Twitter Bot Detection Using Bidirectional Long Short-term Memory Neural Networks and Word Embeddings

Feng Wei and Uyen Trang Nguyen
Department of Electrical Engineering and Computer Science
York University
4700 Keele Street, Toronto, Canada M3J 1P3
{fwei, utn}@eecs.yorku.ca

***Abstract*—Twitter is a web application playing dual roles of online social networking and micro-blogging. The popularity and open structure of Twitter have attracted a large number of automated programs, known as bots. Legitimate bots generate a large amount of benign contextual content, i.e., tweets delivering news and updating feeds, while malicious bots spread spam or malicious contents. To assist human users in identifying who they are interacting with, this paper focuses on the classification of human and spambot accounts on Twitter, by employing recurrent neural networks, specifically bidirectional Long Short-term Memory (BiLSTM), to efficiently capture features across tweets. To the best of our knowledge, our work is the first that develops a recurrent neural model with word embeddings to distinguish Twitter bots from human accounts, that requires no prior knowledge or assumption about users' profiles, friendship networks, or historical behavior on the target account. Moreover, our model does not require any handcrafted features. The preliminary simulation results are very encouraging. Experiments on the cresci-2017 dataset show that our approach can achieve competitive performance compared with existing state-of-the-art bot detection systems.**

***Index Terms*—Online Social Networks, Twitter Bots, Bots Detection, Machine Learning, Neural Networks, Word Embeddings.**

I. INTRODUCTION

Twitter is a popular online social networking and micro-blogging tool. Remarkable simplicity is its distinctive feature: its community interacts via publishing text-based posts, known as tweets. Twitter has its own special memes, i.e., hashtag (#), mention (@), shortened URL (http://t.co) and retweet (RT). Hashtags, namely words or phrases prefixed with a # symbol, can group tweets by topics. For example, *#usopen2019* and *#SheTheNorth* are two trending hashtags on Twitter in September 2019. The symbol @ followed by a user name in a tweet enables the direct delivery of the tweet to that user. Links shared on Twitter, including links shared in private direct messages, will automatically be processed and shortened to an http://t.co link. A retweet is a re-posting of a tweet. Sometimes people type "RT" at the beginning of a Tweet to indicate that they are re-posting someone else's content.

The growing user population and open nature of Twitter have made itself an ideal target of exploitation from automated programs, known as bots. Automation is a double-edged sword to Twitter. On the one hand, legitimate bots generate a large volume of benign tweets, like news and blog updates. On the other hand, malicious bots have been widely exploited to spread spam or malicious contents. The definition of spam in this paper is to spread malicious, phishing, or unsolicited content in tweets. These bots randomly follow human users, expecting many users to follow them back.

The spambot problem in social networks has already received attention from researchers. As an example for spam detection, a branch of research mined the textual content of tweets [1]; others studied the redirection of embedded URLs in tweets [2], or classified the URLs landing pages [3]. Gao et al. [4] move beyond the difficulty of labeling those tweets without URLs as spam tweets, by proposing a composite tool able to match incoming tweets with underlying templates commonly used. Cresci et al. [5] introduced a bio-inspired technique to model online user behaviors.

Most existing work identify spambots through a multi-feature approach, including features on the profile, user behavior, friendship networks, and the timeline of an account. Examples of such analyses include [6]–[12]. In addition, Yang et al. [6] designed a series of new criteria, and demonstrated their effectiveness in detecting spambots that evade previous detection techniques.

In this paper, we propose a recurrent neural network (RNN) model, specifically BiLSTM, with word embeddings to distinguish Twitter bots from human accounts. To the best of our knowledge, our work is the first that develops a RNN model with word embeddings to detect Twitter bots that requires no prior knowledge or assumption about users' profiles, friendship networks, or historical behavior on the target account.

We summarize our contributions as follows.

- We propose a RNN model to distinguish Twitter bots from human accounts, instead of using traditional methods (e.g., Random Forest, Bayes Net or Support Vector Machine). BiLSTM connects two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backwards) and future (forward)

978-1-7281-6741-1/19/$31.00 ©2019 IEEE
DOI 10.1109/TPS-ISA48467.2019.00021

states simultaneously. This method can efficiently capture contextual features across tweets and achieve competitive classification accuracy compared to existing methods.
- We use word embeddings to encode tweets, instead of using traditional feature engineering or natural language processing (NLP) tools that are much more complex. This advantage allows for faster and easier implementation and deployment of the bot detection scheme.
- We conducted experiments on real-world data sets. Experiments on the cresci-2017 dataset show that our approach can achieve competitive performance compared with existing work [5], [6], [8], [11], [12], although our RNN model uses only the contextual content of tweets as the input to the model.

The remainder of the paper is organized as follows. We discuss related work in Section II. The proposed approach is described in Section III. In Section IV, we present experimental results, comparing the performance of our proposed model with that of existing state-of-the-art systems. Section V concludes the paper and outlines our future work.

II. RELATED WORK

We discuss related work on existing techniques of Twitter bot detection and recurrent neural networks.

