A Neural Network-Based Ensemble Approach for Spam Detection in Twitter

Sreekanth Madisetty and Maunendra Sankar Desarkar

***Abstract*—As the social networking sites get more popular, spammers target these sites to spread spam posts. Twitter is one of the most popular online social networking sites where users communicate and interact on various topics. Most of the current spam filtering methods in Twitter focus on detecting the spammers and blocking them. However, spammers can create a new account and start posting new spam tweets again. So there is a need for robust spam detection techniques to detect the spam at tweet level. These types of techniques can prevent the spam in real time. To detect the spam at tweet level, often features are defined, and appropriate machine learning algorithms are applied in the literature. Recently, deep learning methods are showing fruitful results on several natural language processing tasks. We want to use the potential benefits of these two types of methods for our problem. Toward this, we propose an ensemble approach for spam detection at tweet level. We develop various deep learning models based on convolutional neural networks (CNNs). Five CNNs and one feature-based model are used in the ensemble. Each CNN uses different word embeddings (Glove, Word2vec) to train the model. The feature-based model uses content-based, user-based, and n-gram features. Our approach combines both deep learning and traditional feature-based models using a multilayer neural network which acts as a meta-classifier. We evaluate our method on two data sets, one data set is balanced, and another one is imbalanced. The experimental results show that our proposed method outperforms the existing methods.**

***Index Terms*—Classification, social media, spam detection, Twitter.**

I. Introduction

NOWADAYS, online social networks play a very important role in spreading information over the world. Users in social networking sites express opinions and engage in discussing different topics. Among these social networks, Twitter is one of the most popular social network microblogging platforms. In Twitter, there are 317 million monthly active users, 500 million tweets are sent per day. Twitter supports more than 35 languages.[1] Unfortunately, spammers are also active on Twitter for their personal or organizational gains. Spammers use various techniques to spread the spam such as posting malicious links, sending unsolicited messages to legitimate users, aggressive following behavior to get attention, abusing the reply or mentions to post unwanted messages, creating multiple accounts which can be created either manually or with some automated tools, repeatedly posting duplicate updates, posting URLs with unrelated content, and hijacking trending topics to grab attention. Spam tweets often contain URL links with unrelated content, adult related content, and monetary claim content. Spammers make use of URLs in the tweets to redirect the users to malicious sites which contain virus in those sites. They also use URLs for phishing and get the personal details of the users.

TABLE I

Examples of Spam and Non-Spam Tweets. These Are Taken From the Data sets Described in Our Paper

Id	Text	Category
1000	RT @username1: @username2 RETWEET IF YOU WANT MORE FOLLOWERS — #TeamFollowBack — #InstantFollowBack — #TeamFollow — #500ADay — #RT	spam
1001	I've collected 13,615 gold coins! http://t.co/VJZzqMT4Sv #android, #androidgames, #gameinsight	spam
1002	RT @username1: EVERYONE GO AND FOLLOW THIS ACCOUNT NOW FOR GREAT TWEETS AND SHE FOLLOWS BACK – @username2 :)	spam
1003	RT @username: I love the south African accent is so nice	non-spam
1004	Thankyou so much you beautiful boy. You are the reason I'm still here today. Believing. I love you so much and I'll always be here for you	non-spam
1005	RT @username: Today's smartphones have roughly the same computing power as a desktop computer from 2005. http://t.co/PRklvGTBTR	non-spam

The main target of spammers is the trending topics in Twitter. They use the trending keywords or hashtags in their tweets to get the attention of the users [1]. A study shows that spam tweets contain more hashtags compared to normal tweets. Another study reveals nearly 3% of the tweets in Twitter are spam tweets [2]. A recent study shows that nearly 15% of the Twitter accounts are bots [3]. Twitter spam is capable of doing more damage to the users compared to email spam [4]. Also, click through rate of Twitter posts is 0.13% which is greater than the click-through rate 0.0003%–0.0006% of email spam [5]. Twitter also provides several ways to report the spam. Users can report the spammer to Twitter by going to spammer's profile interface and mark him as a spammer. Similarly, users can also report the individual tweets by navigating to spam tweet and mark it as a spam. Most of the existing spam detection techniques in Twitter often detect spammers and block the accounts of spammers [1], [6]–[11]. However, spammers can create new accounts and spread the spam again. So there is a need for robust spam detection techniques which detect the spam at tweet level. There is less work in the literature which focus on tweet level spam detection [12]–[14].

Manuscript received March 9, 2018; revised August 19, 2018; accepted October 21, 2018. Date of publication November 19, 2018; date of current version December 3, 2018. This work was supported by the Visvesvaraya Ph.D. Scheme of Ministry of Electronics and Information Technology, Government of India. *(Corresponding author: Sreekanth Madisetty.)*

The authors are with the Department of Computer Science and Engineering, IIT Hyderabad, Hyderabad 502285, India (e-mail: cs15resch11006@iith.ac.in; maunendra@iith.ac.in).

Digital Object Identifier 10.1109/TCSS.2018.2878852

[1] https://goo.gl/iBc9bg

Few examples of spam and non-spam tweets are shown in Table I. These examples are taken from the data sets described in Section IV. It can be observed that the first and third spam tweets are having keywords and hashtags such as follows, followers, #TeamFollowBack, and #TeamFollow. This indicates that the users are asking to retweet their tweets then they will follow back. The second spam tweet indicates that the user is promoting Android or iPad games aggressively by using the tags #android and #androidgames. Some tweets often contain irrelevant or malicious URLs in the tweet with trending keywords or hashtags. Last three tweets in Table I are examples of non-spam tweets. These tweets do not contain any spam information.

Toward the task of spam detection at tweet level, we proposed a novel method which combines five convolutional neural networks (CNNs) models and one feature-based model via a neural network. Here, neural network acts as a meta-classifier. Each CNN is trained with word embeddings of different dimensions. The feature-based model uses three types of features, namely, user-based, content-based, and n-gram features. We use an ensemble of different classifiers to produce the final output because the performance of an ensemble is often better than individual classifiers. Most of the winners of data challenge competitions often use ensemble methods to win the competitions. For example, Netflix prize competition winners [15] use an ensemble method, and the winners of KDD Cup [16] also use an ensemble method. To increase the performance of an ensemble, we wanted to have different feature sets in each classifier. The difference between our proposed method with existing methods is that we combine both handcrafted features and word embedding features to capture more information about spam and non-spam tweets. There is no such algorithm in the literature for spam detection in Twitter.

The main contributions of our work are as follows.

1) We develop deep learning and feature-based methods for the task of spam detection at tweet level.
2) We use word embedding features in deep learning methods and user-based, content-based, and n-gram features in the feature-based method.
3) We evaluated our approach on two different data sets (balanced and imbalanced).

Rest of this paper is organized as follows. Related literature for current work is presented in Section II. In Section III, details of the proposed method are described. Experimental evaluation of the method is shown in Section IV. We conclude the work by providing directions for future research in Section V.

II. Related Work

We describe the related work in three areas: spam detection in long text data (e-mail, web, reviews), detection of spammers in Twitter, and spam detection at tweet level in Twitter.

Drucker *et al.* [17] studied the use of support vector machines (SVMs) to classify the e-mail as spam or non-spam. Androutsopoulos *et al.* [18] showed that a Naive Bayesian classifier can be used to filter spam e-mails. Carreras and Marquez [19] used boosting trees for anti-spam e-mail filtering and showed that it outperforms the Naive Bayes and Decision trees. Zhou *et al.* [20] presented a three-way decision method for e-mail spam filtering. The main advantage of this method is it provides feedback to the users about emails thereby reducing misclassification.

