Malicious Bot Detection in Online Social Networks: Arming Handcrafted Features with Deep Learning

Guanyi Mou[0000−0002−9987−0342] and Kyumin Lee[0000−0002−9004−1740]

Worcester Polytechnic Institute, Worcester MA 01609, USA
{gmou,kmlee}@wpi.edu

Abstract. Online social networks (OSNs) have long been suffering from various types of malicious bots (e.g., spammers, fake followers, social bots, and content polluters). Recent studies show that they have also been actively involved in delivering hate speeches and disseminating misinformation. Over several years, researchers have proposed multiple approaches to identify some types of them to lower their impact on the OSNs. However, their strategies mostly focused on handcrafted features to capture characteristics of malicious users, or their deep learning approaches may only work under certain situations (e.g., under the dense retweets/sharing behavior). To overcome the limitation of the prior work, in this paper, we propose a novel framework that incorporates handcrafted features and automatically learned features by deep learning methods from various perspectives. It automatically makes the balance between them to make the final prediction toward detecting malicious bots. In particular, we (i) combine publicly available 15 Twitter user datasets and categorize these accounts into two groups (i.e., legitimate accounts and malicious bot accounts); and (ii) propose a deep learning framework that jointly learns various features and detects malicious accounts. Our experimental results show that our proposed model outperforms 7 state-of-the-art methods, achieving 0.901 accuracy. Our ablation study shows that all types of our features positively contribute to enhancing the model performance.

Keywords: Malicious bot detection · Deep learning.

1 Introduction

Malicious bots have misused the power of Online social networks (OSNs) such as Twitter, Facebook, and Weibo, continuously caused significant disturbance to the overall online social environment, and shaped unhealthy trends, bias, and misbelief in societies (e.g., COVID-19 related misinformation [27]). Their accounts[1] have made severe impact and damage to the OSNs by causing inconveniences, intensifying contradictions, and aggravating prejudices [1].

[1] We use terms user and account, interchangeably.

Despite the long history of causing ongoing negative impact, malicious bots did not quit on being the Grand Villain on the OSNs. They have been emerging, evolving, and participating in new types of destructive activities. Reports[2] and analysis of malicious bots involved in hate speech dissemination [2, 44] and fake news propagation [36, 37] show their seemingly ever-lasting significant impact. Efficiently and accurately detecting them is still a crucial problem.

In recent years, OSN service providers established policies for warning, blocking, and suspending malicious accounts[3]. According to our study described in Appendix A.1, some of these malicious accounts are still alive for years without any suspension or proper treatment.

Researchers have proposed approaches to detect specific types of malicious bots [6, 10, 14, 18, 30]. Even though these approaches identified some malicious bots, we are still facing new challenges with new malicious bots such as hashtag promoters and social spambots, especially, political bots and even extremists such as ISIS recruiters [3,4]. Most of the existing frameworks identify new groups of useful handcrafted features and then apply them to traditional machine learning classifiers for satisfying results. They are thus placing the performance and robustness on an intuitively vulnerable position, as manipulators can play with those handcrafted features and deploying direct adversarial attacks against them.

Deep learning techniques, however, were less addressed in this domain. To the best of our knowledge, existing deep learning frameworks for malicious bot detection are, to some extent, limited in analyzing and using some particular perspectives of OSN accounts, usually only focusing on capturing temporal patterns [11, 33] or simple single tweet patterns [28].

To fill this gap, we propose a unified deep learning framework, which analyzes both temporal patterns and posting contents, and also incorporates handcrafted features. There are a few challenges. First, how to collect information for various types of malicious bots? Second, how can we extract features which distinguish between malicious users and legitimate users? Third, how can we create a unified framework that is capable of effectively detect malicious bots?

By keeping these challenges in mind, in this paper, we combine publicly available Twitter datasets, which contain accounts of content polluters, fake followers, traditional spambots, social spambots, and legitimate users. Then, we extract handcrafted features, and automatically learned features by deep learning methods. Finally, we combine both features and make a balance between them toward building a malicious bot detection model.

In this paper, we make the following contributions:

- We propose a novel joint learning framework that is capable of detecting various malicious bots altogether and distinguishing them against legitimate accounts. It combines both handcrafted features (i.e., profile and activity features and LIWC-based personality features) and automatically learned features (i.e., temporal behavior related features and text related features).