A. Twitter Bot Detection

Traditional bot detection systems typically rely on the application of well-known machine learning algorithms on the accounts under investigation, such as [1]–[3], [6], [13]–[27]. However, since 2013, a number of research teams independently started to formalize new approaches for detecting the coordinated and synchronized behavior that characterizes groups of automated malicious accounts [5], [28]–[32]. Despite being based on different key concepts, these studies investigate groups of accounts as a whole, marking a difference with the previous literature. However, our approaches for the spam problem focus on the detection of tweets containing spam instead of detecting spam accounts. The detection of spam tweets itself can be useful for filtering spam on real time search [14], whereas the detection of spammers is related with the detection of existent spam accounts. In fact, a way to detect spammers would be filtering users who have written many spam tweets. In addition, when a spam account is detected, Twitter suspends it or even blocks his IP address temporally, so spammers only need to create a different account to continue sending spam messages or wait a while for his IP address is unlocked.

Existing methods of Twitter bot detection can be divided into two main approaches: supervised machine learning and unsupervised clustering. Both of them require complex handcrafted features.

Supervised machine learning strategy. Lee et al. [7] adopt 30 classification algorithms and tested their performance. Tree-based supervised classifiers showed the highest accuracy results. In particular, Random Forest produced the highest accuracy. In order to improve the Random Forest classifier, standard boosting and bagging techniques have been applied additionally. The authors trained the content polluter classifier based on different feature group combinations. The system in [6] provides a supervised machine learning classifier that infers whether a Twitter account is a human account or a spambot by relying on relationships among accounts, tweeting timing and level of automation. In addition, they design 10 new behavior detection features. According to their evaluation, the detection rate using their new feature set is significantly higher than that of existing work. Alsaleh et al. [9] presented a system that utilizes supervised machine learning techniques to dynamically detect Twitter bot accounts. The classification results show satisfying detection rate for this particular application. Although they adopt multilayer neural network, which is a simple feedforward neural network (FFNN), it still require complicated handcrafted features. Davis et al. [8] group features into six main classes: network, user, friend, temporal, content and sentiment, and employ Random Forest, an ensemble supervised learning method to achieve a high accuracy score. Varol et al. [10] proposed a supervised machine learning system that extracts more than a thousand features in six different classes: users, friends meta-data, tweet content, sentiment, network patterns and activity time series.

Unsupervised clustering approach. The approach in [12] considers vectors made of 126 features, extracted from both accounts and tweets, as input to modified versions of the DenStream [33] and StreamKM++ [34] clustering algorithms, to cluster feature vectors of a set of unlabeled accounts. The methodology in [11] exploits a set of 14 generic statistical behavior features related to URLs, hashtags, mentions and retweets. Feature vectors generated in this way are then compared with one another via an Euclidean distance measure. Chavoshi et al. [35] developed an unsupervised method, named DeBot, which calculates cross-user activity correlations to detect bot accounts in Twitter. Debot detects thousands of bots per day with a 94% precision and generates reports online everyday. Cresci et al. [5] proposed an unsupervised method to detect spambots, by comparing their behavior with the aim of finding similarities between automated accounts. They introduced a bio-inspired technique to model online user behaviors by so-called "digital DNA" sequences. Extracting digital DNA for an account means associating that account to a string that encodes its behavioral information. Although it achieves good detection performances, numerous handcrafted behavioral features are still required.

A recent research direction is to test the limits of current bot detection frameworks in an adversarial setting. The idea is to propose methodologies to engineer systems that can go undetected. Cresci et al. [36] proposed the use of evolutionary algorithms to improve social bot skills. Grimme et al. [37] employed a hybrid approach involving automatic and manual actions to create bots that would be classified as human by a supervised bot detection system. Despite the good intention of pointing to weaknesses in existing systems, this research might also inspire bot creators and give them a competitive advantage.

B. Recurrent Neural Networks

In the past few years, deep neural networks have achieved huge successes in many data modelling and prediction tasks, ranging from speech recognition, computer vision to natural language processing (NLP). In this paper, we apply powerful deep learning methods to social network data modelling to distinguish Twitter bots from human accounts.

Deep learning approaches are able to automatically capture, to some extent, the syntactic and semantic features from contextual content without handcrafted feature engineering, which is labor intensive and time consuming. They attract much research interest in recent years, and achieve state-of-the-art performances in many fields of NLP.

Socher et al. [38] first propose a family of recurrent neural networks (RNN) to learn the compositional semantic of variable-length phrases and sentences. Irsoy et al. [39] present a deep RNN constructed by stacking multiple recurrent layers for compositionality in language. Long short-term memory (LSTM) [40], which is a kind of RNN architecture, is explicitly designed to solve the long-term dependency problem through purpose-built memory cells. BiLSTM [41] incorporates a forward LSTM layer and a backward LSTM layer, in order to learn information from preceding as well as following tokens.

In this paper, we define the problem of Twitter bot detection as a text classification problem: we use only the contextual content of tweets as the input to our RNN model.