Ntoulas *et al.* [21] described several content-based methods for web spam detection and combined these methods to produce an accurate classifier. A method is proposed to detect web spam by propagating both trust and distrust with target differentiation in [22]. The authors assigned two scores for each web page: T-Rank to compute the trustworthiness of the page and D-Rank to compute the untrustworthiness of the webpage. Both trust and distrust are propagated with target differentiation. They used T-Rank for spam demotion and D-Rank for spam detection. In this approach, they also overcome the disadvantages of both TrustRank [23] and Anti-TrustRank [24]. A method to detect web spam by propagating differential trust with community discovery is presented in [25]. The authors used a random walk-based community discovery algorithm to select suspicious communities. Most of the members in these suspicious communities are spam pages. They limit the across-community-boundary trust propagation through these suspicious communities.

There are two ways to detect spam in Twitter. One way is to detect spammers and blocking them. Another way is to detect the spam at tweet level and block those spam tweets from spreading across the Twitter network. Lee *et al.* [6] presented a method to detect spammers in Twitter. The authors have studied the behavior of content polluters for seven months and identified 36 000 candidate spammers. They also did analysis on user behavior over time. Several approaches have been proposed to detect spammers by defining user attributes and content-based attributes [1], [7]–[9]. A system is developed to identify spammers on Twitter by using graph model [26]. The authors used graph-based features, and novel content-based features for this problem. A method to discover spammers in Twitter using multiple view information is proposed in [10]. The authors proposed the method by integrating multiple view information and social regularization term. The multiple views considered are: text, URL, and hashtag features. A method to detect spam and promoting campaigns in the Twitter is proposed in [11]. The authors have proposed URL-driven estimation method to find the similarity between two accounts who post the URLs for the same purpose (promoting or spam campaign). They proposed a graph-based method to extract dense subgraphs as candidate campaigns. They also defined different features to distinguish between spam campaign and promoting the campaign.

The above methods are used to detect spammers but not spam tweets. If an algorithm detects a user account as spammer then that account can be blocked and potential spams that may come from those accounts in the future can be stopped. However, if the spammer is detected and blocked, the spammer can create a new account and spread spams from that account.

The second way to detect the spam in Twitter is at tweet level. Martinez-Romo and Araujo [27] described a method to identify malicious tweets using statistical analysis of a language in trending topics. The method is purely content-based it does not consider details of a user, e.g., connections, past activity in the Twitter space. A method to detect spam tweets by taking the cues of e-mail content-based techniques is proposed in [28]. Wang *et al.* [13] used most of the tweet-inherent features for spam detection on Twitter at tweet level. The authors have used two data sets for their study. A performance evaluation of machine learning-based methods for spam detection at tweet level is described in [14]. The authors extracted 12 lightweight features for their study. A review of Twitter spam detection methods with the comparative analysis is described in [29]. A hashtag-oriented twitter spam data set is created by [30]. The authors have collected 14 million tweets and named the data set as HSpam14. Sedhai and Sun [12] proposed a semisupervised spam detection framework. They have used four lightweight detectors to detect the spam at tweet level. Most of the above methods use content-based, user-based, sentiment, part of speech tag, and network features. We wanted to use word embedding features which act as universal features for our task. A deep learning-based method for spam detection is presented in [31]. Le and Mikolov [32] have constructed the tweet vector by combining the document vector of the tweet which is obtained by Paragraph Vector modeling and word vectors. These combined vectors act as the input features for the machine learning algorithms (random forest and neural networks).

Although there exists sufficient literature on spam detection in emails, web, and reviews, the lengths of the tweets are generally smaller and tweets are often noisy in nature. There are also several works which detect the given user is spammer or non-spammer and block him. However, spammer can create a new account and keep posting spam tweets on Twitter. So there is a need for developing systems that can detect the spam at tweet level to prevent the spam that spreads across the network. There is less focus in the literature for detecting the spam at tweet level. We focus on the problem of detecting the given tweet is a spam or not.

III. Methodology

The problem addressed in this paper can be defined as follows: given a tweet t, classify whether it is a spam or not. In this section, we first discuss our proposed CNN architecture using various word embeddings with different dimensions to detect the spam at tweet level. Next, we discuss the traditional feature-based model which uses user-based, content-based, and n-gram features for the same problem. Finally, we discuss our neural network-based ensemble architecture to detect whether the given tweet is spam tweet or non-spam tweet.

A. Convolutional Neural Networks

For better working of any neural network algorithm, there are two major decisions to take. One is feature representation, and another one is network architecture. Here, we describe about these two aspects in detail.

1) Feature Representation: The main features of tweet come from the words contained in it. Each word in the corpus acts as a feature. There are various ways to represent the features such as one-hot vector representation and dense representation. In one-hot vector feature representation, all entries of the vector are 0s except for the entries in which feature is present. The value of that entry is 1. For example, assume our vocabulary has six words: actor, actress, cricketer, politician, student, and teacher. The one-hot vector for the word "student" could be: 000010. This is a natural representation to start with, although a poor one. One major drawback with one-hot vector representation is that it is high dimensional. The dimensionality of the vector depends on the size of the vocabulary. If the vocabulary size is of $|V|$ then a window of k words correspond to an input vector of at least $|V|.k$ units (vectors are concatenated). This feature representation makes no assumption about the word similarity. All words are equally different from each other, and it is difficult to capture the semantics. For example, "apple," "mango," and "king" are equally distant in the feature space, despite "apple" should be closer to "mango" than "king" in semantic view.

In dense representation, the features (words) are represented in low dimension. The main advantage of dense vector representation is its generalization power [33]. Similar features can have similar vectors in dense representations. However, this behavior is not there in one-hot vector representation. Another advantage is its computational speed because of low dimensions. In cases where there is less number of features and no correlation between the features then one-hot vector representation can be used otherwise dense representation is better.

2) Network Architecture: Now, we describe the neural network architecture used in this paper. CNNs have shown to be useful in computer vision [34], [35]. Recently, they are applied to problems of natural language processing (NLP) domain also. Collobert *et al.* [36] proposed a neural network architecture which can be applied to many NLP tasks such as named entity recognition, parsing, part-of-speech tagging, and chunking [37] used CNN for sentence classification. Our model is inspired from [37] and is shown in Fig. 1. The layers present in our CNN architecture are: input layer, convolution layer, pooling layer, hidden layer, and an output layer. Each tweet is comprised group of words. We use dense representation of the words in our work. Dense representations of the words can be obtained in many different ways. Some of the common ways are: the word vectors which are randomly initialized, pretrained distributional word embeddings of word2vec [38], or global vector (Glove) [39], fastText embeddings [40], or dependence-based embeddings [41]. The tweet vector is formed by concatenating the individual word vectors of the tweet. If the dimension of word vector is d and the length of the tweet is l then the dimension of tweet matrix is $l \times d$. This tweet matrix is input to the first layer of CNN as shown in Fig. 1.

Let a tweet be comprised the sequence of words: $\langle \text{term}_1, \text{term}_2, \text{term}_3, \ldots, \text{term}_n \rangle$. Then, the tweet vector is represented as

$$T_v = w_1 \circ w_2 \circ w_3 \circ \ldots \circ w_n \tag{1}$$

where w_i is the word embedding vector of $term_i$, and $\circ$ is the concatenation operator. Each $w_i \in \mathbb{R}^d$ is associated with its corresponding pretrained word vector.