[2] https://bit.ly/39mGlnm and https://bit.ly/3hlpt38
[3] https://help.twitter.com/en/rules-and-policies/twitter-rules

- Our model outperforms 7 state-of-the-art methods, and we analyze intuitively and logically for the good performance.
- We conduct an ablation study, which shows that all components/feature types positively contribute to our proposed model's performance.

2 Related Work

Specific types of malicious bots were studied in the past. Some researchers focused on analyzing and detecting content polluters and spammers in OSNs [7,21,30,34]. Their classification methods focused on different perspectives such as temporal patterns of behaviors [10], social networks [6] and others [21,29,35]. DARPA held a twitter bot challenge [38] for better understanding and detecting bots. Davis *et al.* [18] proposed a framework *BotOrNot* (later on evolved and renamed as *Botometer*) which was trained on a dataset of malicious users. Adewole *et al.* [1] made a thorough review of 65 bot detection papers. Alfifi *et al.* [3,4] studied the behavior of long-lived and eventually suspended Arabic Twitter accounts in social media (especially ISIS accounts), and also discussed the level of automatic/botness of these accounts. Their results showed that the percentage of automated posting behaviors was relatively high. There were also researches revealing that malicious bots were involved in trending topics by disseminating misinformation and hate speech [2,36,37,44].

In malicious bot detection architectures, researchers have developed many traditional machine learning models [12, 31]. Cresci *et al.* [14, 17] proposed a DNA inspired model that produced a relatively good result without much detailed information from users. Yang *et al.* [43] proposed methods from another perspective, where they enhanced the performance with data selection.

To the best of our knowledge, although models relying on handcrafted features provide somehow compelling and convincing results in their previous experiments, they may face two vital challenges: 1) *Handcrafted features are not entirely scalable:* as the number of proposed features is increasing; it is getting harder to discover novel and helpful features. Thus the performance improvment/gain is reduced as researchers add additional handcrafted features into their models. For example, the *Botometer* [18] used more than one thousand handcrafted features. It is intuitively way too challenging for human intelligence to come up with new ways/ideas of inventing additionally useful features. Many researchers thus turn to focus on more complicated features such as the features extracted from network information based on trust propagation theories. The trend here also went down deeper and thus more computation consuming: for example, to achieve better performance, Beskow *et al.* [6] categorized information into four tiers, and eventually used all of them. The last tier involved using friends' timeline information. This network information is large in size and collection time, so not being feasible in many practical cases. 2) *Handcrafted features provided a clear target for adversarial attacks:* Manipulators of malicious bots can play with their profiles, posting contents and patterns to reduce distinguishing power of the handcrafted features and avoid the detection. Thus, the

arms race might get harder for classifiers and their robustness gets decreased as malicious bots change their behavior [41].

Some deep learning frameworks are applied in learning from temporal behavior information [11, 33] based on the assumption that malicious bots' behaviors are hard to hide as automated mechanisms follow the designed patterns that are relatively regular and not as random as human. Other deep learning frameworks are applied in learning from text information [28]. But, the deep learning frameworks do not focus on enough scope of the whole picture of each user. Some frameworks require the datasets to satisfy specific properties, to enable them to come up with effect: for example, *RTBust* [33] focuses on the retweet-tweet patterns of user timelines, so if the user's retweet counts are way too sparse, the framework will intuitively not work well. We show the performance of this model as a baseline in the experiment section. As certain special-purpose bots, such as fake followers, may not have enough posting timestamps to reveal non-human-like patterns, deep learning models (e.g., Chavoshi *et al.* [11]) only encoding temporal patterns would not work well. Overall, the performance of existing frameworks, which rely on learned features, often cannot reach as high accuracy as handcrafted features. We conjecture one of the main reasons is that those frameworks did not include wide scope of user information.

Our framework differs from the prior researches in the following ways: (1) We combine the advantages of both handcrafted features and automatically learned features via deep learning, thus being scalable and performance promising; (2) Under the limited information in the publicly available datasets, we use only limited user profile information, posting content information, and posting timestamps, while not expanding to the expensive network information; and (3) We design our framework in learning both temporal posting features and language model features, and the mechanism in handling the balance of these features to achieve desirable performance.

Table 1. Dataset Status.

