IEEE Open Journal of the
Computer Society

Received 17 May 2023; accepted 24 July 2023. Date of publication 7 August 2023;
date of current version 31 August 2023. The review of this article was arranged by Associate Editor Milos Stojmenovic.

Digital Object Identifier 10.1109/OJCS.2023.3302286

Twitter Bot Detection Using Neural Networks and Linguistic Embeddings

FENG WEI [1] **(Student Member, IEEE), AND UYEN TRANG NGUYEN** [1] **(Member, IEEE)**

Department of Electrical Engineering and Computer Science, York University, Toronto M3J 1P3, Canada

CORRESPONDING AUTHOR: FENG WEI (e-mail: fwei@eecs.yorku.ca)

This work was supported by a Discovery Grant funded by the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT Twitter is a web application playing the dual role 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. In this article, we propose a Twitter bot detection model using recurrent neural networks, specifically bidirectional lightweight gated recurrent unit (BiLGRU), and linguistic embeddings. To the best of our knowledge, our Twitter bot detection model is the first that does not require any handcrafted features, or prior knowledge or assumptions about account profiles, friendship networks or historical behavior. The proposed model uses only textual content of tweets and linguistic embeddings to classify bot and human accounts on Twitter. Experimental results show that the proposed model performs better or comparably to state-of-the-art Twitter bot detection models while requiring no feature engineering, making it faster and easier to train and deploy in a real network. We also present experimental results that show the performance and computational costs of different types of linguistic embeddings and recurrence network variants for the task of Twitter bot detection. The results will potentially help researchers design high-performance deep-learning models for similar tasks.

INDEX TERMS Bot detection, linguistic embeddings, machine learning, neural networks, online social networks, Twitter bots.

I. INTRODUCTION

Twitter is a popular online social networking and micro-blogging tool. As a social network, Twitter's defining characteristic is its simplicity: 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, allow tweets to be grouped 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 by automated programs, known as bots. Automation is a double-edged sword for Twitter. On the one hand, legitimate bots generate a large volume of informative tweets, like news and blog updates. On the other hand, malicious bots spread spam or malicious content.

The Twitter bot problem has received substantial attention from researchers. Existing Twitter bot detection models generally rely on features obtained from user information such as profile [1], [2], [3], [4], timestamps [1], [2], [4], friendship [1], [4], behavior [5], [6], and network connection [2], [7], [8]. However, feature engineering is very time-consuming and labor-intensive.

In this article, we propose a recurrent neural network (RNN) model using BiLGRU and linguistic embeddings to distinguish Twitter bots from human accounts. We name the proposed model BOTLE (BOT detection using Linguistic Embeddings). We define the problem of Twitter bot detection as a text classification problem with a binary output: bot or non-bot. We use only the textual content of tweets as the input to our RNN model. To the best of our knowledge, our Twitter bot detection model is the first that does not require any handcrafted features; or prior knowledge or assumptions about account profiles, friendship networks or historical behavior.

Following are our contributions:

- We propose an RNN model to distinguish Twitter bots from human accounts. BiLGRU 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 (backward) and future (forward) states simultaneously. This method can effectively capture textual features across tweets and achieve competitive classification performance compared to existing state-of-the-art Twitter bot detection models.
- We use several types of linguistic embeddings to encode tweets, namely, word embeddings, character embeddings, part-of-speech embeddings, and named-entity embeddings. We avoid using handcrafted features, which require time-consuming and labor-intensive feature engineering. This advantage allows for faster and easier implementation and deployment of the bot detection scheme.
- We conducted experiments on real-world datasets. Experiments on the cresci-2017 dataset show that BOTLE performs better or comparably to state-of-the-art Twitter bot detection models [1], [2], [3], [5], [6], although our model does not require prior knowledge or assumptions about users' profiles, friendship networks, or historical behavior of the target accounts.
- We validate the effectiveness of various linguistic embeddings and recurrent variants for the task of Twitter bot detection. The results will potentially help researchers design high-performance deep-learning models for similar tasks.
- In addition to the classification performance, we also provide the running time (for training and inference) and memory usage of the proposed model, which show the model is feasible to deploy in real life.

The remainder of the article is organized as follows. We discuss related work in Section II. The proposed model 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. Sections V and VI validate the effectiveness of a variety of linguistic embeddings and recurrent variants, respectively, on the proposed neural model for Twitter bot detection. Section VII concludes the article and outlines our future work.

II. RELATED WORK

We discuss existing techniques of Twitter bot detection and datasets available for this task.

A. TWITTER BOT DETECTION METHODS

Existing methods of Twitter bot detection can be divided into two main approaches: supervised learning and unsupervised learning.

1) SUPERVISED LEARNING

Lee et al. [9] implemented 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 combinations.

The system by Yang et al. [2] provides a supervised machine learning classifier that infers whether a Twitter account is a human account or a bot by relying on relationships among accounts, tweet timestamps and level of automation. In addition, they designed ten new behavior-based features. According to their evaluation, the detection rate using their new feature set is significantly higher than that of existing work.

Alsaleh et al. [10] presented a system that utilizes feedforward neural networks (FFNN) to dynamically detect Twitter bot accounts. The classification results show satisfying detection rates. But it still requires complicated handcrafted features.

Davis et al. [1] grouped features into six main classes: network, user, friends, temporal features, content and sentiment, and employed random forest.

Varol et al. [4] 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.

Guo et al. [11] introduced, BGSRD, a model that symmetrically combines BERT [12] and graph convolutional network for Twitter bot detection. The model constructs a heterogeneous graph over the dataset and represents Twitter as nodes using BERT representations.

More recently, Feng et al. [7] introduced a bot detection framework that leveraged the topological structure of user-formed heterogeneous graphs. To evade detection, new emerging Twitter bots steal genuine users' tweets and dilute malicious content with benign tweets. Lei et al. [8] proposed a model that makes the text and graph modalities interactive and detects tweet semantic inconsistency. Periasamy et al. [13] introduced a transformer-based model for Twitter bot detection. Mohanty et al. [14] applied three boosting-based ensemble learning methods (i.e., adaptive boost, gradient boosting, extreme gradient boosting) to detect Twitter bots. Tan et al. [15] presented, BotPercent, a multi-dataset multi-model community-oriented bot detection pipeline to overcome generalization issues in existing individual-level models. BotPercent consists of two parts: training data and model architecture. Yang et al. [16] introduced, FedACK, a federated adversarial contrastive knowledge distillation framework for Twitter bot detection. The authors devise a GAN-based federated knowledge distillation mechanism for efficiently transferring knowledge of data distribution among clients. Chawla et al. [17] presented a hybrid technique utilizing digital DNA as a base approach and augmenting it with the state-of-the-art BERT model pre-trained on the sentiment classification task. Wu et al. [18] introduced, BotTriNet, a unified embedding framework that utilizes textual content posted by accounts for Twitter bot detection. The authors designed a triplet network to tune the raw embeddings, which are produced by traditional natural language processing techniques, for improving classification performance. Di et al. [19] introduced an algorithm to transform sequences of digital DNA into images and applied pre-trained convolutional neural networks (CNNs) models over the generated images for Twitter bot detection. DNA sequences are extracted from Twitter account profiles.