III. OUR PROPOSED APPROACH

In this section, we discuss our proposed method of bidirectional LSTM (BiLSTM) with word embeddings.

A. Word Embeddings

Human vocabulary comes in free text. In order to make a machine learning model understand and process the natural language, we need to transform the free-text words into numeric values. One of the simplest transformation approaches is to do a one-hot encoding, in which each distinct word stands for one dimension of the resulting vector and a binary value indicates whether the word presents (one) or not (zero).

Word embedding is a dense representation of words in the form of numeric vectors. It can be learned using a variety of language models. The most exciting point from word embedding is that, similar words are located together in the vector space, and arithmetic operations on word vectors can pose semantic or syntactic relationships. For example, vector "cat" - vector "kitten" is similar to vector "dog" - vector "puppy". However, traditional machine learning approach (e.g., Latent Dirichlet Allocation) cannot maintain such a linear relationship in the vector space.

The Global Vector (GloVe) model, proposed by Pennington et al. [42] aims to combine the count-based matrix factorization and the context-based skip-gram model together. In other words, the motivation of GloVe is to force the model to learn such linear relationship based on the co-occurreence matrix explicitly. Essentially, GloVe is a log-bilinear model with a weighted least-squares objective. Obviously, it is a hybrid method that uses machine learning based on the statistic matrix.

Word embeddings, also known as distributed word representation, is an important research topic in NLP. In recent years, it has been widely used in various NLP task, including information retrievals [43]–[45], text classification [46], machine translation [47] and machine comprehension [48]. The success of word embedding [42], [49] encourages researchers to focus on machine-learned representation instead of heavy feature engineering in NLP. By using word embeddings as the typical feature representation for words, neural networks become more competitive compared to traditional approaches in NLP.

An important advantage of word embeddings compared to conventional NLP techniques of representation (e.g., bag-of-words [50], part-of-speech tagging [51]) is that it achieves a significant dimensional reduction of the feature set needed to represent tweets, resulting in a reduction in training and inference time of the algorithms.

In this work, we adopt pre-trained GloVe word vectors on Twitter. It is based on 2 billion tweets, including 27 billion tokens, with the vocabulary size of 1.2 million. We define our vocabulary as the intersection between the words in all training samples and those in the pre-trained 200-dimensional GloVe. Given a word w, if it is in the vocabulary, we set its word-level embedding α_w to its GloVe word vector, which is fixed during the training; otherwise we have $\alpha_w = \alpha_o \in \mathbb{R}^{200}$, where α_o is a trainable parameter serving as the shared word vector of all out-of-vocabulary (OOV) words. Each tweet is sent to the Stanford CoreNLP [52] toolkit for sentence splitting and tokenization. All words containing Twitter special memes, i.e., hashtag (#), mention (@) and shortened URL (http://t.co), are mapped to several pre-defined tokens individually, i.e., $\langle HASHTAG \rangle$, $\langle USER \rangle$ and $\langle URL \rangle$, using regular expression matches.

B. BiLSTM Neural Networks

First, we briefly describe RNN, LSTM and BiLSTM individually. Then, our proposed model is described.

RNN [53] is a class of artificial neural sequence model, as shown in Fig. 1, where connections between units form a directed cycle. It takes arbitrary embedding sequences $x = (x_1, ..., x_T)$ as input, uses its internal memory network to exhibit dynamic temporal behavior. It consists of a hidden unit h and an optional output y. T is the last time step and is also the length of input sentence in this text sequence learning task. At each time step t, the hidden state h_t of the RNN is computed based on the previous hidden state h_{t-1} and the input at the current step x_t:

$$h_t = g(Ux_t + Wh_{t-1}) \quad (1)$$

where U and W are weight matrices of the network; $g(\cdot)$ is a non-linear activation function, such as an element-wise logistic sigmoid function. The output at time step t is computed as $y_t = softmax(Vh_t)$, where V is another weight parameter

of the network; $softmax$ is an activation function often implemented at the final layer of a network.

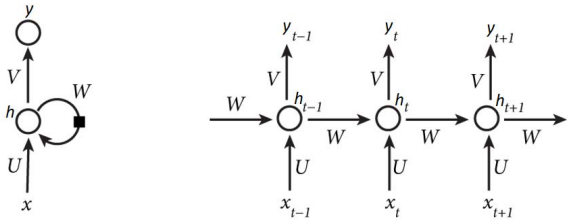

Fig. 1: **Left**: Recurrent neural network; **Right**: Recurrent neural network unfold.