Source	Lee	Cresci	Cresci	Gilani	Varol	Cresci	Midterm	Botometer	Botwiki	Celebrity	Cresci	Political	Pronbots	Vendor	Verified	Sum
Year	'11	'15	'17	'17	'17	'18	'18	'19	'19	'19	'19	'19	'19	'19	'19	
Legit	18537	-	1077	1140	1343	5269	7027	61	-	1293	296	-	-	-	1898	37941
Bot	17241	4685	5465	970	665	5875	25	18	98	-	269	13	1568	605	-	37497
Sum	35778	4685	6542	2110	2008	11144	7052	79	98	1293	565	13	1568	605	1898	75438

3 Dataset and Account Types

We chose Twitter as the primary social networking site in this study because Twitter has a generous data sharing policy to the third party researchers, and some public datasets with rich information are available. But, our proposed framework is generally applicable to other social networking sites with minor modification. To avoid potential bias and subjective labeling caused by collecting and labeling data by ourselves and include various types of malicious bots, we used 15 Twitter benchmark datasets from Botometer's repository[4]. They come from various sources [13–16, 23, 30, 33, 39, 42, 43] and contain many types of users (content polluters, fake accounts, traditional & social spambots, stock

[4] https://botometer.iuni.iu.edu/bot-repository/datasets.html

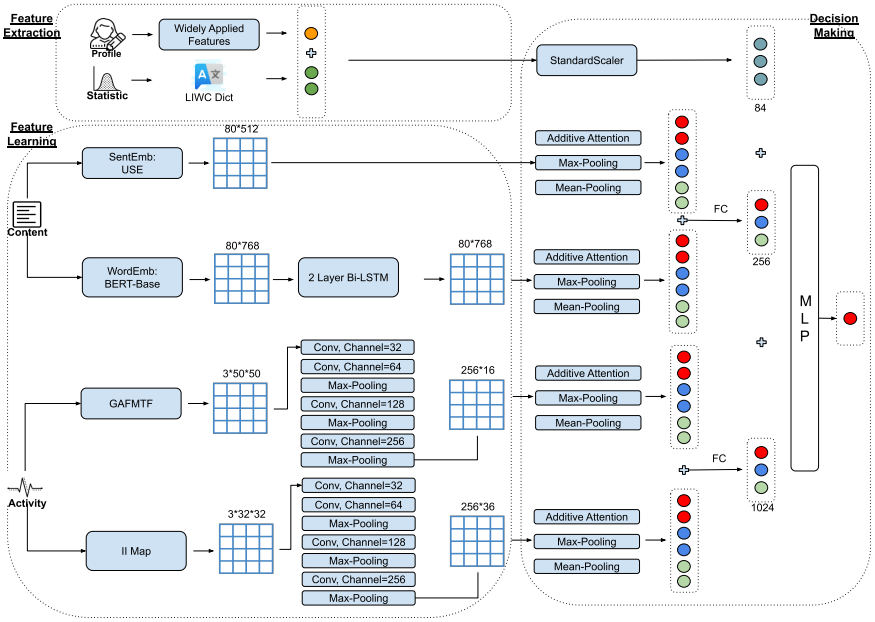

Fig. 1. Our overall framework. *FC* stands for fully connected layer.

related bots, political bots, fake followers, verified accounts, celebrity accounts, legitimate users, etc.). Each dataset's name as well as their original user types are described in Appendix A.2.

In the datasets, *Lee'11*, *Cresci'15*, and *Cresci'17* included the original profile information, posted tweets, and timestamps, while the other datasets only included the user IDs on Twitter. Thus we kept the above mentioned three datasets' original data and collected the other datasets' user information via Twitter API in April 2020. We grouped legitimate users, verified accounts and celebrity accounts as legitimate, while other types of accounts as malicious bots.

We then filtered out accounts with inconsistent labels, no posting information, or non-English posting. Table 1 shows the final detailed number of accounts that we used for the experiment. Some datasets originally only contain one type of user accounts (e.g., Vendor'19 and Botwiki'19). Overall, our final dataset consists of 37,941 legitimate accounts and 37,497 malicious bots. The dataset is thus almost naturally and perfectly balanced. The alive malicious bots that we successfully fetched from Twitter API are intuitively harder to detect as they managed to evade the platform's detection mechanisms. In other words, detecting these long-surviving malicious bots is a hard and important problem.

4 Our Framework

In this section, we introduce our framework, which combines handcrafted features and learned features through deep neural networks. We show a general view of our framework in Figure 1. The framework is composed of three parts:

- ***Feature Extraction***: handles with handcrafted features.

Table 2. Features and their notations.