2) UNSUPERVISED LEARNING

The method by Miller et al. [3] considers vectors made of 126 features, extracted from both accounts and tweets, as input to modified versions of the DenStream [20] and StreamKM++ [21] clustering algorithms, to cluster feature vectors of a set of unlabeled accounts. The classifier by [6] 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 a Euclidean distance measure. Chavoshi et al. [22] 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 every day. Cresci et al. [5] proposed an unsupervised method to detect bots, by comparing their behavior with the aim of finding similarities between automated accounts. They introduced a bio-inspired technique to model online user behaviors using so-called "digital DNA" sequences. Extracting digital DNA for an account means associating that account with a string that encodes its behavioral information. Although it achieves good detection performance, 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 goal is to create bots that are difficult or impossible to detect. Cresci et al. [23] proposed the use of evolutionary algorithms to improve social bot skills. Grimme et al. [24] employed a hybrid approach involving automatic and manual actions to create bots that would be classified as human by a supervised bot detection system.

B. DATASETS FOR TWITTER BOT DETECTION RESEARCH

Cresci et al. [25] provided a widely used dataset for Twitter bot detection. It contains 3,474 human accounts and 10,894 bot accounts. The dataset is composed of both user information and tweets.

Feng et al. [26] introduced TwiBot-20, the first publicly available bot detection dataset that includes user follow relationships. The dataset contains 229,573 Twitter users, 8,723,736 user property items, 33,488,192 tweets and 455,958 follow links.

Feng et al. [27] presented TwiBot-22, a comprehensive graph-based Twitter bot detection dataset. It provides diverse entities and relations on the Twitter network, and has considerably good annotation quality. The dataset contains 1,000,000 Twitter users and 88,217,457 tweets.

For a more comprehensive review of Twitter bot detection methods/techniques, readers are referred to [28], [29], [30], [31], [32].

To the best of our knowledge, our work is the first that does not require sophisticated or labor-intensive handcrafted feature engineering but performs comparably to or better than state-of-the-art bot detection models/systems.

An earlier version of our neural model and preliminary results were reported in [33]. The earlier model and results in [33] were based on GloVe [34] word embeddings and BiLSTM. Compared with [33], in this article,

- we incorporated more linguistic embeddings into the new model, namely, character embedding, part-of-speech embedding, and named-entity embedding.
- we experimented with different recurrent variants, namely GRU, MGU, LSTM and LGRU, to select the best performer, which is bi-directional LGRU.
- we conducted extensive experiments to study the detection performance, running time and memory consumption of the different types of linguistic embeddings and recurrent variants.
- based on the above experiments, we selected BiLGRU for the bot detection task. Compared with the BiLSTM model presented in [33], the new model provides better detection performance, faster inference time and lower memory consumption.

III. OUR PROPOSED MODEL

In this section, we discuss different types of linguistic embeddings, BiLGRU, and our proposed Twitter bot detection model BOTLE.

A. LINGUISTIC EMBEDDINGS

We discuss different types of linguistic embeddings, namely, word embeddings, character embeddings, part-of-speech embeddings, and named-entity embeddings.

1) WORD EMBEDDINGS

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 tasks, including information retrievals [35], [36], [37], text classification [38], machine translation [39] and machine comprehension [40]. The success of word embedding [34], [41] 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 [42], part-of-speech tagging [43]) 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 an NLP model.

In this work, we adopt pre-trained GloVe [34] word vectors on Twitter. It is based on two billion tweets, including 27 billion tokens, with a vocabulary size of 1.2 million words. 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 [44] 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.

2) CHARACTER EMBEDDINGS

Kim et al. [45] introduced a character-level embedding to provide additional features in dealing with unknown or out-of-vocabulary (OOV) words in NLP tasks. Robust approaches for obtaining shape and morphological knowledge from words need to consider all characters of the word and choose the significant characteristics for the task to be carried out [46]. For instance, in the task of Twitter bot detection, abbreviations, spelling errors, and special characters ('#'), are very common. Moreover, important information can appear in different parts of a hashtag (e.g., "#ComingSoon", "#ILikePRODUCT"). Since character information can provide knowledge on structural differences between words, character embeddings are beneficial to a Twitter bot detection model from learning unknown and uncommon words [47]. Additionally, character embeddings can improve the detection performance using suffixes and prefixes such as '-able' and 'dis-', which carry semantic knowledge [48].

In this experiment, we incorporate convolutional-based character embeddings (CharCNN) [45] to our Twitter bot detection model. Specifically, let V be the vocabulary size of words, C be the vocabulary of characters. Given a word $w \in V$ composed of sequential characters $\{c_1, c_2, \ldots, c_n\}$, where n is the length of the word w, the character embedding of w is denoted as $C^w \in \mathbb{R}^{d\times n}$. A feature map $f^w \in \mathbb{R}^{n-h+1}$ is obtained after applying a convolution between C^w and a kernel $K \in \mathbb{R}^{d\times h}$ of width h. Then, max-pooling is performed over each feature map. Finally, the character-level representation of the word w is obtained as $E^w = [e^w_1, e^w_2, \ldots, e^w_k]$, with a total of k kernels. Following [49], we set k to 100 and h to 5 in this experiment.

Learned Character Embedding: According to [45], every CharCNN filter learns to identify certain character n-grams. In order to obtain a better visualization of what the character embedding is learning, the learned character representations via principal components analysis (PCA) are plotted as shown in Fig. 1. Each character is fed into the CharCNN and the CharCNN's output is used as the fixed dimensional representation for the corresponding character n-gram. From Fig. 1, we can observe that the learned model is able to distinguish between suffixes (grey), prefixes (blue), and others (red). Note that the embeddings are particularly sensitive to character n-grams containing hyphens (green) in that this is a solid indication of a word's part-of-speech property (e.g., hyphen used if the two words work together as an adjective before the noun they illustrate.) [45].