LSTM [40] is a variant of RNN designed to deal with vanishing gradients problem. However, Hochreiter and Schmidhuber [40] found that the proposed architecture, which uses purpose-built memory cells to store information, is better at finding and exploiting long range context. Fig. 2 illustrates a single LSTM memory cell. For the version of LSTM used in [54], $g(\cdot)$ is implemented by the following composite function:

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + V_i c_{t-1}) \quad (2)$$

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + V_f c_{t-1}) \quad (3)$$

$$c_t = f_t c_{t-1} + i_t \; tanh(U_c x_t + W_c h_{t-1}) \quad (4)$$

$$o_t = \sigma(U_o x_t + W_o h_{t-1} + V_o c_t) \quad (5)$$

$$h_t = o_t \; tanh(c_t) \quad (6)$$

where σ is the logistic sigmoid function, and i, f, o and c are the $input\ gate$, $forget\ gate$, $output\ gate$ and $cell$ activation vectors, respectively, all of which are of the same size as the hidden vector h; U, W and V are weight matrices of the network. The weight matrices from the cell to gate vectors (e.g., W_i) are diagonal, so element m in each gate vector only receives input from element m of the cell vector.

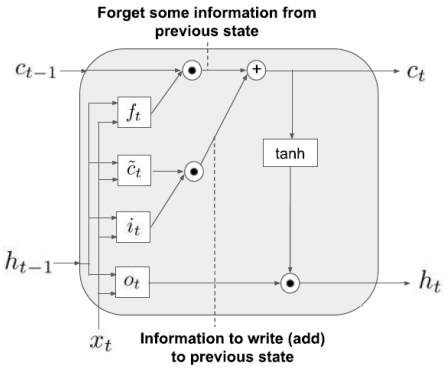

Fig. 2: Long Short-term Memory Cell.

BiLSTM uses two LSTMs to learn each token of the sequence based on both the past and the future context of the token. As shown in Fig. 3, one LSTM processes the sequence from left to right; the other one from right to left. At each time step t, a hidden forward layer with hidden unit function $\overrightarrow{h}$ is computed based on the previous hidden state $\overrightarrow{h}_{t-1}$ and the input at the current step x_t. Additionally, a hidden backward layer with hidden unit function $\overleftarrow{h}$ is computed based on the future hidden state $\overrightarrow{h}_{t+1}$ and the input at the current step x_t. The forward and backward context representations, generated by $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ respectively, are concatenated into a long vector. The combined outputs are the predictions of target sequences.

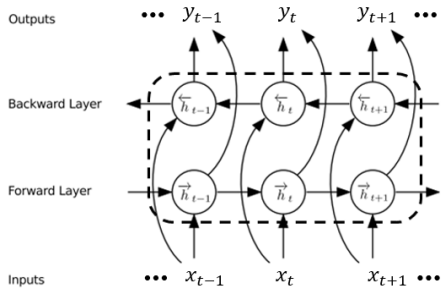

Fig. 3: Bidirectional LSTM.

Our Proposed Model. As shown in Fig. 4, we make use of a fully connected softmax layer to output posterior probabilities over labels from two classes, standing for Twitter bots or human. The input is a sequence of n tokens, $(x_1, ..., x_n)$. The predictions in both directions are modeled by three-layer BiLSTMs with hidden states $\overrightarrow{h}_{i,\ell}$ and $\overleftarrow{h}_{i,\ell}$ for input token x_i at the layer level $\ell = 1, ..., L$. The final layer's hidden state $h_{i,L} = [\overrightarrow{h}_{i,L}; \overleftarrow{h}_{i,L}]$ is used to output the probabilities over binary labels after softmax normalization. They share the word embedding layer and the softmax layer, parameterized by Θ_e and Θ_s respectively. The model is trained to minimize the negative log likelihood in both directions:

$$\mathcal{L} = -\sum_{i=1}^{n}(logp(y|x_1, ..., x_n; \Theta_e, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + logp(y|x_1, ..., x_n; \Theta_e, \overleftarrow{\Theta}_{LSTM}, \Theta_s)) \quad (7)$$

IV. EXPERIMENTAL RESULTS

In this section, we present our experimental setup and results, comparing the performance of our neural model with that of existing work [5], [6], [8], [11], [12].

A. Existing Systems for Comparison

Davis et al. [8] generate more than 1,000 features and group them into six main classes: network, user, friend, temporal, content and sentiment. Yang et al. [6] use 25 features and group them into six categories: profile-based features, content-based features, graph-based features, neighbor-based features, timing-based features and automation-based features. Miller et al. [12] consider vectors made of 126 features, extracted from both accounts and tweets. Ahmed et al. [11] exploit a set of 14 generic statistical behavior features related to URLs, hashtags, mentions and retweets. Cresci et al. [5] design two groups of user behaviour features: tweet type DNA and tweet content DNA. It is worth noting that most state-of-the-art algorithms/systems for spambot detection require a large number of data-demanding features. As shown in