Next layer is the convolution layer. Activation functions like tanh, relu, and sigmoid are used to get the convolution feature

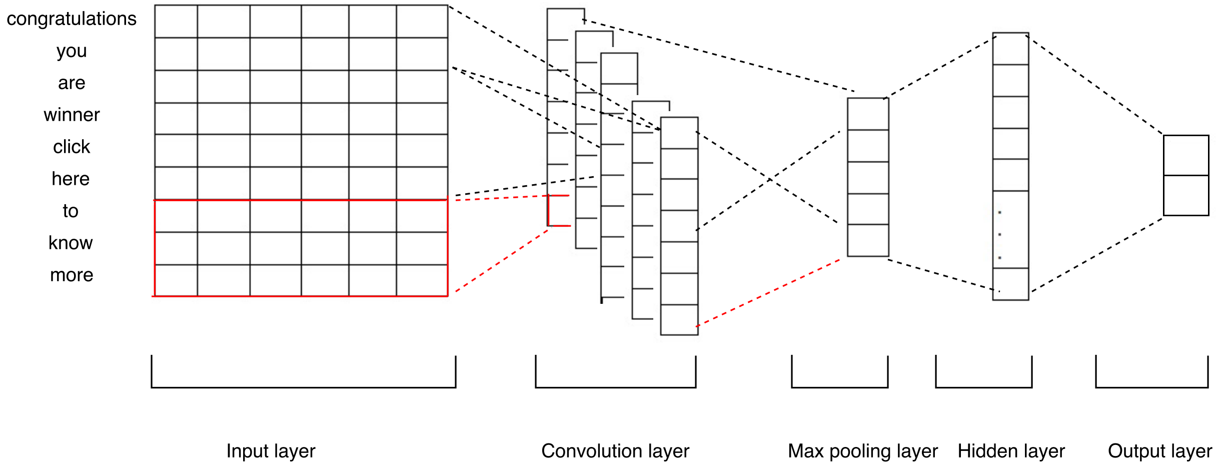


Fig. 1. CNN architecture for an example tweet.

maps. Single filter or multiple filters can be applied depending on the task. Filter length can be 1, 3, 5, and so on. If filter length is one, then the context of the words in the sentence is ignored, and target word feature map is calculated. If filter length is three, then target word feature map is calculated by considering target word, one word left and one word right to the target word. Here, context is preserved. Similarly, if filter length is five, then target word, two words left, and two words right of target word are considered. After finding convolution feature maps, most important activation should be selected. This is done by pooling layer. Normally max pooling is used in NLP tasks whereas mean pooling and min pooling are also used in computer vision. We apply max pooling on the convolution layer. Next layer is fully connected hidden dense layer. Finally, sigmoid activation function is applied to classify the given tweet. We used $l2$ regularization to avoid overfitting. The parameters used in our method are as follows: Number of filters: 250, dropout: 0.2, batch size: 32, optimizer: adam, and loss function: binary cross entropy.

Word Embeddings: We now describe the details of the word embeddings used in our work. Word embeddings are the distributional representation of words in lower dimensional space. These word embeddings can be obtained by using word2vec model [38]. There are two methods to train word embeddings in word2vec: continuous bag of words (CBOWs), and skip-gram. In CBOW, target word is predicted using the context whereas in skip-gram context is predicted using target word. CBOW is faster than skip-gram. However, skip-gram performance is better than CBOW. Mikolov *et al.* [38] have created word embeddings of Google news corpus by using word2vec model. This corpus contains 3 million words and phrases with 300 dimensions.

Glove for word representation is described in [39]. The model is created using two methods, global matrix factorization and local context window. Pretrained Twitter specific glove embeddings are available. Twitter word embeddings are created using 2 Billion tweets. It contains 27 Billion tokens, 1.2 Million vocabulary uncased and several variations of dimensions (25, 50, 100, and 200).

We have created the word embeddings of HSpam14 [30] using word2vec model with the skip-gram method and 200 dimensions. We have also used Edinburgh corpus Twitter word embeddings which are trained on 10 million tweets with 100 dimensions and 400 dimensions [42], [43].

B. Feature-Based Model

Apart from using CNNs which use word embeddings, we also use the feature-based model which uses user-based, content-based, and n-gram features. We use the following set of features to train the model.

1) User-Based Features:

1) *Verified or Not:* This feature checks whether the user profile is verified or not. This is a binary feature, returns one if it is true and returns 0 otherwise.
2) *Length of the Description:* It finds the length of the user profile description.
3) *Location Given or Not:* It checks whether location information is given by the user or not. This is also a binary feature.
4) *Followers Count:* Counts the number of followers of the user.
5) *Friends Count*: Counts the number of friends of the user. Here, friends refer to the accounts which the user is following in Twitter.
6) *Reputation Score:* This feature finds the reputation score of the user. It is calculated as follows:

$$\text{Reputation Score} = \frac{\#\text{followers}}{\#\text{followers} + \#\text{friends}}. \tag{2}$$

The reputation score is close to 1 for the users who are having more number of followers and less number of friends. For spammers, this score is lower than normal users.
7) *Status Count*: This feature finds the number of tweets posted by the user on his timeline.
8) *Registration Age of the User*: The age of the account in number of days since its creation till the most recent tweet posted by the user.
9) *Number of Lists*: It finds the number of lists that the user has subscribed to.

2) Content-Based Features:

1) *Number of Words:* This feature finds the number of words in the tweet.
2) *Length of the Tweet:* It calculates the length of the tweet.
3) *Number of Capitalized Words:* This feature finds the number of capitalization words in the tweet.
4) *Number of Exclamation Mark Symbols:* It calculates the number of exclamation mark symbols present in the tweet.
5) *Number of Question Mark Symbols:* This feature finds the number of question mark symbols present in the tweet.
6) *Number of URL Links:* It counts the number of URL links present in the tweet. Spammers often post the tweets with malicious links.
7) *Number of Hashtags:* This feature finds the number of hashtags present in the tweet.
8) *Number of Mentions:* It calculates the number of mentions in the tweet.

3) N-Gram Features:

1) *Unigrams:* Unigrams with term frequency (tf).
2) *Bigrams:* Bigrams with term frequency (tf).

Feature Statistics:

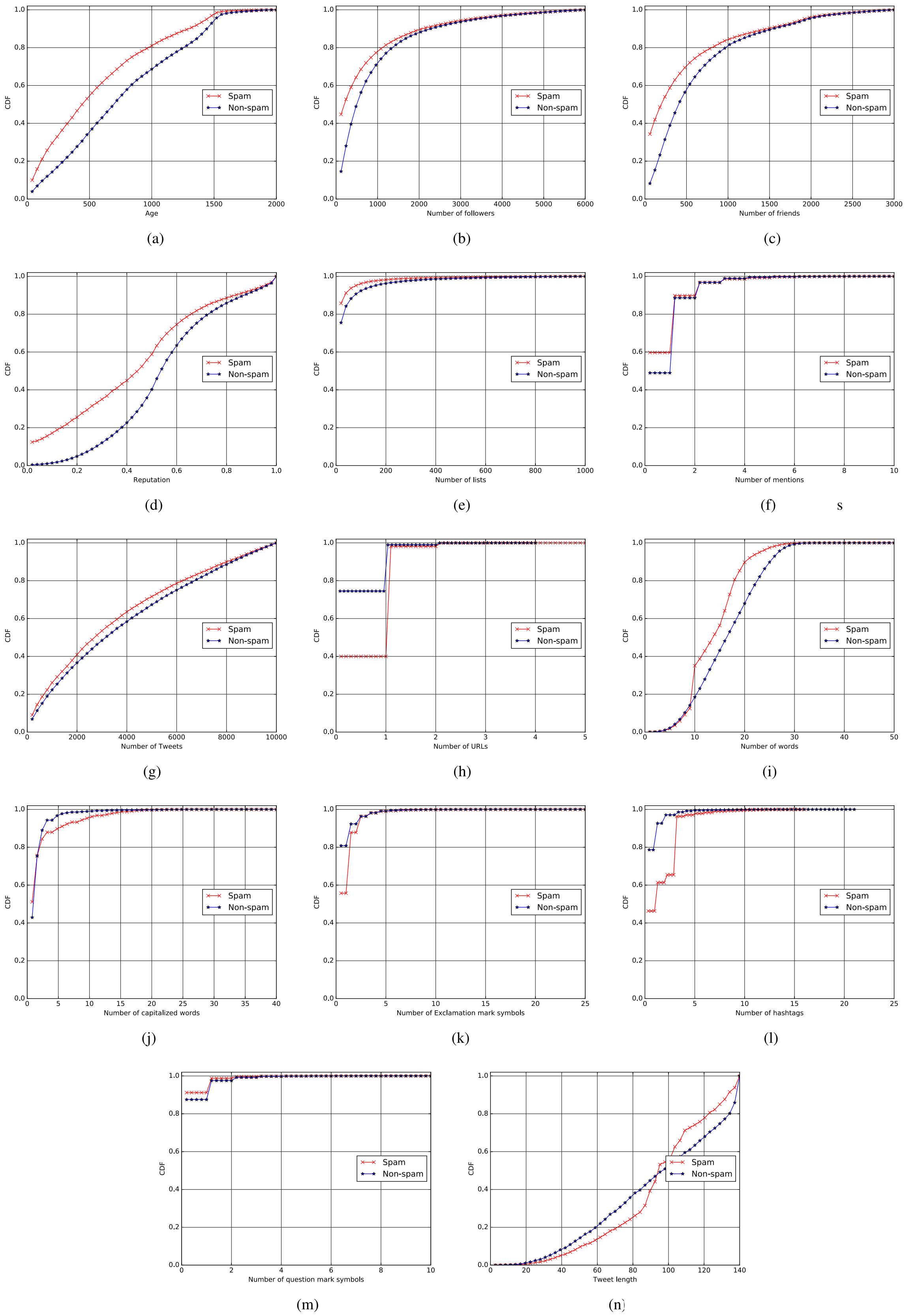
We explored the feature characteristics to differentiate the spam tweets and non-spam tweets. Fig. 2 shows the cumulative distribution functions (CDFs) of features defined in Section III-B for HSpam data set. Fig. 2(a) analyzes the age of the account in number of days for each user. Spammers are having less age compared to non-spammers. For example, 58% of the spammers have account age less than 500 days, whereas only 38% of the non-spammers have account age less than 500 days. This is due to the fact that spammers are often blocked by the spam detection algorithms, and they create new accounts again to spread the spam across the network. Number of followers feature is illustrated in Fig. 2(b). Generally, spammers have less number of followers compared to non-spammers. Fig. 2(c) analyzes the number of friends for each user. Normally, spammers follow more number of people to get followed back by them. However, this is not the case for our data set as shown in Fig. 2(c). CDF function for reputation is shown in Fig. 2(d). Usually, non-spammers have more reputation. This is due to non-spammers often have more number of followers and less number of friends. However, this is not the case for spammers. Spammers tend to subscribe more number of lists to get the attention of the people. This behavior is also not observed as shown in Fig. 2(e). From Fig. 2(e), non-spammers are subscribed to more lists. Non-spammers often have more number of mentions in their tweets compared to spammers as shown in Fig. 2(f). This is because non-spammers used to interact with people frequently whereas spammers are not. Number of tweets is illustrated in Fig. 2(g). It shows both are having similar number of tweets. Spammers often post the tweets with URLs for advertising, redirecting to external sites, and phishing. This behavior is shown in Fig. 2(h). It can be observed that spammers post more number of URLs in their tweets compared to non-spammers.

Fig. 2(i) shows the CDF function for number of words in the tweet. It shows that more number of words in the tweets are used by non-spammers. CDF function for number of capitalization words in the tweets is described in Fig. 2(j). Spammers use more number of capitalization words in the tweets. This is because they try to attract more number of people by using these types of words. Similarly, spammers use more number of exclamation mark symbols as shown in Fig. 2(k). Non-spammers use less number of hashtags compared to spammers. For example, 79% of non-spammers use less than two hashtags in their tweets, whereas only 45% of spammers use less than two hashtags. Remaining 55% of spammers uses more than two hashtags in their tweets. This is shown in Fig. 2(l). This is because spammers target trending hashtags and post the tweets with those hashtags to get the attention. Non-spammers use more number of question mark symbols as shown in Fig. 2(m). Length of the tweet is described in Fig. 2(n). Similar behavior is observed for all the features except number of question mark symbols for 1KS10KN data set as shown in Fig. 3. For number of question mark symbols feature, spammers used more number of question marks in the tweets compared to non-spammers as shown in Fig. 3(m). All other features behave similar to features of HSpam data set.

C. Ensemble

1) Methods in the Ensemble: Overview of our proposed ensemble method is shown in Fig. 4. Our ensemble method works as follows: we combine five CNN-based methods and one feature-based method in the ensemble. We also experimented with two feature-based models in the ensemble, but the results were not promising. So we use only one feature-based model in the ensemble. Each CNN is trained with word embeddings of different dimensions. The first CNN is trained with Twitter Glove word embeddings with dimensions 25, 50, 100, and 200 in static mode and with 200 dimensions in nonstatic mode. The second CNN is trained with Google news corpus word2vec embeddings with 300 dimensions in both static and nonstatic channels. The third CNN is trained with Edinburgh Twitter corpus word2vec embeddings with dimensions 100 and 400 in static mode and with 400 dimensions in nonstatic mode. The fourth CNN is trained with HSpam14 Twitter corpus with 200 dimensions in static and nonstatic modes. The final CNN is trained with random embeddings with 200 dimensions in static mode. The best method is selected from each of the above CNNs-based on F-measure and is added to the ensemble. Next, feature-based model uses user-based, content-based, and n-gram features. Random forest classifier and SVM classifier are applied by using these features. The best method is selected from these two classifiers and put it in the ensemble.

2) Meta-Classifier for Combining the Outputs: There are several ways to ensemble the classifiers: bootstrap aggregating (bagging), boosting, majority voting, weighted voting, simple averaging, and stacking. Most of the winners of different data challenge competitions use ensemble methods [15], [16]. In ensemble learning, the performance of ensemble often better than the performance of individual methods in the



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

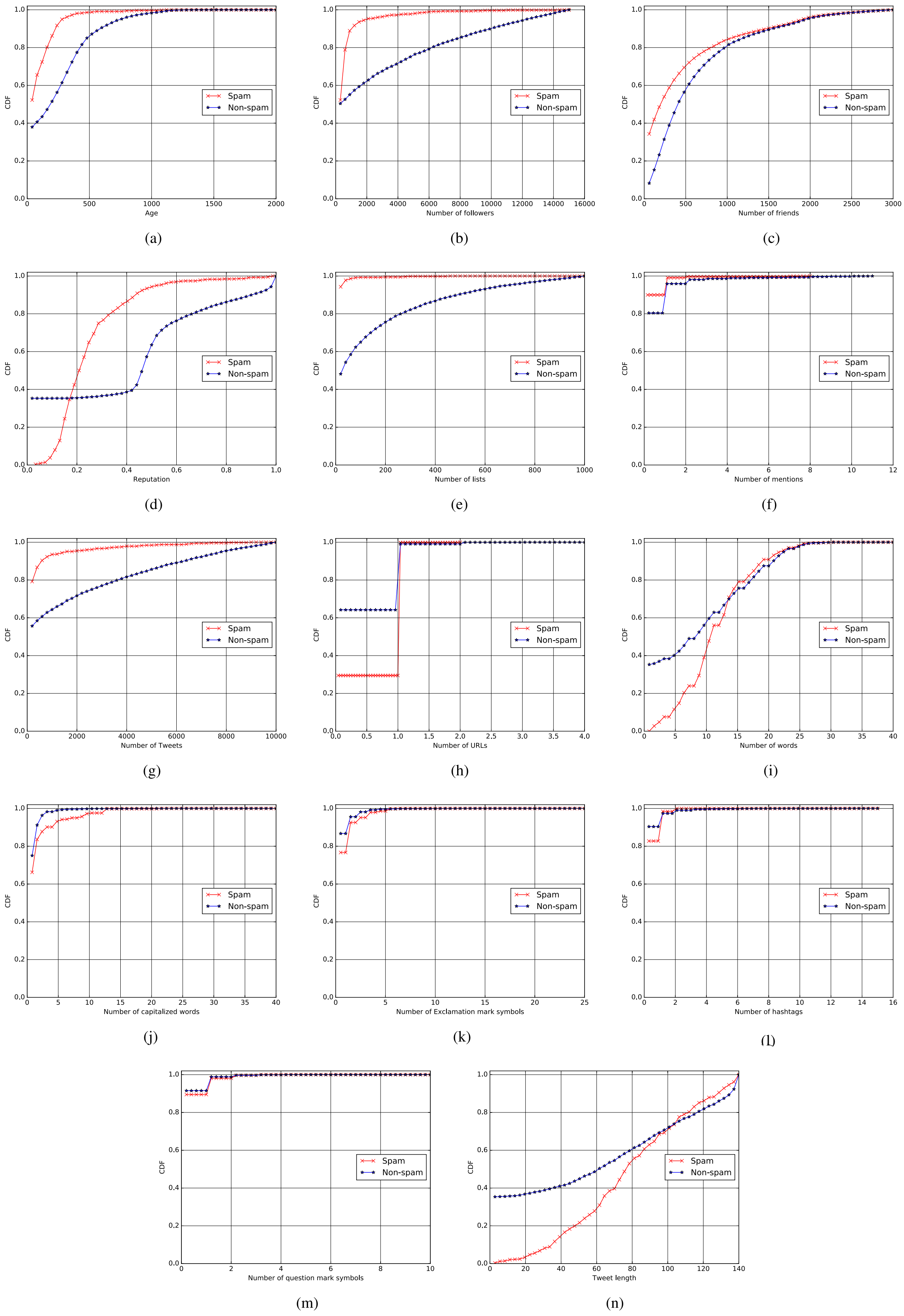
(l)