Feature Type	Notation	Description
Traditional Features (handcrafted)	t	|Number of tweets posted by user|
	t/d	|Average tweets(posts) posted per day|
	dd	|Days since account creation|
	ut/t	Unique tweet(post) ratio
	h/t	|Hashtags posted per tweet(post)|
	uh/t	|Unique hashtags posted per tweet(post)|
	m/t	|Mentions posted per tweet(post)|
	um/t	|Unique mentions posted per tweet(post)|
	l/t	|Links (URLs) posted per tweet(post)|
	ul/t	|Unique links (URLs) posted per tweet(post)|
	rt/t	|Retweet posted per tweet(post)|
	$len(sn)$	Length of screen name
	$len(des)$	Length of description
	fer	# Followers
	$fing$	# Followings
	fav	# Favorites
	$fing/d$	|New followings per day|
	fer/d	|New followers per day|
	ff	Following follower ratio
	cr	Content compression ratio
LIWC Features (handcrafted)	-	64 LIWC features
Text Features	-	Sentence level embedding
	-	Word level embedding + Bi-LSTM
Temporal Behavior	-	Inter-posting-time difference pair features
	-	GASF, GADF, MTF features

- ***Feature Learning***: learns useful embeddings from multiple perspectives.
- ***Decision Making***: combines all the features and embeddings together and makes the final prediction.

We show some of the hyper-parameters in the figure. The detailed settings are described in Section 5. Table 2 presents a list of our extracted/learned features.

4.1 Feature Extraction

We extract handcrafted features in two ways:

We first extracted 20 widely used handcrafted features (called traditional features) from each account. They have been proven to be useful, and positively contributed to the performance of models in the literature [30, 39]. Those features mainly come from user profile information, and some counting based frequency/ratio from user posting contents. As user profile information is naturally categorized, it is intuitively straightforward to use handcrafted features for better measurement. For unique posts, we first translate all links to the same word "URL", anonymize mentions, hashtags, and special tags, and then count the number of unique posts based on these transformed data. For the compression ratio of user tweets, we used Python's zip package with its default zip setting. For extreme cases where a user has no follower, we adopt the #Followings as the ff ratio. We transform the existing/living seconds of accounts into the unit of days in floating number, and as described above, we guarantee every account

in our dataset has at least one posting record. Thus the ratios of the features will not face NaN or Inf problems.

We also extracted Linguistic Inquiry and Word Count (LIWC)[5], a dictionary for text analysis and personality analysis. It categorizes words into each meaningful types/groups. For each account in our dataset, we concatenate their tweets, count the number of meaningful words, and then calculate the occurrence of words belonging to each category. We naturally treat the proportion of these occurrences as features. Thus, the number of features we extract from LIWC equals to the number of categories of LIWC. We extracted 64 features by using the LIWC 2007 dictionary. From a high level of view, LIWC features capture the general statistics of each user's profile, activities, and their preferences in terms of word usage. Malicious bots serve for different purposes against legitimate accounts. Since malicious bots disseminate specially designed messages, these features may reveal the difference.
The aforementioned ones sum up to 84 handcrafted features.

4.2 Feature Learning

We automatically learned useful embeddings/features from posting contents and temporal behaviors by using neural networks. Thus, the learning part is divided into two components: *Text Embeddings* and *Temporal Behavior Embeddings*.

Text Embeddings Given strings $S = [s_1, s_2, \ldots, s_m]$, we encode them into 2D matrices $TextEmb(S) = [Encoder(s_i)]$, where i=*1, 2, ..., m*. Each string in this context is a tweet's content. The Encoder differs in the following two ways:

1. **Sentence level embeddings (*SentEmb*)**: For all the tweets posted by the same user, each tweet is encoded by a fixed-length vector, which represents its general information. A sentence embedding of each tweet would be $SentEmb(s_i) = v_i$, which is a 1D vector. By stacking all tweet vectors together, we derive a 2D matrix that contains all high-level information of user postings. These postings do not necessarily have any sequential relatedness, as each one is a unique sentence embedding.

2. **Word-level embeddings (*WordEmb*)**: For all tokens in all tweets, each word can be encoded by another fixed-length vector, which represents its semantic meaning. Word embeddings of each tweet would be $WordEmb(s_i) = W_i$, which is a 2D matrix. By concatenating all 2D matrices together, we also derive a 2D matrix. This matrix does contain sequential information as words' semantics are connected. Together they form unique synthetics.