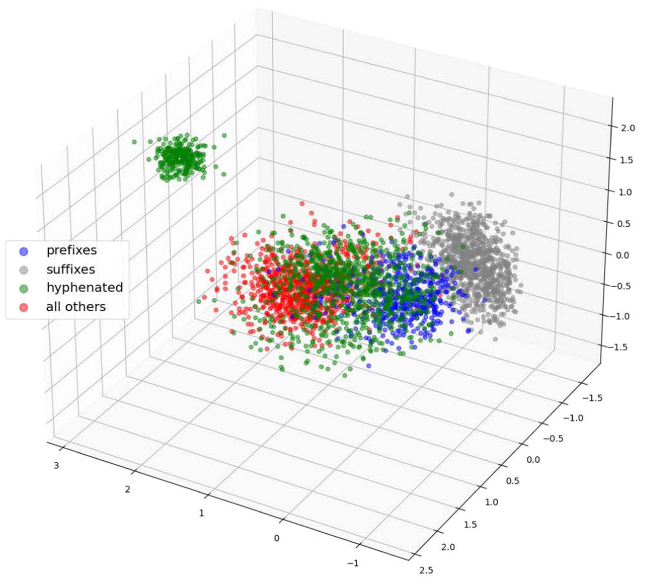


FIGURE 1. Plot of character n-gram representations via PCA. Colors correspond to prefixes (blue), suffixes (grey), hyphenated (green), and all others (red). Prefixes refer to character n-grams that start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.

3) PART-OF-SPEECH EMBEDDINGS

Words are typically ambiguous in their part of speech. In a sentence, the context of a word could address this vagueness [50]. For example, *work* can be either a noun or a verb. In the sentence *"There is plenty of work to be done in the garden."*, *work* can only be a noun.

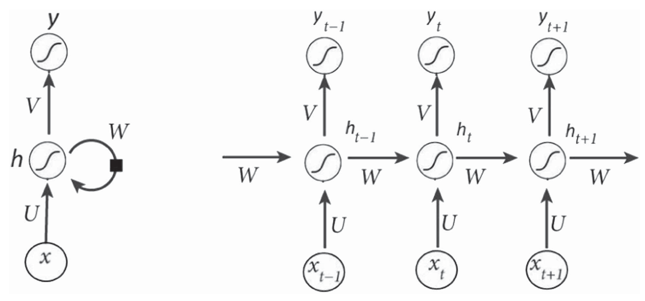
Part-of-speech (POS) tagging is a process that assigns the part of speech (e.g., adjective, verb) to words using their contextual information. In order to be effectively applied to neural networks, POS tags are converted into vectors of fixed length, and considered as trainable variables that are randomly initialized. Applications of POS embedding exist in many NLP tasks such as information extraction, and machine reading comprehension [51].

Following [51], we use 9-dimensional POS embedding for a total of 56 distinct types of POS tags.

TABLE 1. Example Illustrating Named-Entity Identification

PERSON [PER]	GEO-POLITICAL ENTITY [GPE]	ORGANIZATION [ORG]	MISCELLANEOU [MISC]

Michael Jordan [PER] played 13 seasons [MISC] with the Bulls [ORG] with a year and a half [MISC]
hiatus to play baseball in the Chicago [GPE] White Sox [ORG] minor league system.



(a) Recurrent neural network. (b) Recurrent neural network unfold.

FIGURE 2. (a) Recurrent neural network and (b) recurrent neural network unfold. Adapted from [53].

4) NAMED-ENTITY EMBEDDINGS

'Named entity' (NE) is a term widely used in natural language processing (NLP). A named entity refers to an entity such as a location, organization, product or person that can be named using a proper noun. Table 1 illustrates the necessity of identifying named entities in a wide variety of NLP tasks. In the absence of contextual information, it would be difficult or impossible to know that "Bulls" is an organization. Likewise, the term "White Sox" mentioned in the example is an organization, not a type of clothing.

Named-entity embedding of context words can improve the accuracy of machine reading comprehension [52]. The method of encoding named-entity tags is similar to that of POS tags. Named-entity tags are mapped into a fixed-size matrix, and considered as learnable variables in neural models.

Following [51], we use 8-dimensional named-entity embeddings for a total of 18 distinct types of named-entity tags.

B. BILGRU NEURAL NETWORKS

We briefly describe RNN, LGRU, BiLGRU, and then our proposed Twitter bot detection model.

RNN [54] is a class of artificial neural sequence models, as shown in Fig. 2, where connections between units form a directed cycle. It takes arbitrary embedding sequences $x = (x_1, \ldots, x_T)$ as input, and 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 the 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:

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

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.

There are several types of RNNs, such as long short-term memory (LSTM), gated recurrent unit (GRU), and minimal gated unit (MGU).

LSTM is a variant of RNN designed to deal with the vanishing gradients problem. Hochreiter and Schmidhuber [55] found that the LSTM, which uses purpose-built memory cells to store information, is better at finding and utilizing long-range context.

GRU [56] is introduced to avoid the issues of vanishing or exploding gradients suffered by a vanilla recurrent neural network. Compared to LSTM, GRUs have fewer memory cells in their architecture, which results in lower memory requirements. In GRUs, recurrent units adaptively capture the dependencies of various time scales. There are two gating units, namely, the reset gate and the forget gate, that modulate the flow of information inside the GRU. Unlike LSTM, GRUs do not have separate memory cells; furthermore, GRUs have no mechanism to control the degree to which its state or memory content is exposed, while LSTM can control how much memory content it wants to reveal [57].

Minimal Gated Unit (MGU) [58] is an alternative to GRU. MGU is composed of one gate, namely, the forget gate, and does not utilize the complex peephole connection [59]. This leads to fewer parameters compared to LSTM and GRUs. As a result, MGUs can obtain comparable classification performance but need less training time than GRUs for the tasks of sentiment analysis, image classification, and language modelling [58].

LGRU [60] is a recently introduced recurrent structure that is simpler than but as effective as LSTM. Like LSTM, LGRU relies on gating mechanisms to obtain long-term dependencies in sequential data:

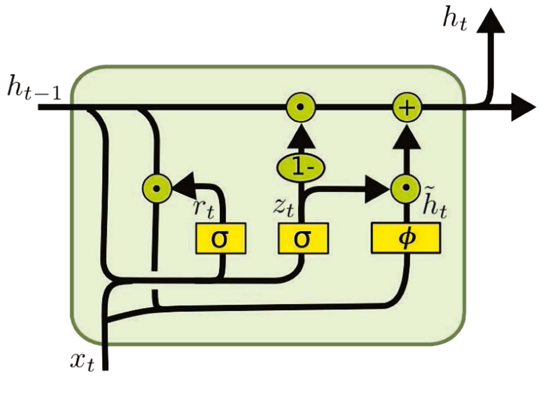
$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$\widetilde{h_t} = \phi(W_h[f_t \cdot h_{t-1}, x_t] + b_h) \tag{3}$$

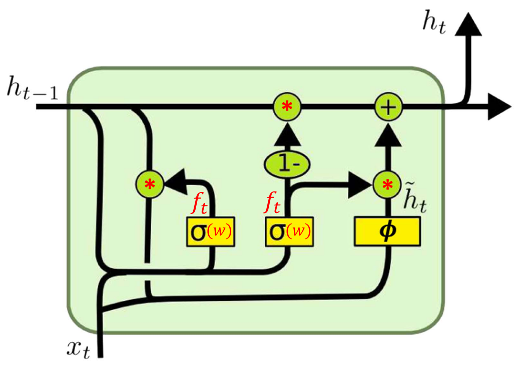
$$h_t = (1 - f_t) \cdot h_{t-1} + f_t \cdot \widetilde{h_t} \tag{4}$$

where f_t denotes forget gate; w_f is weight vectors of the network; the symbol $\cdot$ stands for scalar multiplication.