(m)

(n)

Fig. 2. CDFs of features for HSpam data set. (a) Age. (b) Number of followers. (c) Number of friends. (d) Reputation. (e) Number of lists. (f) Number of mentions. (g) Number of tweets. (h) Number of URLs. (i) Number of words. (j) Number of capitalized words. (k) Number of exclamation mark symbols. (l) Number of hashtags. (m) Number of question mark symbols. (n) Length of the tweet.

ensemble. There are different variations of CNN as described in [37]. CNN-rand uses random word embeddings for initializing the word vectors used in CNN model. CNN-static uses static pretrained word embeddings in which weights are



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n)

Fig. 3. CDFs of features for 1KS10KN data set. (a) Age. (b) Number of followers. (c) Number of friends. (d) Reputation. (e) Number of lists. (f) Number of mentions. (g) Number of tweets. (h) Number of URLs. (i) Number of words. (j) Number of capitalized words. (k) Number of exclamation mark symbols. (l) Number of hashtags. (m) Number of question mark symbols. (n) Length of the tweet.

not updated in learning, whereas weights are updated through back-propagation in CNN-nonstatic. Task-specific word embeddings are learned in CNN-nonstatic. Now, a new data set is created by the predictions of five CNNs and one

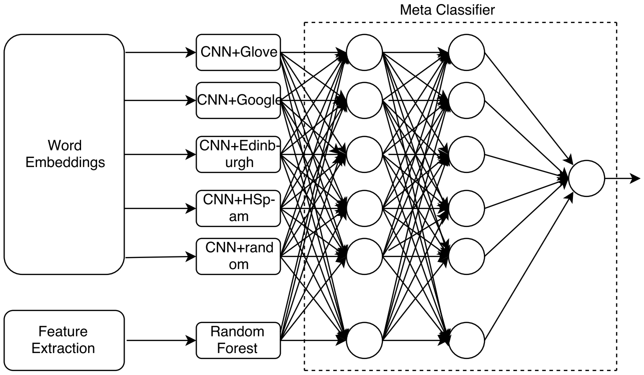


Fig. 4. Neural network-based ensemble architecture.

feature-based method. A neural network-based meta-classifier is applied to the newly created data set to classify the given tweet to spam or non-spam. This neural network has two hidden layers with six nodes each. Activation function relu is used in the hidden layers, and sigmoid is used in the output layer. We use sigmoid activation function in the output layer to make sure that the output ranges between 0 and 1.

IV. Experiments

In this section, we evaluate our proposed ensemble-based approach for detecting spam posts in social media. Before discussing the experimental results, we give a brief description of the data set and evaluation metrics used for our experiments and also the methods that we use for comparisons.

A. Data Set

We have used two benchmark data sets for our experiments in this paper. First one is the subset of HSpam14 [30] data set. We refer to it as HSpam data set. The original data set contains 14 million tweets and the data set collection process last for two months. We consider the first one million of those tweets. There were 69 650 spams in these 1 million tweets, among the non-spam tweets we randomly selected 70 000 non-spam tweets so as to have a roughly equal number of spam and non-spam tweets in our collection. This data set contains 69 650 spam tweets and 70 000 non-spam tweets. For both the data sets, we split the data into training and test sets using a 2:1 split. For HSpam14 data set, each instance was a tweet, and the associated class label for the tweet (spam or non-spam) was already available. For this data set, for each class, 67% of the data points were randomly selected and added to the training set. The remaining 33% were added to the test set. The second data set is taken from [44]. This data set contains ids of tweets posted by several users. Among the users who posted the tweets, 1000 were spammers, and remaining 10 000 were legitimate users. We use the tweet ids to retrieve the tweets. Some tweets could not be retrieved because they might be deleted from the Twitter. We ignore the users whose tweets are not present. Users were marked as spammers and non-spammers in the given data set. For each spammer, one tweet was randomly selected and marked as spam. For non-spammers, one tweet was randomly selected and was marked as non-spam. Once we get the initial data in this manner, we proceed to split that data into training and test. Here also, from the spam tweets, 67% were randomly selected and put into the training set. The remaining 33% were added to the test set. Similarly, for the non-spam tweets also, 67% were selected into training set and the remaining 33% were added into the test data set. The same approach for creating the data set for spam detection was taken in [13]. The resultant list contains 1000 spam tweets and 9835 non-spam tweets. We refer to this data set as 1KS10KN data set. So in our experiments, we have two data sets, with one being of balanced- containing a roughly equal number of spam and non-spam tweets whereas the other data set having a class imbalance with the non-spam class having a significantly larger number of examples. The data set aside for training was used for fivefold cross validation.

B. Evaluation Metrics

The metrics used for evaluating our proposed approach are: accuracy, precision, recall, F-Measure, execution time, and area under curve (AUC). We consider spam class to be positive class and non-spam class to be the negative class. True positive (tp) refers to the number of spam tweets correctly classified as spam, false negative (fn) denotes the number of spam tweets wrongly classified as non-spam, false positive (fp) refers to the number of non-spam tweets are wrongly classified as spam, and true negative (tn) denotes the number of non-spam tweets correctly classified as non-spam.

C. Methods Used for Comparison

The following are the methods used for comparison in this paper.

1) $CNN + Twitter\ Glove$: This method uses the CNN architecture described in Section III-A with Twitter Glove word embeddings.
2) $CNN + Google\ News$: CNN with Google news corpus word2vec embeddings are used in this method.
3) $CNN + Edinburgh$: This method uses CNN with Edinburgh Twitter corpus word2vec embeddings.
4) $CNN + HSpam$: CNN with HSpam14 Twitter corpus word2vec embeddings are used in this method.
5) $CNN + Random$: This method uses CNN with random embeddings.
6) *Feature-Based Model:* This method uses user-based, content-based, and n-gram features.
7) *Chen et al. [45]:* This method is taken from literature. User-based and content-based features are used in this method.
8) *Wang et al. [13]:* This method is also taken from literature. It uses user-based features, n-grams, sentiment features, and content features.
9) *Proposed Method:* This is our proposed ensemble-based method which is described in Section III.