SentEmb contains high-level information and thus can be used for possible sentence-level similarity comparison or repeated pattern learning. In contrast, *WordEmb* contains more detailed, more abundant information, which may help learn the difference between malicious or legitimate accounts in terms of frequent word usage difference, synthetic structure difference, sentiment difference, etc. By combining both embeddings, we derive better text representation. We used Universal Sentence Encoder (USE) [9] for SentEmb and BERT-Base [19] + Bi-LSTM for WordEmb representation.

[5] http://liwc.wpengine.com/

Temporal Behavior Embeddings Given the sequences of each user's posting timestamp $T = [t_0, t_1, \ldots, t_n]$, we applied two methods for mapping these sequences into 3D images: **GAFMTF** [40] and **II Map** [11] for informative pattern recognition. Then, we design two convolutional neural networks for learning features out of them.

1. **GAFMTF**: *Gramian Angular Field (GAF)* methods were proposed for a better encoding of timeseries in polar coordinate system than Cartesian coordinate representations. The authors [40] claimed their mapping method is invertible and also preserves absolute temporal relations. We first encode time difference sequences $TD = [t'_1, t'_2, \ldots, t'_n]$ into polar coordinates, where $t'_i = t_i$ - t_{i-1}. Then, we map it into a 2D plane. The GAF method provides summation-graph (GASF) and difference-graph (GADF),

$$GAF(TD) = \begin{bmatrix} < \widetilde{t'_1}, \widetilde{t'_1} > & < \widetilde{t'_1}, \widetilde{t'_2} > & \cdots & < \widetilde{t'_1}, \widetilde{t'_n} > \\ < \widetilde{t'_2}, \widetilde{t'_1} > & < \widetilde{t'_2}, \widetilde{t'_2} > & \cdots & < \widetilde{t'_2}, \widetilde{t'_n} > \\ \vdots & \vdots & \ddots & \vdots \\ < \widetilde{t'_n}, \widetilde{t'_1} > & < \widetilde{t'_n}, \widetilde{t'_2} > & \cdots & < \widetilde{t'_n}, \widetilde{t'_n} > \end{bmatrix}$$

$$< \widetilde{t'_i}, \widetilde{t'_j} >= \widetilde{t'_i} \cdot \widetilde{t'_j} - \sqrt{1 - \widetilde{t'_i}^2} \cdot \sqrt{1 - \widetilde{t'_j}^2}, \text{ for GASF}$$

$$< \widetilde{t'_i}, \widetilde{t'_j} >= \sqrt{1 - \widetilde{t'_i}^2} \cdot \widetilde{t'_j} - \widetilde{t'_i} \cdot \sqrt{1 - \widetilde{t'_j}^2}, \text{ for GADF}$$

and $\widetilde{t'_i}$ is t'_i in polar coordinates.

Markov Transition Field (MTF) method analyzes the timeseries change from another perspective: the transition probability of value changes across the time series. The authors also reported its effectiveness, especially stacked with GAF outcome matrices.

We first split TD into Q quantile bins (here we set $Q = 2$), and then assign t'_i into corresponding quantile bins $q_i, i \in Q$. Next, we measure the transition probability of sequence value change in terms of different bins.

$$MTF(TD) = \begin{bmatrix} w_{ij|t'_1 \in q_i, t'_1 \in q_j} & \cdots & w_{ij|t'_1 \in q_i, t'_n \in q_j} \\ w_{ij|t'_2 \in q_i, t'_1 \in q_j} & \cdots & w_{ij|t'_2 \in q_i, t'_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij|t'_n \in q_i, t'_1 \in q_j} & \cdots & w_{ij|t'_n \in q_i, t'_n \in q_j} \end{bmatrix}$$

where w_{ij} is a transition probability of t'_{k+1} in quantile bin q_j, given t'_k in quantile bin q_i, for all $k \in N$.

We stacked 3 2D matrices(i.e., GASF, GADF & MTF) to produce a 3D matrix.

2. **II Map**: Unlike GAFMTF as a general sequence encoding method, this method was explicitly proposed for bot detection. This method tends to focus more on the pairwise sequence neighborhood pattern.

$IPT(T, lag) = [(t_0, t_{0+lag}), (t_1, t_{1+lag}), \cdots, (t_{n-lag}, t_n)]$,

where pairs of tweet posting timestamp differences are mapped into 2D planes. Upon the control of different "lag" values, we also come up with a 3D matrix. In this paper, we set $lag = 1, 2, 3$. We can interpret this method as inter-posting-time (IPT) mapping or IPT embedding.