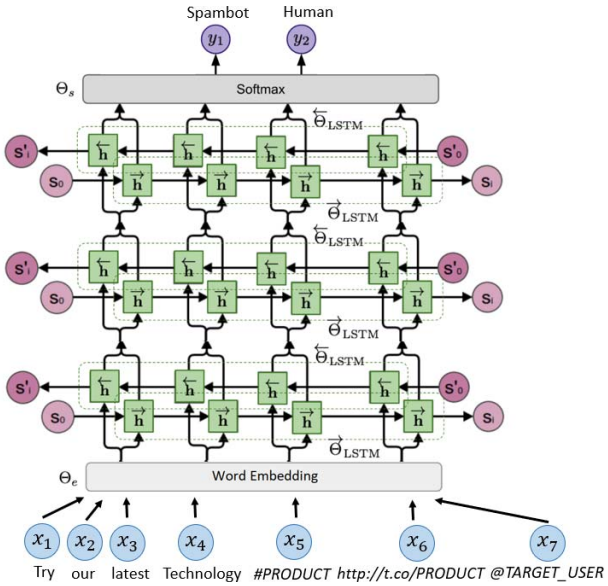

Fig. 4: An illustration of our BiLSTM neural model with word embeddings for distinguishing Twitter bots from human accounts.

Fig. 5, the existing work requires feature engineering based on six [5] to more than 1000 features [8]. Feature engineering is very expensive in terms of data collection, pre-processing and computation power. Our proposed RNN model does not rely on any feature engineering and uses only the contextual content of the tweets.

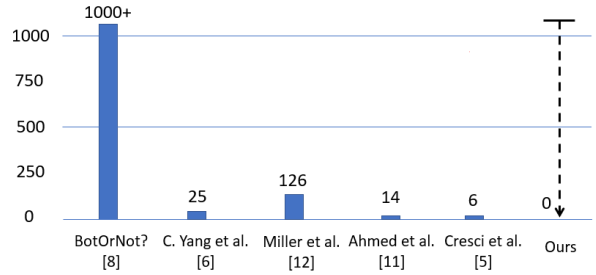

Fig. 5: The number of handcrafted feature comparison among various spambot detection techniques and algorithms reported on the cresci-2017 dataset.

B. Dataset

We evaluate our proposed model using the public annotated dataset cresci-2017 [55], consisting of 3,474 human accounts along with 8.4 million tweets and 1,455 bots along with 3 million tweets. We prepared two test sets following [56]. Test set #1 and test set #2 refer to groups where human accounts are mixed with accounts from dataset social-bot-1 and dataset social-bot-3, respectively. Social-bot-1 is about retweeters of an Italian political candidate, while social-bot-3 is about spammers of products on sale at Amazon.com. Test set #1 is composed of 1,982 accounts and 4,061,598 tweets, while test set #2 is composed of 928 accounts and 2,628,181 tweets. The statistics of datasets are shown in Table I in detail.

C. Neural Network Model Setup

Based on the design of the experiments, we tested several sets of parameters to select one that gives the experiments the best performance. These parameters are as follows:

- *Learning rate*: the model is trained using the Stochastic Gradient Descent algorithm, while the learning rate is set to 0.01.
- *Network structure*: three stacked BiLSTM layers with 200 recurrent units and one fully connected softmax layer.
- *Dropout* [57] is adopted during training, initially set to 0.5, slowly decreasing during training until it reaches 0.1 at the end.
- *Number of epochs*: 30.
- *Momentum*: 0.9.
- *Mini-batch*: 64.

D. Evaluation Metrics

To evaluate the effectiveness of our proposed RNN approach, we use four standard indicators:

- *True Positives (TP)*: the number of spambots correctly recognized;
- *True Negatives (TN)*: the number of human accounts correctly recognized;
- *False Positives (FP)*: the number of human accounts erroneously recognized as spambots;
- *False Negatives (FN)*: the number of spambots erroneously recognized as human accounts.

For each test set, we use the following standard evaluation metrics to compare the performance of the classifiers:

- *Precision*, the ratio of predicted positive cases, i.e., Twitter bots, that are indeed real positives: $\frac{TP}{TP+FP}$;
- *Recall* (also known as Sensitivity), the ratio of real positive cases that are indeed predicted as positives: $\frac{TP}{TP+FN}$;
- *Specificity*, the ratio of real negative cases, i.e., human accounts, that are correctly identified as negative: $\frac{TN}{TN+FP}$;
- *Accuracy*, the ratio of correctly classified users (both positives and negatives) among all the users: $\frac{TP+TN}{TP+TN+FP+FN}$;
- *F-measure*, the harmonic mean of Precision and Recall: $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$;
- *Matthews Correlation Coefficient (MCC)* [58], the estimator of the correlation between the predicted class and the real class of the users: $\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FN) \cdot (TP+FP) \cdot (TN+FP) \cdot (TN+FN)}}$

Each of the above metrics captures a different aspect of the prediction performance. *Accuracy* measures how many users are correctly classified in both of the classes, but it does not express whether the positive class is better recognized than the other one. Furthermore, there are situations where some predictive models perform better than others, even having a lower

TABLE I: Statistics about the datasets used for this work.