For the first four methods ($CNN + Twitter\ Glove$, $CNN + Google\ news$, $CNN+Edinburgh$, $CNN+HSpam$), we consider different dimensions of embeddings space and also static and nonstatic versions of generating the embeddings.

D. Results and Discussion

1) Results on 1KS10KN Data Set: The results of CNN with different word embeddings and feature-based model for the

TABLE II
EVALUATION RESULTS FOR 1KS10KN DATA SET

Method	Execution Time (ms)	Precision	Recall	F-Measure
CNN + Glove25d	0.070	0.933	0.828	0.877
CNN + Glove50d	0.074	0.941	0.835	0.885
CNN + Glove100d	0.122	0.905	0.867	0.885
CNN + Glove200d	0.182	0.920	0.853	0.885
CNN + Glove200d ns	0.187	0.860	0.860	0.860
CNN + Google300d	0.269	0.838	0.856	0.847
CNN + Google300d ns	0.262	0.926	0.839	0.880
CNN + Edinburgh100d	0.116	0.910	0.814	0.859
CNN + Edinburgh400d	0.344	0.893	0.821	0.856
CNN + Edinburgh400d ns	0.347	0.851	0.825	0.838
CNN + HSpam200d	0.190	0.903	0.846	0.873
CNN + HSpam200d ns	0.198	0.899	0.842	0.870
CNN + Random	0.181	0.985	0.670	0.797
Random Forest (Feature based)	0.018	0.760	0.958	0.848
SVM (Feature based)	0.017	0.716	0.972	0.824
Chen et al. [45]	0.001	0.631	0.979	0.768
Wang et al. [13]	0.005	0.643	0.965	0.771
Proposed Method	1.147	0.922	0.867	0.893

1KS10KN data set are presented in Table II. Table II contains the results of CNN with Twitter Glove word embeddings of dimensions 25, 50, 100, and 200. The results of static and nonstatic channels of CNN are also shown. A static channel is where the word embeddings are kept static throughout the training whereas in the nonstatic channel the word embeddings are tuned during the training. We can observe that accuracy, precision, and F-Measure for word embeddings with 100 dimensions is better than all other dimensions whereas for *recall* evaluation metric word embeddings with 25 dimensions is performing better. Table II also contain the results of CNN with Google word embeddings with 300 dimensions for static and nonstatic channels. It can be observed that for recall evaluation metric static channel is doing good, whereas nonstatic channel is performing better for the remaining evaluation metrics. Similar to Google embeddings, recall metric is performing better for static channel, but nonstatic channel performance is better for all other evaluation metrics in Edinburgh Twitter word2vec embeddings. The results of CNN with HSpam14 Twitter word2vec embeddings are also shown in Table II. Similar to Google and Edinburgh embeddings, recall metric values are better for static channel, and remaining metric values are better for nonstatic channel. Although precision (predicted spam tweets are actually spam) increases, the ability of recall (actual spam tweets are not detected) goes down for nonstatic channel of each CNN.

From the results of classical features, we can observe that for accuracy, precision, and F-Measure metrics Random Forest algorithm is performing better and for recall, SVM algorithm is doing good. Training is affected by this imbalanced data set because more number of training instances are non-spam and less number of instances are spam. The optimal parameters of random forest were number_of_trees = 50, min_samples_split = 10, and those of SVM were $C = 0.8$, kernel = linear, and penalty = l2.

To create the ensemble, best performing method from each of the above word embedding type and from the feature-based model are selected along with random embedding. Outputs of these selected models are fed to neural network meta-classifier which gives the final prediction. From Table II, it can be noted that our proposed ensemble method is performing better than the existing methods in the literature. The precision is low for literature methods [13], [45] compared to deep learning methods and the proposed method. This is because the features defined in these methods are not able to properly detect the spam content. The precision of our feature-based method is also low. However, its value is better than literature methods. We defined more features in our feature-based method. The extra features which are defined in our proposed method are useful. However, these extra features also inadequate as the precision value is still inferior compared to the deep learning methods.

Twitter is a real-time system and it is extremely important to block the spam tweets in real time before they spread in the network and start causing potential damages. We have used the execution time of the detection algorithm as an evaluation metric to understand whether the algorithm can mark a tweet as spam or non-spam in real time. We performed a detailed timing analysis. Table II also reports execution times in milliseconds. Execution time is the time taken to classify one tweet as a spam or a non-spam. We observed that the proposed method, apart from having to work with larger dimensions of feature vectors, and having to work in multiple stages (the meta-classifier can be triggered only after each method in the ensemble has completed its execution), the detection can be completed in around 1–2 ms. Feature-based methods and literature methods have less execution time. This is because less number of features are used in those methods. On the other hand, deep learning methods have more execution time because of more number of dense features used in these methods. It can be observed that execution time increases with increase in dimensions of the embeddings. So less dimensional embeddings have less execution time and vice versa. Execution time for our proposed method is more compared to all other methods. It is expected because our ensemble method combines the predictions of individual methods in the ensemble. Each test tweet is given to the classifiers in the ensemble. So, total time to classify the tweet depend on the running time of individual classifier and final meta-classifier. Even with that setting time taken by the algorithm to classify the tweet is 1.14 ms which is still very good for real-time setting. For parallel execution of these classifiers and with better machine configuration we can reduce the running time even more. The experiments were run on a system with 64-GB RAM and Intel Xeon processor with clock frequency 2.6 GHz

Receiver operating characteristic (ROC) curve for 1KS10KN data set is shown in Fig. 5 with an AUC of 0.9906. ROC curve is drawn by taking false positive rate (fpr) on the x-axis and true positive rate (tpr) on the y-axis. False positive rate can be calculated as follows:

$$fpr = fp/n = fp/(fp + tn) \tag{3}$$

where n is total number of negatives. True positive rate can be calculated as follows:

$$tpr = tp/p = tp/(tp + fn) \tag{4}$$

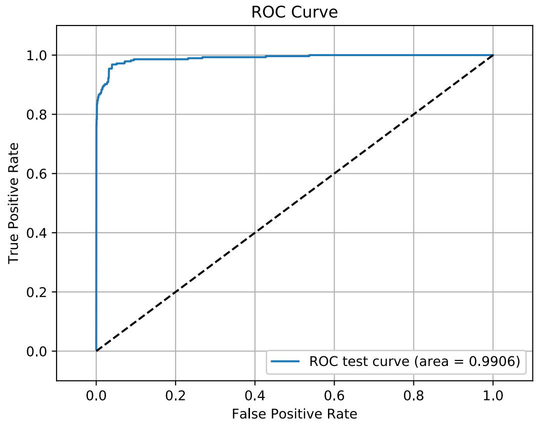


Fig. 5. ROC curve for 1KS10KN data set. For a random (spam, non-spam) pair, 99% of the times spam instance have higher value than non-spam instance.

TABLE III
EVALUATION RESULTS FOR HSPAM DATA SET

Method	Execution Time (ms)	Precision	Recall	F-Measure
CNN + Glove25d	0.069	0.947	0.908	0.927
CNN + Glove50d	0.128	0.936	0.931	0.934
CNN + Glove100d	0.209	0.937	0.937	0.937
CNN + Glove200d	0.372	0.929	0.945	0.937
CNN + Glove200d ns	0.392	0.947	0.943	0.945
CNN + Google300d	0.554	0.934	0.932	0.933
CNN + Google300d ns	0.551	0.943	0.941	0.942
CNN + Edinburgh100d	0.213	0.918	0.937	0.928
CNN + Edinburgh400d	0.707	0.936	0.933	0.935
CNN + Edinburgh400d ns	0.729	0.945	0.938	0.941
CNN + HSpam200d	0.362	0.943	0.933	0.938
CNN + HSpam200d ns	0.387	0.943	0.946	0.944
CNN + Random	0.363	0.955	0.828	0.887
Random Forest (Feature based)	0.010	0.956	0.840	0.894
SVM (Feature based)	0.011	0.934	0.902	0.918
Chen et al. [45]	0.002	0.856	0.641	0.733
Wang et al. [13]	0.027	0.947	0.797	0.865
Proposed Method	2.444	0.942	0.950	0.946

where p is total number of positives. For each given instance, our classifier outputs a score between 0 and 1. The higher the score more is the chance that the tweet is spam. We select a score threshold t $(0 < t < 1)$ above which the instance is classified as belonging to positive class (spam). If the score is less than or equal to threshold then it is assigned to negative class (non-spam). We select the threshold using the ROC curve. AUC for the ROC curve 0.9906 which also indicates that given a pair of (spam, non-spam) tweets the spam tweet gets a higher score than the non-spam tweet in 99.06% times.