Given the generated two 3D matrices (by *GAFMTF* and *II Map*) as inputs, we design two similar, but independent convolutional neural networks for

learning useful features. Each convolutional neural network consists of four convolutional layers and three max-pooling layers as shown in Figure 1. In all convolutional layers, the filter size is 3*3 with $stride = (1, 1)$ and $padding = (1, 1)$. In all max-pooling layers, the window size is 2*2 and $stride = (2, 2)$. We used batch normalization after each convolution layer, and LeakyReLU [32] as the default activation function. To the best of our knowledge, we are the first applying GAFMTF in bot detection domain.

4.3 Decision Making

Given extracted features and learned embeddings/features from the different components, we have to unify them and make full use of them for the final prediction. The design of decision making is non-trivial in three reasons:

- Learned embeddings are so far matrices, while handcrafted features are 1D vectors. A good mechanism for flattening those matrices is needed. A simple direct flattening may create too much redundant information. How to balance the relative size (number of features) among different parts of features? Auto learned features are scalable but maybe way larger than handcrafted features in terms of the number of features. To avoid handcrafted features being overwhelmed, we have to design mechanisms to balance their contribution and significance well.
- How to handle the possible redundant information/noise inside each part of learned features? Especially, learned embeddings/features are large in size, so there may be possible correlations among them. A good design should have mechanisms to reduce the size of each part as well as to reduce correlations of each feature toward improving the decision making performance.
- Handcrafted features may have greatly varying value scale, which might not fit well with the other features. We had to handle such a problem to enable our framework to have a smoother learning curve.

To overcome these difficulties, we design decision making part as follows:

- **To rescale a value scope of the handcrafted features**, we normalize them in the training set and apply the normalizer in the validation set and testing set, to ensure not having information leak across experiment sets. The details of data split are described in Section 5.
- **For flattening 2D matrices into 1D vectors**, those representation matrices were then fed into two independent additive attention [5] + max-pooling + mean-pooling mechanisms. This was partly inspired by ULMFit [24], where the authors of this language model reported such design helped increasing performance. The two vectors of each part were then concatenated together as the final text embedding representation.
- **For balancing the size of different components**, after doing matrix flattening and concatenation, text embeddings and temporal behavior embeddings go through two different fully connected layers for resizing the feature dimensions, where the output size of those two layers are tunable as hyper-parameters.

Table 3. A general view of 7 baselines and our model (*JntL*).

Models	Using Info.			Feature		Algo.		Domain Specific		
Notation	Profile	Text	Timeseries	Handcraft	Auto	NN.	Trad. ML	Classify	Twitter Bot	Detect
Lee'11	✓	✓		✓			✓	✓	✓	✓
Kim'14		✓			✓		✓	✓		
Tweet2Vec'16		✓			✓	✓		✓	✓	
Chavoshi'18			✓		✓	✓		✓	✓	✓
Kudugunta'18		✓		✓	✓	✓		✓	✓	✓
RTBust'19			✓		✓	✓	✓	✓	✓	✓
Yang'19	✓	✓	✓	✓			✓	✓	✓	✓
JntL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

- Eventually, we concatenate all three parts of features together and feed them into multiple fully connected layers (MLP) for the final prediction. To reduce feature co-adapting and model overfitting, we applied dropout at all layers except the last fully connected layer as implicit Bayesian approximation [22].

5 Experiment

5.1 Experimental Setting

Dataset. Given 37,497 malicious bots and 37,941 legitimate accounts. We randomly split the dataset with 60% for training, 20% for validation, and 20% for testing. Through the dataset splitting procedure, we manually guarantee that all source datasets have at least one account shown up in each split, so to ensure that the experiment results are fair to all sources.

Data coverage and goal. As we mentioned in Sections 2 and 3, baselines and our approach only access limited user profile information, posting content information, and posting timestamps without other expensive information such as social network information because of the limited information commonly available across the 15 public datasets. Our goal is to maximize the malicious bot detection performance under the available information. Additional information like the social network could potentially further improve model performance.