In the above equations, the update gate and the reset gate in GRU are substituted by a forget gate f_t in LGRU. Inspired by [61], we further simplify LGRU by i) transforming the



(a) GRU



(b) LGRU

FIGURE 3. Comparison of data flow and operations between (a) GRU and (b) our LGRU. The differences are highlighted in red.

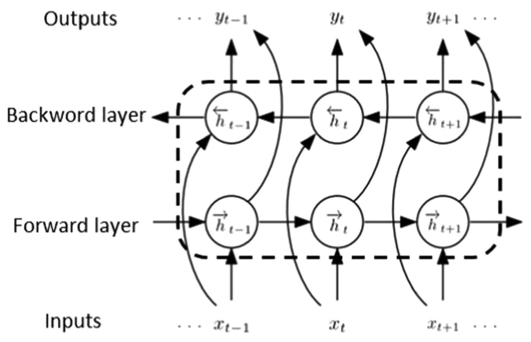


FIGURE 4. Bidirectional LGRU. Adapted from [62].

weight matrices of forget gates into vectors and ii) replacing element-wise multiplications with scalar multiplications. The motivation is that applying fewer learnable parameters in f_t and simplifying operations in $\widetilde{h_t}$ and h_t can result in lower complexity in training recurrent models without compromising the performance. As a result, LGRU is expected to be more lightweight than all other variants of GRU (including LSTM) while offering comparable performance. Fig. 3 illustrates the differences in data flows and operations between GRU and our simplified LGRU.

BiLGRU uses two LGRUs to learn each token of the sequence based on both the past and the future context of the token. As shown in Fig. 4, one LGRU processes the sequence from left to right; the other one, from right to left. At each time step t, a hidden forward layer with the 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 the hidden unit function $\overleftarrow{h}$ is computed based on the future hidden state $\overleftarrow{h}_{t+1}$ 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.

Our Proposed Model: As shown in Fig. 5, we make use of a fully connected softmax layer to output posterior probabilities over labels from two classes, standing for Twitter bots or humans. The input is a sequence of n tokens, $(x_1, \ldots, x_n)$. The predictions in both directions are modeled by three-layer BiLGRUs with hidden states $\overrightarrow{h}_{i,\ell}$ and $\overleftarrow{h}_{i,\ell}$ for input token x_i at the layer level $\ell = 1,\ldots, L$. 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:

$$\mathcal{L} = -\sum_{i=1}^{n}(logp(y|x_1, \ldots, x_n; \Theta_e, \overrightarrow{\Theta}_{LGRU}, \Theta_s) + logp(y|x_1, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{LGRU}, \Theta_s)) \tag{5}$$

IV. EXPERIMENTAL RESULTS

We first briefly describe the datasets in Section IV-A. In Sections IV-B, IV-C, and IV-D, the existing systems/models used for performance comparisons, settings, and evaluation metrics are described, respectively. Section IV-E presents experimental results, comparing the performance of the proposed model with that of existing state-of-the-art systems and analyzing the results in detail.

A. DATASETS

We evaluate our proposed model using the public annotated dataset cresci-2017 [63], consisting of 3,474 human accounts that generated 8.4 million tweets and 1,455 bots that generated 3 million tweets. We prepared two test sets following [25]. The 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 on Amazon.com. The test set #1 is composed of 1,982 accounts and 4,061,598 tweets, while the test set #2 is composed of 928 accounts

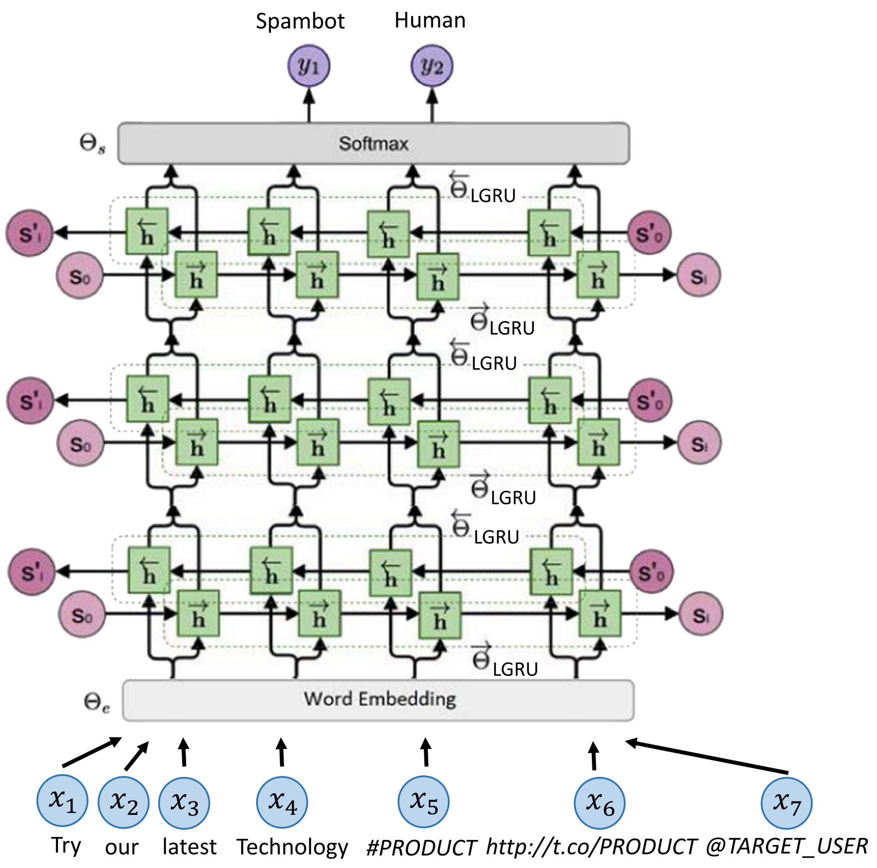

Spambot
Human
y_1
y_2
Θ_s
Softmax
$\overleftarrow{\Theta}$LGRU
$\overrightarrow{\Theta}$LGRU
Θ_e
Word Embedding
x_1 x_2 x_3 x_4 x_5 x_6 x_7
Try our latest Technology #PRODUCT http://t.co/PRODUCT @TARGET_USER

FIGURE 5. An illustration of the proposed BiLGRU neural model with word embeddings for distinguishing Twitter bots from human accounts.