2) Results on HSpam Data Set: The results of our CNN approach with Twitter Glove word embeddings, Google news word2vec embeddings, Twitter Edinburgh corpus word embeddings, and HSpam14 corpus word2vec embeddings for HSpam data set are presented in Table III. The method which is using Twitter Glove embeddings with 200 dimensions and nonstatic channel is performing better than all other Glove embeddings. Similarly, the method with 300 dimensions and nonstatic mode is doing good among Google news word2vec embeddings. In Edinburgh word embeddings, the method with 400 dimensions and nonstatic channel evaluation results are higher. In HSpam14 word embeddings, the CNN method with 200 dimensions and nonstatic mode results are better. One common observation in all the above methods for HSpam data set is that the method with the highest dimension size and nonstatic channel is performing better than other parameter combinations. However, for 1KS10KN data set, this behavior is not observed because the data set is imbalanced. It can be observed that our proposed method outperforms the other existing methods from the literature. Execution times in milliseconds are also reported in Table III. Chen *et al.* [45] took less execution time compared to all other methods. This is because it uses very less number of features (12). Our method takes more time compared to literature methods. However, the value is less (2.144 ms). ROC curve for HSpam data set is shown in Fig. 6. AUC for the ROC curve 0.9643 which also indicates that given a pair of (spam, non-spam) tweets the spam tweet gets a higher score than the non-spam tweet in 96.43% times.

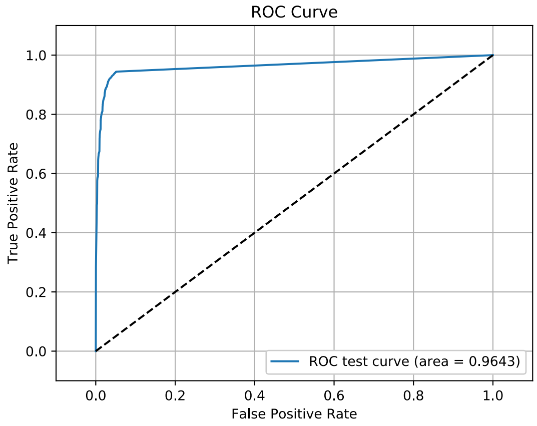


Fig. 6. ROC curve for HSpam data set. For a random (spam, non-spam) pair, 96% of the times spam tweet have higher value than non-spam tweet.

Although the proposed method gives better or comparative performance with the baselines for both the data sets, the gap in this performance difference is higher for 1KS10KN data set and lower for HSpam data set. This is because the 1KS10KN data set is smaller and unbalanced whereas HSpam data set is much larger and balanced. The performance of individual methods is much better in HSpam data set. Hence, when we apply ensemble there is no significant boost in the final performance since the individual methods are quite good. However, for the other data set (1KS10KN), the individual methods suffer from size and imbalance of the data and their performances are moderate. The meta-classifier in the ensemble is able to lift the performance by a significant margin for this data set.

3) Performance of Learned Model on Entire HSpam14 Data set: Spammers often try to fool spam detection techniques by changing the spamming strategies. Due to this, the spam detection algorithms need to be changed or at least retrained periodically. Also, as time passes, people post tweets about newer events or topics, resulting in many new words and hashtags getting added to the vocabulary. Spammers try to know the features of the detection system and they can change the type of spam tweets such that identified features will not be present in their tweets. The systems which use only feature-based methods find it difficult to identify these types of spam

TABLE IV

EVALUATION RESULTS FOR ENTIRE HSPAM14 DATA SET

Method	Accuracy	Precision	Recall	F-Measure
CNN + Glove200d ns	0.912	0.711	0.945	0.812
CNN + Google300d ns	0.952	0.869	0.895	0.882
CNN + Edinburgh400d ns	0.936	0.782	0.943	0.855
CNN + HSpam200d ns	0.939	0.796	0.938	0.861
CNN + Random	0.922	0.785	0.839	0.811
Random Forest (Feature based)	0.823	0.549	0.667	0.602
Chen et al. [45]	0.836	0.579	0.670	0.621
Wang et al. [13]	0.910	0.788	0.757	0.772
Proposed Method	0.957	0.880	0.909	0.894

tweets. However, our algorithm combines both feature-based and deep-learning based methods. It is very difficult to know the features of deep learning methods because they use word embedding features. So even though spammers try to fool the detection system, our method is robust enough to detect the spam tweets. In this experiment, we wanted to evaluate the performance of the proposed method when it is trained on a small data set and is tested on a much larger data set. If the performance is poor, then the algorithm needs to be retrained quite often. On the other hand, if the algorithm performs well, then the frequency of periodic retraining can be much lesser, which is always desirable. We perform an experiment where we apply our model trained with 0.15 million tweets from the HSpam14 data set to identify spams in remaining tweets from the same HSpam14 data set. Although the original data set is of 14 million tweet ids, all those tweets could not be downloaded (tweets might be deleted by the users or by Twitter). Only 7.2 million (roughly 50 times the size of the training set) tweets were downloaded. We designed this experiment to see whether the proposed model can extract useful signatures/signals from a small set of labeled data. It appears that the proposed method is able to do that. The ensemble method performs quite well on this large pool of unseen tweets and obtains an F-score of 0.894 outperforming all other methods considered (baselines and algorithms from the literature). The detailed comparison is provided in Table IV. Results of this experiment establish the robustness of the proposed ensemble approach.

V. CONCLUSION

Spam detection at tweet level is difficult compared to spammer detection in Twitter. In this paper, we have proposed a neural network-based ensemble technique consisting of deep learning methods and traditional feature-based methods to detect the spam at tweet level. We experimented with multiple word embeddings using CNN. We have used two data sets HSpam and 1KS10KN. HSpam data set is a balanced data set, whereas 1KS10KN is an imbalanced data set. In the 1KS10KN data set, the majority of the instances are non-spam. Machine learning algorithms are often biased toward majority class. This is why the recall of 1KS10KN data set is low whereas recall of HSpam data set is high for CNNs with nonstatic channel. Our proposed algorithm outperforms all other methods for both 1KS10KN and HSpam data sets. Even the model trained with small number of examples showed excellent performance upon being applied to large number of unseen tweets. In all the experiments, performance of the proposed approach was found to be superior than the baseline methods.

The feature-based methods perform quite poorly when compared against the deep learning methods for the HSpam14 data set. We would like to identify better feature representation for the data. For the deep learning-based methods, the inputs were only the tweets without any additional information. It might be interesting to see whether the performances of the deep learning methods can be further improved by considering additional information about the tweets or their authors.

ACKNOWLEDGMENT

This paper is an outcome of the research and development work done under the Visvesvaraya PhD Scheme of Ministry of Electronics and Information Technology, Government of India.

REFERENCES

[1] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, “Detecting spammers on Twitter,” in *Proc. Collaboration, Electron. Messaging, Anti-Abuse Spam Conf. (CEAS)*, vol. 6, 2010, p. 12.