dataset	description	statistics	
		accounts	tweets
human accounts	verified accounts that are human-operated	3,474	8,377,522
social-bot-1	retweeters of an Italian political candidate	991	1,610,176
social-bot-3	spammers of products on sale at Amazon.com	464	1,418,626
test set #1	mixed set of 50% human accounts + 50% social-bot-1	1,982	4,061,598
test set #2	mixed set of 50% human accounts + 50% social-bot-3	928	2,628,181

accuracy. A high *Precision* indicates that many of the users identified as spambots are indeed real spambots, but it does not give any information about the number of spambots that have not been identified as such. This information is instead provided by the *Recall* metric: a low *Recall* means that many spambots are left undetected. *Specificity* instead measures the ability to identify human users as such. Finally, *F-Measure* and *MCC* convey in one single value the overall quality of the prediction, combining the other metrics. Moreover, *MCC* is considered the unbiased version of the *F-Measure*, since it uses all the four elements of the confusion matrix. Being a correlation coefficient, $MCC \approx 1$ means that the prediction is very accurate, $MCC \approx 0$ means that the prediction is no better than random guessing, and $MCC \approx -1$ means that the prediction is heavily in disagreement with the real class.

E. Results and Discussion

Table II shows the performance of our proposed neural model along with that of existing traditional techniques and algorithms reported on the cresci-2017 dataset. On test set #1, our *Recall* score outperforms the best, Cresci et al. [5], by 0.4% (absolute). On test set #2, our *F-Measure* surpasses the best, Cresci et al. [5] and Ahmed et al. [11], by 0.3% (absolute); the *Accuracy* score is the same as the best, Cresci et al. [5]. Most of our other scores are comparable to those of existing work.

As shown in Table II, Davis et al. [8], Yang et al. [6] and Miller et al. [12] achieve rather unsatisfactory results for the test set #1. The low values of *F-Measure* and *Mathews Correlation Coefficient* (MCC), respectively smaller or equal to 0.435 and 0.174, are mainly due to the low *Recall*. In turn, this represents a tendency of predicting social bots as genuine accounts. As for the test set #2, results in Table II show that Miller et al. [12] achieved the worst performances among all those that we have benchmarked in this study. Low values of both Precision and Recall mean incomplete and unreliable bot detection. As reported in Table II, our approach proved effective in detecting Twitter bot, with an *MCC* = 0.920 for test set #1 and *MCC* = 0.857 for test set #2, comparing to the other four systems, i.e., Davis et al. [8], Yang et al. [6], Miller et al. [12] and Ahmed et al. [11].

Our model outperforms the current state-of-the-art algorithm by Cresci et al. [5] on several metrics such as accuracy and F-measure (on test set #2) and recall (on test set #1). Although our model performs slightly below the algorithm in [5] on some other metrics, our model offers many significant advantages over [5]:

No handcrafted features required: Our model does not rely on any human-engineered features. The technique by Cresci et al., on the other hand, requires two large groups of (i.e., a set of six) user behaviour features and introduces a bio-inspired technique to model online user behaviors by so-called "digital DNA" sequences. The process of digital DNA fingerprinting has four main steps: (i) acquisition of behavioral data; (ii) extraction of DNA sequences; (iii) comparison of DNA sequences; (iv) evaluation. It is very time consuming and labour intensive to select and collect good handcrafted features.

No prior knowledge required: Our model does not require prior knowledge or assumptions about users' profiles, friendship networks, or historical behavior on the target accounts. We only rely on the textual contents of users' tweets. The technique by Cresci et al., on the other hand, requires both tweet type feature and tweet content feature, and thus feature engineering. It is expensive to collect, store and pre-process a large amount of data based on features. Our model can avoid these costs. Without feature engineering, our model can be implemented and deployed much faster and earlier than the other algorithms.

In order to gain more insight into the datasets, and thus the effectiveness of our proposed model, we generated a *word-cloud* for comparison of the most frequent words in the two datasets, i.e., human accounts and social-bot-3, as shown in Fig. 6. Word-cloud is a visualisation method that displays how frequently words appear in a given body of text, by making the size of each word proportional to its frequency. It is worth noting that Amazon social bots on Twitter usually prefer to use exaggerated words such as *Check awesome*, *Read Fascinating* and *Creative Writing*, to attract people's attention in order to advertise their products or services, or are talking about trends in a particular domain. Furthermore, a manual analysis of 100 randomly selected tweets in social-bot-3 showed that a majority of their tweets contains links to external web pages. This is in contrast to the general human accounts (in the random sample), which describe the accounts' owners using words such as *love, happy birthday, haha, lol, thank*, and *friend*, and most of whom seldom tweet links to external web pages.