TABLE 2. Statistics of the Datasets Used in the Experiments

Dataset	Description	Statistics	
		Accounts	Tweets
Human accounts	Verified accounts that are human-operated	3,474	2,839,361
Social-bot-1	Retweeters of an Italian political candidate	991	1,610,034
Social-bot-3	Spammers of products on sale at Amazon.com	464	1,418,557
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

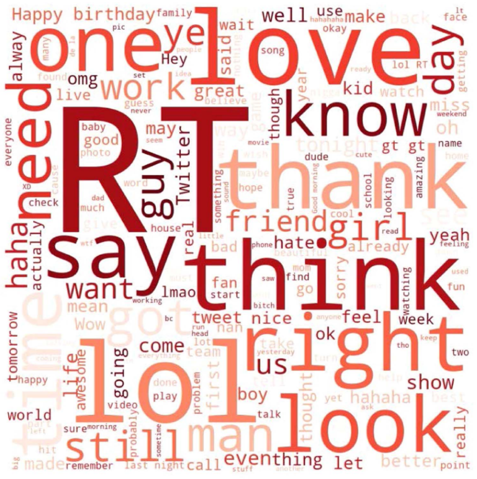
and 2,628,181 tweets. The statistics of datasets are shown in Table 2 in detail.

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 of appearing in the text. Note that Amazon social bots on Twitter usually 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. Furthermore, a manual analysis of 100 randomly selected tweets in social-bot-3 showed that a majority of their tweets contain links to external web pages. This is in contrast to the 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.

B. EXISTING SYSTEMS FOR COMPARISON

We compare the proposed model with the following baseline models that use the same dataset cresi-2017 [63]:

- Davis et al. [1] generate more than 1,000 features and group them into six main classes: network, user, friends, temporal feature, content and sentiment.


(a) human accounts

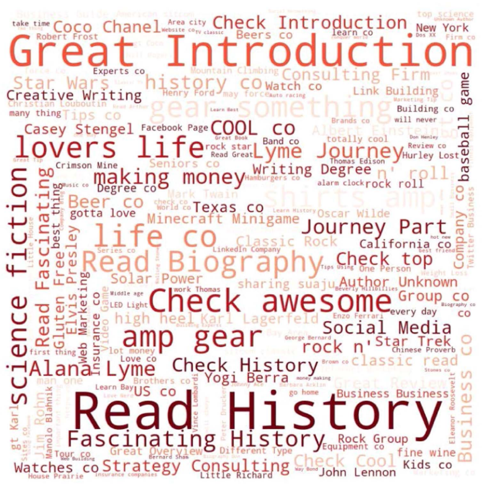

(b) social-bot-3

FIGURE 6. **Word-cloud comparison of (a) human account dataset and (b) social-bot-3 dataset.**

TABLE 3. **The Numbers of Handcrafted Features Used in Twitter Bot Detection Methods/Models**

Model	BotOrNot [1]	Miller et al. [3]	Yang et al. [2]	Ahmed et al. [6]	Cresci et al. [5]	BOTLE
Number of features	1000+	126	25	14	6	**0**

- Yang et al. [2] use 25 features and group them into six categories: profile-based features, content-based features, graph-based features, neighbor-based features, temporal features and automation-based features.
- Miller et al. [3] generate vectors made of 126 features, extracted from both user accounts and tweets.
- Ahmed et al. [6] use a set of 14 generic statistical behavior features related to URLs, hashtags, mentions and retweets.
- Cresci et al. [5] create two groups of user behavior features: tweet type DNA and tweet content DNA.

To obtain the classification performance of the above methods/systems for a comparison, we extracted the results reported Cresci et al. [5], which are deemed the best results the authors could achieve when they evaluated their proposed methods/systems.

Note that most state-of-the-art algorithms/systems for Twitter bot detection require a large number of data-demanding features. The above systems require feature engineering based on six features [5] to more than 1,000 [1], as listed in Table 3. Feature engineering is very labor-consuming and expensive in terms of data collection and pre-processing. Our proposed RNN model does not rely on any feature engineering and uses only the textual content of the tweets.

C. NEURAL NETWORK MODEL SETUP

Based on the design of the experiments, we tested several sets of parameters to select one that gave the experiments the best performance. We applied the grid search method to obtain the optimal combination of hyperparameters. These hyperparameters are as follows:

- *Learning rate:* the model is trained using the stochastic gradient descent [64] algorithm, while the learning rate is set to 0.01.
- *Network structure:* three stacked BiLGRU layers with 200 recurrent units and one fully connected softmax layer.
- *Dropout* [65] 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 Twitter bot detection model, we use four standard indicators:

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

For each test set, we use the four standard evaluation metrics, namely, precision, recall, accuracy, and F1-score, to compare the performance of the classifiers.

Precision and recall are two widely applied metrics to evaluate a Twitter bot detection system. If precision is low, the system requires manual verification by humans. It is labour intensive, if the number of false positives is high. However,

TABLE 4. Performance Comparison of Various Bot Detection methods/models Using the cresci-2017 Dataset

	Model	Type	Detection Result			
			Precision	Recall	Accuracy	F1-score
Test Set #1	Human annotators	Manual	0.267	0.080	0.698	0.123
	BotOrNot? [1]	Supervised	0.471	0.208	0.734	0.288
	Yang *et al.* [2]	Supervised	0.563	0.170	0.506	0.261
	Wei & Nguyen [33]	Supervised	0.940	0.976	0.960	0.958
	BOTLE	Supervised	0.956	0.977	0.969	0.967
	Miller *et al.* [3]	Unsupervised	0.555	0.358	0.526	0.435
	Ahmed *et al.* [6]	Unsupervised	0.945	0.944	0.943	0.944
	Cresci *et al.* [5]	Unsupervised	0.982	0.972	0.976	0.977
Test Set #2	Human annotators	Manual	0.647	0.509	0.829	0.570
	BotOrNot? [1]	Supervised	0.635	0.950	0.922	0.761
	Yang *et al.* [2]	Supervised	0.727	0.409	0.629	0.524
	Wei & Nguyen [33]	Supervised	0.933	0.919	0.929	0.926
	BOTLE	Supervised	0.938	0.934	0.939	0.936
	Miller *et al.* [3]	Unsupervised	0.467	0.306	0.481	0.370
	Ahmed *et al.* [6]	Unsupervised	0.913	0.935	0.923	0.923
	Cresci *et al.* [5]	Unsupervised	1.000	0.858	0.929	0.923

The Proposed Model BOTLE Uses LGRU and Four Linguistic Embeddings (Word, Character, POS, and NE). The Best and Second Best Results are Highlighted in Green and Blue, Respectively. The results of Human Annotators and of the Baseline Models BotOrNot?[1], Yang et al. [2], Miller et al. [3], Ahmed et al. [6], and Cresci et al. [5] are taken from [5]; the Results of Wei & Nguyen [33] are from [33].

if recall is low, the system fails to catch many bot accounts. Since the F1-score takes both false positives (precision) and false negatives (recall) into account, it reflects the balanced overall performance of the system.