[2] Pear Analytics, San Antonio, TX, USA. (2009). *Twitter Study–August 2009*. [Online]. Available: http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf

[3] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini. (2017). “Online human-bot interactions: Detection, estimation, and characterization.” [Online]. Available: https://arxiv.org/abs/1703.03107

[4] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, “Design and evaluation of a real-time URL spam filtering service,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2011, pp. 447–462.

[5] C. Grier, K. Thomas, V. Paxson, and M. Zhang, “@spam: The underground on 140 characters or less,” in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 27–37.

[6] K. Lee, B. D. Eoff, and J. Caverlee, “Seven months with the devils: A long-term study of content polluters on Twitter,” in *Proc. ICWSM*, 2011, pp. 185–192.

[7] M. McCord and M. Chuah, “Spam detection on Twitter using traditional classifiers,” in *Proc. Int. Conf. Autonomic Trusted Comput.* Berlin, Germany: Springer, 2011, pp. 175–186.

[8] Z. Chu, I. Widjaja, and H. Wang, “Detecting social spam campaigns on Twitter,” in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2012, pp. 455–472.

[9] X. Zheng, Z. Zeng, Z. Chen, Y. Yu, and C. Rong, “Detecting spammers on social networks,” *Neurocomputing*, vol. 159, pp. 27–34, Jul. 2015.

[10] H. Shen, F. Ma, X. Zhang, L. Zong, X. Liu, and W. Liang, “Discovering social spammers from multiple views,” *Neurocomputing*, vol. 225, pp. 49–57, Feb. 2017.

[11] X. Zhang, Z. Li, S. Zhu, and W. Liang, “Detecting spam and promoting campaigns in Twitter,” *ACM Trans. Web*, vol. 10, no. 1, 2016, Art. no. 4.

[12] S. Sedhai and A. Sun, “Semi-supervised spam detection in Twitter stream,” *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 1, pp. 169–175, Mar. 2017.

[13] B. Wang, A. Zubiaga, M. Liakata, and R. Procter, “Making the most of tweet-inherent features for social spam detection on Twitter,” in *Proc. Making Sense Microposts*, 2015, pp. 10–16.

[14] C. Chen *et al.*, “A performance evaluation of machine learning-based streaming spam tweets detection,” *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 3, pp. 65–76, Sep. 2015.

[15] A. Töscher, M. Jahrer, and R. M. Bell, “The bigchaos solution to the netflix grand prize,” *Netflix Prize Document*, pp. 1–52, Sep. 2009

[16] A. Niculescu-Mizil *et al.*, “Winning the KDD cup orange challenge with ensemble selection,” in *Proc. Workshop Conf. (JMLR)*, vol. 7, 2009, pp. 23–34.

[17] H. Drucker, D. Wu, and V. N. Vapnik, “Support vector machines for spam categorization,” *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1048–1054, Sep. 1999.

[18] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, “An evaluation of Naïve Bayesian anti-spam filtering techniques,” in *Proc. 11th Eur. Conf. Mach. Learn., Workshop Mach. Learn. New Inf. Age*, May 2000, pp. 9–17.

[19] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," in *Proc. 4th Int. Conf. Recent Adv. Natural Lang. Process.*, Sep. 2001, pp. 58–64.

[20] B. Zhou, Y. Yao, and J. Luo, "A three-way decision approach to email spam filtering," in *Proc. Can. Conf. Artif. Intell.* Berlin, Germany: Springer, May 2010, pp. 28–39.

[21] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam Web pages through content analysis," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 83–92.

[22] X. Zhang, Y. Wang, N. Mou, and W. Liang, "Propagating both trust and distrust with target differentiation for combating link-based Web spam," *ACM Trans. Web*, vol. 8, no. 3, 2014, Art. no. 15.

[23] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating Web spam with trustrank," in *Proc. 13th Int. Conf. Very Large Data Bases (VLDB)*, vol. 30, 2004, pp. 576–587.

[24] V. Krishnan and R. Raj, "Web spam detection with anti-trust rank," in *Proc. AIRWeb*, vol. 6, 2006, pp. 37–40.

[25] X. Zhang, Y. Feng, H. Shen, and W. Liang, "Differential trust propagation with community discovery for link-based Web spam demotion," in *Proc. Int. Conf. Web-Age Inf. Manage.* Cham, Switzerland: Springer, 2015, pp. 452–456.

[26] A. H. Wang, "Don't follow me: Spam detection in Twitter," in *Proc. Int. Conf. Secur. Cryptogr. (SECRYPT)*, Jul. 2010, pp. 1–10.

[27] J. Martinez-Romo and L. Araujo, "Detecting malicious tweets in trending topics using a statistical analysis of language," *Expert Syst. Appl.*, vol. 40, no. 8, pp. 2992–3000, 2013.

[28] I. Santos, I. Miñambres-Marcos, C. Laorden, P. Galán-García, A. Santamaría-Ibirika, and P. G. Bringas, "Twitter content-based spam filtering," in *Proc. Int. Joint Conf. SOCO'13-CISIS'13-ICEUTE'13*. Cham, Switzerland: Springer, 2014, pp. 449–458.

[29] T. Wu, S. Wen, Y. Xiang, and W. Zhou, "Twitter spam detection: Survey of new approaches and comparative study," *Comput. Secur.*, vol. 76, pp. 265–284, Jul. 2017.

[30] S. Sedhai and A. Sun, "HSpam14: A collection of 14 million tweets for hashtag-oriented spam research," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 223–232.

[31] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in *Proc. Australas. Comput. Sci. Week Multiconf.*, 2017, Art. no. 3.

[32] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. II-1188–II-1196.

[33] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Intell. Res.*, vol. 57, pp. 345–420, Nov. 2016.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[36] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Aug. 2011.

[37] Y. Kim. (2014). "Convolutional neural networks for sentence classification." [Online]. Available: https://arxiv.org/abs/1408.5882

[38] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[39] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP*, vol. 14, 2014, pp. 1532–1543.

[40] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. (2016). "Enriching word vectors with subword information." [Online]. Available: https://arxiv.org/abs/1607.04606

[41] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proc. ACL*, vol. 2, 2014, pp. 302–308.

[42] F. Bravo-Marquez, E. Frank, S. M. Mohammad, and B. Pfahringer, "Determining word-emotion associations from tweets by multi-label classification," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Oct. 2016, pp. 536–539.

[43] S. Petrović, M. Osborne, and V. Lavrenko, "The Edinburgh Twitter corpus," in *Proc. NAACL HLT Workshop Comput. Linguistics World Social Media*, 2010, pp. 25–26.

[44] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on Twitter," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 71–80.

[45] 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 *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 7065–7070.

Sreekanth Madisetty received the B.Tech. degree in computer science and engineering from the Jawaharlal Nehru Technological University Ananthapur, Ananthapur, India, in 2012, and the M.Tech. degree in computer science and technology from Andhra University, Visakhapatnam, India, in 2014. He is currently pursuing the Ph.D. degree in computer science and engineering with IIT Hyderabad, Hyderabad, India.

From 2014 to 2015, he was a Lecturer with the Computer Science and Engineering Department, IIIT Basar, Basar, India and IIIT RK Valley, Vempalli, India. His current research interests include information retrieval, social media analysis, natural language processing, and data mining.

Mr. Madisetty received Gold Medal for his academic excellence in M.Tech. from Andhra University.

Maunendra Sankar Desarkar received the B.E. degree from The University of Burdwan, Bardhaman, India, the M.Tech. degree from IIT Kanpur, Kanpur, India, and the Ph.D. degree from IIT Kharagpur, Kharagpur, India, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, IIT Hyderabad, Hyderabad, Inida. His current research interests include information retrieval, social media analytics, natural language processing, and machine learning.

Dr. Desarkar was a recipient of the Microsoft India Ph.D. Fellowship Award and the Honorable Mention Award in Yahoo Key Scientific Challenges Program in 2012.