Overall, the promising preliminary experimental results are yielded by the effective and efficient modeling ability of a deep bidirectional recurrent neural network architecture with the word embedding technique, as well as the availability

TABLE II: Performance comparison among various spambot detection techniques and algorithms reported on the cresci-2017 dataset.

technique	type	detection results					
		Precision	Recall	Specificity	Accuracy	F-Measure	MCC
test set #1							
Human annotators	manual	0.267	0.080	0.921	0.698	0.123	0.001
BotOrNot? [8]	supervised	0.471	0.208	0.918	0.734	0.288	0.174
C. Yang *et al.* [6]	supervised	0.563	0.170	0.860	0.506	0.261	0.043
ours	supervised	0.940	**0.976**	0.935	0.961	0.963	0.920
Miller *et al.* [12]	unsupervised	0.555	0.358	0.698	0.526	0.435	0.059
Ahmed *et al.* [11]	unsupervised	0.945	0.944	0.945	0.943	0.944	0.886
Cresci *et al.* [5]	unsupervised	**0.982**	0.972	**0.981**	**0.976**	**0.977**	**0.952**
test set #2							
Human annotators	manual	0.647	0.509	0.921	0.829	0.570	0.470
BotOrNot? [8]	supervised	0.635	**0.950**	0.981	0.922	0.761	0.738
C. Yang *et al.* [6]	supervised	0.727	0.409	0.848	0.629	0.524	0.287
ours	supervised	0.933	0.919	0.938	**0.929**	**0.926**	0.857
Miller *et al.* [12]	unsupervised	0.467	0.306	0.654	0.481	0.370	-0.043
Ahmed *et al.* [11]	unsupervised	0.913	0.935	0.912	0.923	0.923	0.847
Cresci *et al.* [5]	unsupervised	**1.000**	0.858	**1.000**	**0.929**	0.923	**0.867**

(a) human account

(b) social-bot-3

Fig. 6: Word-cloud comparison of human account dataset and social-bot-3 dataset.

of a large set of public annotated training data [55] that we used in this paper. Our proposed model outperforms the state-of-the-art algorithm [5] in several metrics and has the significant advantage of not using any feature engineering or prior knowledge. This will save time and money to select, collect, store and pre-process data. This advantage also enables faster and easier implementation and deployment of a bot detection scheme in real life.

V. CONCLUSION

This paper presents a RNN model, specifically BiLSTM, with word embeddings to distinguish Twitter bots from human accounts. Our model requires no prior knowledge or assumption about users' profiles, friendship networks, or historical behavior on the target account. To the best of our knowledge, our work is the first that develops a RNN model with word embeddings to detect bots that relies only on tweets and does not require heavy feature engineering. The preliminary simulation results are very encouraging. Experiments on the public dataset cresci-2017 show that our model can achieve similar performance compared with existing work, without handcrafted feature engineering, which is labor intensive and time consuming. This advantage allows for faster and easier implementation and deployment of the bot detection scheme. In addition, our proposed bidirectional recurrent neural architecture can be relatively easily adapted to a new problem, for example, using BiLSTM with word embeddings to detect phishing email, webpages or SMS.

REFERENCES

[1] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. N. Choudhary, "Towards online spam filtering in social networks." in *Proc. 19th Network and Distributed System Security Symp., (NDSS)*, 2012.

[2] S. Lee and J. Kim, "Warningbird: A near real-time detection system for suspicious urls in twitter stream," *IEEE Trans. Depend. Sec. Comput.*, vol. 10, no. 3, pp. 183–195, 2013.

[3] 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. Security Privacy*, 2011.

[4] H. Gao, Y. Yang, K. Bu, Y. Chen, D. Downey, K. Lee, and A. Choudhary, "Spam ain't as diverse as it seems: throttling osn spam with templates underneath," in *Proc. 30th Annu. Comput. Security Appl. Conf.*, 2014.

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

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

[7] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on twitter," in *Proc. Fifth Int. AAAI Conf. Weblogs Social Media*, 2011.

[8] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots," in *Proc. 25th Int. Conf. Companion on World Wide Web*, 2016.

[9] M. Alsaleh, A. Alarifi, A. M. Al-Salman, M. Alfayez, and A. Almuhaysin, "Tsd: Detecting sybil accounts in twitter," in *Proc. 13th Int. Conf. Mach. Learning and Appl.*, 2014.

[10] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. Eleventh Int. AAAI Conf. web social media*, 2017.

[11] F. Ahmed and M. Abulaish, "A generic statistical approach for spam detection in online social networks," *Computer Communications*, vol. 36, no. 10-11, pp. 1120–1129, 2013.

[12] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Information Sciences*, vol. 260, pp. 64–73, 2014.

[13] S. Yardi, D. Romero, G. Schoenebeck *et al.*, "Detecting spam in a twitter network," *First Monday*, vol. 15, no. 1, 2010.

[14] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Proc. Collaboration, Electronic Messaging, Anti-abuse Spam Conf., (CEAS)*, 2010.

[15] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi, "Understanding and combating link farming in the twitter social network," in *Proc. 21st Int. Conf. on World Wide Web, (WWW)*, 2012.

[16] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots+ machine learning," in *Proc. 33rd Int. ACM SIGIR Conf. Res. develop. inform. Retrieval*, 2010.

[17] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th annu. Comput. security Appl. Conf.*, 2010.

[18] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 363–374, 2011.

[19] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proc. 9th USENIX Conf. Networked Syst. Design Implementation*, 2012.

[20] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, and Z. M. Mao, "Innocent by association: early recognition of legitimate users," in *Proc. ACM Conf. Comput. Commun. Security*, 2012.

