et al. [35]	Facebook, Twitter, MySpace	Random Forest	0.20	-	-	-	-
Benevenuto et al. [11]	Twitter	SVM	0.036	0.876	-	-	-
Ahmed et al. [14]	Facebook, Twitter	Jrip	0.014	0.987	-	-	-
Miler et al. [19]	Twitter	Combination of StreamKM++ and DenStream	0.0217	0.980	0.793	1	0.885
Zheng et al. [12]	Sina Weibo	SVM	-	-	0.999	0.991	0.995
Zheng et al. [18]	Sina Weibo	ELM	-	-	0.999	0.990	0.995
Liu et al. [16]	Twitter, Sina Weibo	Random Forest	-	-	0.988	0.958	0.978
Rathore et al. [15]	Facebook	Bayesian Network		0.984	0.987	0.980	0.984
Wu et al. [31]	Twitter	Sparse Group Modeling	-	-	0.937	0.969	0.953
Proposed Framework	Facebook, Twitter	Bagging ELM	0.010	0.990	0.998	0.996	0.997

Table 8

Summary of the existing approaches, their open issues, and comparison with proposed framework

Existing approach	Open issue	Proposed framework
Jin et al. [27], Gao et al. [28], Sedhai et al. [30]	These systems detect spam messages rather than spammers in SNSs.	Our framework provides the detection of SNS spammers by utilizing significant and effectives features that are responsible for spam behaviors in SNSs.
Ahmed et al. [14], Liu et al. [16], Rathore et al. [15], Amleshwaram et al. [13], Zheng et al. [12], Herzallah et al. [17]	These approaches use traditional machine learning classifiers, which suffer from numerous issues, such as poor generalization performance, longer training time, and higher false positive rates.	Our framework relies on an advance machine learning classifier called Bagging ELM that provides good generalization performance at a faster training speed.
Miller et al. [19]	The proposed method employed over an old and biased dataset that contains very less quantity of spam profiles than legitimate ones. The biased dataset may result in inaccurate detection of SNS spammer.	Our framework constructs a baseline dataset of Twitter and Facebook that is updated and contains reasonable amount of both spam and legitimate profiles.

Table 9

Comparison of proposed work with various recent approaches based on existing metrics

Existing approach	Platform	Biased dataset	Detection	Traditional machine learning classifier	Proposed advanced classification algorithm	Training time
Jin et al. [27]	Facebook	No	Spam message	No	Yes	-
Gao et al. [28]	Facebook, Twitter	No	Spam message	No	Yes	Low latency
Ahmed et al. [14]	Facebook, Twitter	No	Spammer	Yes	No	-
Miller et al. [19]	Twitter	Yes	Spammer	No	Yes	-
Liu et al. [16]	Twitter, Sina Weibo	No	Spammer	Yes	No	-
Rathore et al. [15]	Facebook	No	Spammer	Yes	No	-
Amleshwaram et al. [13]	Twitter	No	Spammer	Yes	No	Low latency
Zheng et al. [12]	Sina Weibo	No	Spammer	Yes	No	-
Zheng et al. [18]	Sina Weibo	No	Spammer	No	Yes	Lower
Sohrabi et al. [29]	Facebook	No	Spam message	No	Yes	-
Herzallah et al. [17]	Twitter	No	Spammer	Yes	No	-
Sedhai et al. [30]	Twitter	No	Spam tweet	No	Yes	-
Wu et al. [31]	Twitter	Yes	Spammer	No	Yes	Lower
Proposed work	Facebook, Twitter	No	Spammer	No	Yes	Lower