E. RESULTS AND DISCUSSION

Table 4 shows the performance of our proposed neural model BOTLE along with that of the baseline models reported in [5]. Using the test set #1, BOTLE achieves a new recall score of 0.977, outperforming the best (Cresci et al. [5]) by 0.5 percentage points (0.977 vs. 0.972). Using the test set #2, BOTLE achieves a new F1-score of 0.936, surpassing the best (Cresci et al. [5] and Ahmed et al. [6]) by 1.3 percentage points (0.936 vs. 0.923); our accuracy score of 0.939 also outperforms the best (Cresci et al. [5]), by 1.0 percentage points (0.939 vs. 0.929). In addition, BOTLE achieves the second-best results in the other cases (highlighted in blue color in Table 4). In summary, BOTLE offers the best or second-best results in all cases.

BOTLE outperforms the current state-of-the-art system by Cresci et al. [5] on several metrics such as accuracy and F1-score (using the test set #2) and recall (using the test set #1). Although BOTLE performs slightly below Cresci et al. [5] in some cases, it offers many significant advantages over [5] as follows.

No handcrafted features required: BOTLE does not rely on any human-engineered features. Existing classifiers require feature engineering, ranging from six features (Cresci et al. [5]) to more than 1,000 (BotOrNot [1]). One of the best performers, Cresci et al. [5], still requires six features, namely, URLs, hashtags, media, entities, mentions, and plain text. Furthermore, the process of constructing bio-inspired "DNA" sequences in [5] is very time-consuming and labor-intensive.

No prior knowledge required: BOTLE does not require prior knowledge or assumptions about users' profiles, friendship networks, or historical behavior of the target accounts. It uses only the textual contents of users' tweets for classification. The method by Cresci et al. [5], nonetheless, requires prior knowledge such as replies or retweets to construct bio-inspired "DNA" sequences. It is expensive to collect, store and pre-process a large amount of data based on features. Without feature engineering, BOTLE can be implemented and deployed much faster and earlier than the other models.

V. EFFECTIVENESS OF LINGUISTIC EMBEDDINGS

For the results In Table 4, we use word embeddings, character embeddings, part-of-speech embeddings and named-entity embeddings. In this section, we validate the effectiveness of these linguistic embeddings by adding them one at a time to the bot detection model and measure the performance improvement as each type of embedding is added.

A. LINGUISTIC EMBEDDINGS: DETECTION PERFORMANCE

As shown in Table 5, we started with only word embeddings. We then added character, POS, and NE embeddings, one type at a time, in that order. Every time a type of embedding was added, we conducted the same experiments using test sets #1 and #2, and recorded the resulting precision, recall, accuracy and F1-score.

The results in Table 5 show that the detection performance is improved every time a new type of embedding was added. For example, using the test set #1, the accuracy is improved from 0.961 to 0.965, 0.966 and 0.969 as character, POS, and NE embeddings were added one by one, respectively. The same observation applies to the other performance metrics, and to test set #2 as well. This consistently demonstrates the

TABLE 5. Empirical Assessment of Different Linguistic Embeddings: Word, Character, Part-of-Speech (POS), and Named-Entity (NE) Embeddings

	Linguistic Embeddings				Precision	Recall	Accuracy	F1-score
	Word	Char	POS	NE				
Test Set #1	✓				0.943	0.974	0.961	0.958
	✓	✓			0.949	0.975	0.965	0.962
	✓	✓	✓		0.953	0.974	0.966	0.963
	✓	✓	✓	✓	0.956	0.977	0.969	0.967
Test Set #2	✓				0.934	0.919	0.930	0.927
	✓	✓			0.935	0.926	0.933	0.931
	✓	✓	✓		0.936	0.929	0.936	0.933
	✓	✓	✓	✓	0.938	0.934	0.939	0.936

All Models Use LGRU. The Best Results are Highlighted in Green. The Performance of the Bot Detector Improves as More Embeddings are Added to the Model.

TABLE 6. Empirical Assessment of Running Time and Memory Usage Incurred by the Linguistic Embeddings: Word, Character, Part-of-Speech (POS), and Named-Entity (NE) Embeddings

Linguistic Embeddings				Training time (minutes)	Average inference time per sample (milliseconds)	Memory usage (gigabytes)
Word	Char	POS	NE			
✓				2,257	12.6	4.5
✓	✓			2,551	18.2	5.2
✓	✓	✓		2,782	22.5	5.6
✓	✓	✓	✓	2,997	25.8	5.9

All Models Use LGRU.

TABLE 7. Performance of Four Recurrent Variants: LSTM, GRU, MGU, and LGRU

	Recurrent Variants	Precision	Recall	Accuracy	F1-score
Test Set #1	LSTM	0.952	0.977	0.967	0.965
	GRU	0.948	0.977	0.965	0.962
	MGU	0.949	0.975	0.964	0.962
	LGRU	0.956	0.977	0.969	0.967
Test Set #2	LSTM	0.936	0.931	0.937	0.934
	GRU	0.935	0.934	0.938	0.934
	MGU	0.933	0.934	0.937	0.933
	LGRU	0.938	0.934	0.939	0.936

Model Structures are All Bidirectional RNNs, but Each Model Uses a Different Recurrent Variant. The Models are Enhanced With All Four Embeddings: Word, Character, POS and NE. The Best Results are Highlighted in Green.

effectiveness of the additional types of linguistic embeddings. They enable the proposed model to encode rich linguistic and syntactic knowledge such as affixes, grammar, and part-of-speech.

On the other hand, incorporating additional linguistic embeddings will increase the run time and memory usage of a model. In the next section we examine the overheads incurred by additional linguistic embeddings in the proposed model BOTLE.