[21] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, "Social turing tests: Crowdsourcing sybil detection," *arXiv preprint arXiv:1205.3856*, 2012.

[22] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM Trans. Knowledge Discovery Data, (TKDD)*, vol. 8, no. 1, p. 2, 2014.

[23] Y. Liu, B. Wu, B. Wang, and G. Li, "Sdhm: A hybrid model for spammer detection in weibo," in *Proc. IEEE/ACM Int. Conf. Advances Social Networks Analysis Mining, (ASONAM)*, 2014.

[24] A. Paradise, R. Puzis, and A. Shabtai, "Anti-reconnaissance tools: Detecting targeted socialbots," *IEEE Internet Computing*, vol. 18, no. 5, pp. 11–19, 2014.

[25] S. Cresci, M. Petrocchi, A. Spognardi, M. Tesconi, and R. Di Pietro, "A criticism to society (as seen by twitter analytics)," in *Proc. IEEE 34th Int. Conf. Distributed Computing Systems Workshops, (ICDCSW)*, 2014.

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

[27] "Bot or not?" http://truthy.indiana.edu/botornot/, accessed: 2019-09-03.

[28] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proc. 22nd Int. Conf. World Wide Web, (WWW)*, 2013.

[29] M. Giatsoglou, D. Chatzakou, N. Shah, A. Beutel, C. Faloutsos, and A. Vakali, "Nd-sync: Detecting synchronized fraud activities," in *Proc. Pacific-Asia Conf. Knowledge Discovery Data Mining*, 2015.

[30] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Catching synchronized behaviors in large networks: A graph mining approach," *ACM Trans. Knowledge Discovery Data, (TKDD)*, vol. 10, no. 4, p. 35, 2016.

[31] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014.

[32] B. Viswanath, M. A. Bashir, M. B. Zafar, S. Bouget, S. Guha, K. P. Gummadi, A. Kate, and A. Mislove, "Strength in numbers: Robust tamper detection in crowd computations," in *Proc. ACM on Conf. Online Social Networks*, 2015.

[33] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Mining*, 2006.

[34] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++: A clustering algorithm for data streams," *J. Experimental Algorithmics, (JEA)*, vol. 17, pp. 2–4, 2012.

[35] N. Chavoshi, H. Hamooni, and A. Mueen, "Debot: Twitter bot detection via warped correlation." in *Proc. 16th Int. Conf. Data Mining, (ICDM)*, 2016.

[36] S. Cresci, M. Petrocchi, A. Spognardi, and S. Tognazzi, "On the capability of evolved spambots to evade detection via genetic engineering," *Online Social Networks and Media*, vol. 9, pp. 1–16, 2019.

[37] C. Grimme, D. Assenmacher, and L. Adam, "Changing perspectives: Is it sufficient to detect social bots?" in *Proc. Int. Conf. Social Comput. Social Media*, 2018.

[38] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Language Processing, (EMNLP)*, 2013.

[39] O. Irsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *Proc. Advances Neural Inform. Processing Syst., (NeurIPS)*, 2014.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing*, 2013.

[42] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Language Processing, (EMNLP)*, 2014.

[43] M. Xu, F. Wei, S. Watcharawittayakul, Y. Kang, and H. Jiang, "The yorknrm systems for trilingual edl tasks at tac kbp 2016." in *Proc. Text Analysis Conference, (TAC)*, 2016.

[44] M. Xu, N. Nosirova, K. Jiang, F. Wei, and H. Jiang, "Fofe-based deep neural networks for entity discovery and linking." in *Proc. Text Analysis Conference, (TAC)*, 2017.

[45] F. Wei, U. T. Nguyen, and H. Jiang, "Dual-fofe-net neural models for entity linking with pagerank," in *Proc. 28th Int. Conf. Artificial Neural Networks, (ICANN)*, 2019.

[46] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Investigating dynamic routing in tree-structured lstm for sentiment analysis," in *Proc. Conf. Empirical Methods Natural Language Processing, (EMNLP)*, 2019.

[47] S. Clinchant, K. W. Jung, and V. Nikoulina, "On the use of bert for neural machine translation," *arXiv preprint arXiv:1909.12744*, 2019.

[48] Y. Xu, Z. Lin, Y. Liu, R. Liu, W. Wang, and D. Meng, "Ranking and sampling in open-domain question answering," in *Proc. Conf. Empirical Methods Natural Language Processing, (EMNLP)*, 2019.

[49] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Advances Neural Infom. Processing Syst., (NeurIPS)*, 2013.

[50] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[51] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000.

[52] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Computational Linguistics: System demonstrations, (ACL)*, 2014.

[53] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[54] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *J. Mach. Learning Res., (JMLR)*, vol. 3, no. Aug, pp. 115–143, 2002.

[55] S. Cresci, "Mib datasets," http://mib.projects.iit.cnr.it/dataset.html, 2017.

[56] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools

for the arms race," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017.
[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[58] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.