B. LINGUISTIC EMBEDDINGS: RUNNING TIME AND MEMORY USAGE

In this section, we report the runtime (for training and inference) and memory usage of BOTLE as we added character, POS, and NE embeddings, one type at a time, in that order, resulting in four linguistic embedding combinations: 1) word embeddings only; 2) word and character embeddings; 3) word, character and POS embeddings; and 4) word, character, POS and NE embeddings. For each combination (model), we conducted the same experiments using test set #2, and recorded the resulting run time and memory usage.

To obtain the training time, we trained a model with each of the above four combinations using 2,099,261 samples from the test set #2. To obtain the average inference time, we ran a model with each combination to classify 528,920 samples from the test set #2. The total inference time for classifying 528,920 samples was then divided by 528,920 to obtain the average inference time of one sample.

TABLE 8. **Empirical Assessment of Running Time and Memory Usage Incurred by Four Recurrent Variants LSTM, GRU, MGU, and LGRU**

Linguistic Embeddings				Training time (minutes)	Average inference time per sample (milliseconds)	Memory usage (gigabytes)
LSTM	GRU	MGU	LGRU			
✓				4,168	31.3	6.6
	✓			3,705	29.0	6.4
		✓		3,380	27.7	6.2
			✓	2,997	25.8	5.9

Model Structures are All Bidirectional RNNs, but Each Model Uses a Different Recurrent Variant. The Models are Enhanced With All Four Embeddings: Word, Character, POS and NE. The Best Results are Highlighted in Green.

To obtain the GPU memory consumption of a model, we profiled the job using NVIDIA NVML [66]. The NVIDIA library tracked and summarized the memory usage throughout the whole process of training and inference. All the experiments were run on a Linux workstation with the following configuration: NVIDIA GeForce GTX 1080 graphics card, Intel Core i7-6700 3.4 GHz processor, 12 GB DDR4 memory, and 1 TB solid state drive.

Table 6 illustrates the total training time, average inference time, and memory usage of the four linguistic embedding combinations. We observe that the run time and memory usage increase as more linguistic embeddings are added to the detection model. These increases are expected because they are the price to pay for higher classification performance (shown in Table 5) as more linguistic embeddings are incorporated into the classifier. These increases are acceptable for the task of Twitter bot detection on laptops, desktops or cloud services to achieve higher detection performance via the use of all types of linguistic embeddings. For example, using all four types of linguistic embeddings, the average inference time per sample is only 25.8 msec, and the combined memory consumption for training and inference is 5.9 gigabytes. (For other applications on low power, low memory devices, a trade-off analysis should be made to select the type(s) of linguistic embeddings optimal for both classification performance and resource constraints.)

VI. PERFORMANCE COMPARISON OF DIFFERENT RECURRENT VARIANTS

During the design process, we considered several recurrent variants, namely, long short-term memory (LSTM), gated recurrent unit (GRU), minimal gated unit (MGU) and lightweight gated recurrent unit (LGRU), which are described in Section III-B. We implemented and tested these variants and selected the best performer for our task, which is LGRU. In this section, we report the results of this selection process to help other researchers decide on a variant for similar tasks. The selection criteria are detection performance, run time and memory usage.

A. RNN VARIANTS: DETECTION PERFORMANCE

Table 7 shows the classification performance of the recurrent variants LSTM, GRU, MGU, and LGRU with all four embeddings (word, character, POS, and NE) on the cresci-2017 dataset. The other parameters are the same as those in Section IV-C. Overall, LGRU yields the best performance among the variants. Although the classification improvement of LGRU over the other variants seems marginal, LGRU really shines when it comes to run time and memory usage as discussed in the next section.

B. RNN VARIANTS: RUNNING TIME AND MEMORY USAGE

In this section, we report the run time (training and inference) and memory usage of the four recurrent variants LSTM, GRU, MGU and LGRU obtained from the same experiments discussed above in Section VI-A. The total training time, average inference time and memory usage were measured and collected as described in Section V-B and are shown in Table 8.

The training time of LGRU is 39%, 24%, and 12.8% shorter than that of LSTM, GRU and MRU (2,997 minutes vs. 4,168, 3,705 and 3,380 minutes), respectively. The average inference time of LGRU is 21.3%, 12.4% and 7.4% shorter than that of LSTM, GRU and MRU (25.8 msec vs. 31.3, 29, and 27.7 msec), respectively. LGRU also consumes less memory than LSTM, GRU and MRU: 11.9%, 8.5% and 5.1% less memory (5.9 gigabytes vs. 6.6, 6.4 and 6.2), respectively.

The above results demonstrate that LGRU is the best performer among the RNN variants. It provides the best of both worlds: higher classification performance and lower overheads in terms of memory usage and run time.

VII. CONCLUSION

This article presents an RNN model, specifically BiLGRU, with four linguistic embeddings (i.e., word, character, POS, and NE 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 an RNN model with several types of linguistic embeddings to detect bots that relies only on tweets and does not require any feature engineering. Experiments on the public dataset cresci-2017 show that our model performs similarly or better than existing work, but 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 BiLGRU with linguistic embeddings to detect phishing emails, spam webpages or spam SMS.

REFERENCES

[1] 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 World Wide Web*, 2016, pp. 273–274.

[2] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving Twitter spammers," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1280–1293, Aug. 2013.

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

[4] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Proc. 11th Int. AAAI Conf. Web Social Media*, 2017, pp. 280–289.

[5] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "DNA-inspired online behavioral modeling and its application to spambot detection," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 58–64, Sep./Oct. 2016.

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

[7] S. Feng, Z. Tan, R. Li, and M. Luo, "Heterogeneity-aware Twitter bot detection with relational graph transformers," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3977–3985.

[8] Z. Lei et al., "BIC: Twitter bot detection with text-graph interaction and semantic consistency, 2022, *arXiv:2208.08320*.

[9] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on Twitter," in *Proc. 5th Int. AAAI Conf. Weblogs Social Media*, 2011, pp. 185–192.

[10] 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. Learn. Appl.*, 2014, pp. 463–469.

[11] Q. Guo, H. Xie, Y. Li, W. Ma, and C. Zhang, "Social bots detection via fusing BERT and graph convolutional networks," *Symmetry*, vol. 14, no. 1, 2021, Art. no. 30.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[13] J. Periasamy, R. Srinidhi, and S. Srividhya, "A deep learning approach for Twitter bot detection," in *Proc. 1st Int. Conf. Comput. Sci. Technol.*, 2022, pp. 374–377.

[14] S. Mohanty et al., "Enhancing the detection of social bots on Twitter using ensemble machine learning technique," in *Proc. Int. Conf. Advance. Smart, Secure Intell. Comput.*, 2022, pp. 1–6.

[15] Z. Tan et al., "Botpercent: Estimating Twitter bot populations from groups to crowds, 2023, *arXiv:2302.00381*.

[16] Y. Yang et al., "FedACK: Federated adversarial contrastive knowledge distillation for cross-lingual and cross-model social bot detection," in *Proc. ACM Web Conf.*, 2023, pp. 1314–1323.

[17] V. Chawla and Y. Kapoor, "A hybrid framework for bot detection on Twitter: Fusing digital DNA with BERT," *Multimedia Tools Appl.*, vol. 82, pp. 30831–30854, 2023.

[18] J. Wu, X. Ye, and M. Y. Yuet, "BotTriNet: A unified and efficient embedding for social bots detection via metric learning," in *Proc. 11th Int. Symp. Digit. Forensics Secur.*, 2023, pp. 1–6.

[19] E. Di Paolo, M. Petrocchi, and A. Spognardi, "From online behaviours to images: A novel approach to social bot detection, 2023, *arXiv:2304.07535*.

[20] 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, pp. 328–339.

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

[22] N. Chavoshi, H. Hamooni, and A. Mueen, "DeBot: Twitter bot detection via warped correlation," in *Proc. 16th Int. Conf. Data Mining*, 2016, pp. 817–822.

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

[24] 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, pp. 445–461.

[25] S. Cresci, R. D. 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, pp. 963–972.

[26] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "Twibot-20: A comprehensive Twitter bot detection benchmark," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 4485–4494.

[27] S. Feng et al., "TwiBot-22: Towards graph-based Twitter bot detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 35254–35269.

[28] A. Derhab, R. Alawwad, K. Dehwah, N. Tariq, F. A. Khan, and J. Al-Muhtadi, "Tweet-based bot detection using Big Data analytics," *IEEE Access*, vol. 9, pp. 65988–66005, 2021.

[29] K. Hayawi, S. Saha, M. M. Masud, S. S. Mathew, and M. Kaosar, "Social media bot detection with deep learning methods: A systematic review," *Neural Comput. Appl.*, vol. 35, pp. 8903–8918, 2023.

[30] Z. Ellaky, F. Benabbou, and S. Ouahabi, "Systematic literature review of social media bots detection systems," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, 2023, Art. no. 101551. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157823000976

[31] R. Al-azawi and O. Safaa, "Feature extractions and selection of bot detection on Twitter A systematic literature review," *Inteligencia Artif.*, vol. 25, no. 69, pp. 57–86, 2022.

[32] E. Ferrara, "Twitter spam and false accounts prevalence, detection and characterization: A survey, 2022, *arXiv:2211.05913*.

[33] F. Wei and U. T. Nguyen, "Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings," in *Proc. 1st IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl.*, 2019, pp. 101–109.

[34] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[35] 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 Anal. Conf.*, 2016, pp. 1–7.

[36] M. Xu, N. Nosirova, K. Jiang, F. Wei, and H. Jiang, "FOFE-based deep neural networks for entity discovery and linking," in *Proc. Text Anal. Conf.*, 2017, pp. 1–8.

[37] F. Wei, U. T. Nguyen, and H. Jiang, "Dual-FOFE-net neural models for entity linking with PageRank," in *Proc. 28th Int. Conf. Artif. Neural Netw.*, 2019, pp. 635–645.

[38] 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 Lang. Process.*, 2019, pp. 3432–3437.

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

[40] 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 Lang. Process.*, 2019, pp. 2412–2421.

[41] T. Mikolov, I. Sutskever, K. Chen, G. S. 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.

[42] Z. S. Harris, "Distributional Structure," *Word*, vol. 10, no. 2/3, pp. 146–162, 1954.

[43] D. Jurafsky, *Speech & Language Processing*. Noida, India: Pearson Educ., 2000.

[44] 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. Comput. Linguistics: System Demonstrations*, 2014, pp. 55–60.

[45] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2741–2749.

[46] C. Dos Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1818–1826.

[47] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2Vec: Character-based distributed representations for social media," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 269–274.

[48] L. Verwimp et al., "Character-word LSTM language models," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2017, pp. 417–427.

[49] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension, 2016, *arXiv:1611.01603*.

[50] H. Schmid, "Part-of-speech tagging with neural networks," in *Proc. 15th Conf. Comput. Linguistics*, 1994, pp. 172–176.

[51] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension, 2017, *arXiv:1712.03556*.

[52] S. Liu, X. Zhang, S. Zhang, H. Wang, and W. Zhang, "Neural machine reading comprehension: Methods and trends," *Appl. Sci.*, vol. 9, no. 18, 2019, Art. no. 3698.

[53] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[54] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

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

[56] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, *arXiv:1412.3555*.

[57] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks, 2017, *arXiv:1801.01078*.

[58] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *Int. J. Automat. Comput.*, vol. 13, no. 3, pp. 226–234, 2016.

[59] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *J. Mach. Learn. Res.*, vol. 3, pp. 115–143, 2002.

[60] F. Wei and T. Nguyen, "A lightweight deep neural model for SMS spam detection," in *Proc. Int. Symp. Netw. Comput. Commun.*, 2020, pp. 1–6.

[61] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent," in *Backpropagation: Theory, Architectures, and Applications*, Mahwah, NJ, USA: Lawrence Erlbaum Associates, 1995, pp. 433–486.

[62] Y. Samih et al., "A neural architecture for dialectal arabic segmentation," in *Proc. 3rd Arabic Natural Lang. Process. Workshop*, 2017, pp. 46–54.

[63] S. Cresci, "MIB datasets," 2017. [Online]. Available: http://mib.projects.iit.cnr.it/dataset.html

[64] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Trans. Electron. Comput.*, vol. EC-16, no. 3, pp. 299–307, Jul. 1967.

[65] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors, 2012, *arXiv:1207.0580*.

[66] NVIDIA, "NVML API reference guide," 2022. [Online]. Available: https://docs.nvidia.com/deploy/nvml-api/index.html

FENG WEI (Student Member, IEEE) is currently working toward the Ph.D. degree with the Department of Electrical Engineering and Computer Science, York University, Toronto, ON, Canada. His research interests include machine learning, cybersecurity, natural language processing, and computer vision.

UYEN TRANG NGUYEN (Member, IEEE) received the Bachelor of Computer Science and Master of Computer Science degrees from Concordia University, Montreal, QC, Canada, and the Doctoral degree in computer science from the University of Toronto, Toronto, ON, Canada. She is currently an Associate Professor with the Lassonde School of Engineering, York University, Toronto, ON, Canada. Her research interests include wireless networking, mobile computing, online social networking, information security and financial technology. Her research has been funded by NSERC (National Sciences & Engineering Research Council of Canada), MITACS (Mathematics of Information Technology and Complex Systems), York University and industry partners. She is a frequent reviewer of grant applications for NSERC, MITACS, the U.S. National Science Foundation, and the National Science Centre of Poland.