Baselines and our model. We implemented the 7 state-of-the-art baselines based on various perspectives of user account information. The baselines consist of *Lee'11* [30], *Kim'14* [25], *Tweet2Vec'16* [20], *Chavoshi'18* [11], *Kudugunta'19* [28], *RTBust'19* [33], *Yang'19* [43]. Our proposed framework called *JntL*, which is a joint learning model shown in Figure 1. Table 3 shows a general view of the baselines and our model in terms of used user information, feature types and algorithm types. The detailed description of the baselines is presented in Appendix A.3. Our source code is available at https://github.com/GMouYes/MaliciousBotDetection.

Parameter tuning and evaluation metrics. We used the reported best hyper-parameters of the baselines. If the authors of them did not report the best hyper-parameters, we conducted grid search to obtain the optimal baselines.

To help the reproducibility of our model, we report our settings and hyper-parameters other than those already shown in Figure 1. Cross Entropy was chosen as the default loss function, and we used Adam as our optimizer. LeakyReLU

Table 4. Experimental results.

Class	Overall		Legitimate Users			Malicious Bots		
Measure	ACC.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
Lee'11	.874	.874	.875	.875	.875	.874	.874	.874
Kim'14	.829	.829	.838	.818	.828	.821	.841	.830
Tweet2Vec'16	.660	.660	.652	.696	.673	.670	.624	.646
Chavoshi'18	815	.815	.809	.828	.818	.822	.803	.812
Kudugunta'18	.837	.837	.854	.816	.834	.822	.859	.840
RTBust'19	.497	.342	.500	**.979**	.662	.322	.010	.019
Yang'19	.872	.872	.834	.922	.878	.912	.822	.865
JntL	**.901**	**.901**	**.886**	.922	**.903**	**.918**	**.880**	**.898**

with default parameters was chosen as the activation function for decision making layers. The Batch size is 128. The two-layer Bi-LSTM has hidden dimension of 256, to produce WordEmb based features, so the output of two directions is 512. The dropout between them is 0.2. The resizing fully connected layer for the text embedding layer is 256, while the resizing fully connected layer for the temporal behavior embedding layer is 1024. The decision making component (i.e., MLP) has two hidden layers, which have 512 nodes and 128 nodes. Its output layer produces a probability of being a malicious bot. The dropout rate of each layer in the decision making component is both 0.05. We evaluate all models based on Precision(Pre), Recall(Rec), F1, and Accuracy(Acc).

5.2 Experimental Results

Our model vs. baselines. We compared our model against 7 baselines. Table 4 presents the experimental results. The best result of each column is marked in bold. Our model outperformed the baselines achieving 0.901 accuracy, improving 3% compared with the best baselines (*Lee'11* and *Yang'19*). This result indicates that our proposed framework is better than the baselines in terms of identifying both types of accounts.

From the results, we conclude that jointly and thoroughly learned features provide better results than partially observed user's information. While analyzing user's postings provide rich information of malicious intent, text features like *Kim'14* take only the order of postings but does not pay attention to the exact posting time. Thus, incorporating temporal behavior would be complementary and helpful. However, given less active malicious users, posted messages and posting behavior may not be sufficient, so incorporating profile information would be helpful and necessary. Statistical information provides another general view of user behavior without caring about specific activities. Scalable auto learned features help boost the detection performance. *RTBust'19* does not perform well enough, as it actually requires the retweet behavior to reach a certain threshold. However this requirement is not generally met across all source datasets. In other words, the approach may require collecting each user's data with a longer time span. Baselines such as *Lee'11* and *Yang'19* using handcrafted features were relatively performed well, confirming usefulness of handcrafted features despite two drawbacks (i.e., not entirely scalable and a clear target for adversarial attacks) mentioned in Section 2. The other deep learning based baselines reached comparable results but not sufficient to beat the

Table 5. Ablation study result. "-" represents removing a corresponding feature type.

Class	Overall		Legitimate Users			Malicious Bots		
Model	Acc.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
JntL	.901	.901	.886	.922	.903	.918	.880	.898
-SentEmb	.889	.889	.901	.876	.888	.878	.902	.890
-WordEmb	.892	.892	.878	.912	.895	.907	.872	.890
-IPTEmb	.899	.899	.899	.899	.899	.898	.898	.898
-GAFMTF	.886	.886	.905	.864	.884	.868	.908	.888
-TraditionalFeatures	.887	.887	.882	.895	.888	.892	.880	.886

handcrafted features based baselines. We conjecture that this is mainly due to the fact that their work only focused on a part of user behaviors like retweets. On the contrary, our model's higher performance is because we considered a wider scope of user information, incorporated both handcrafted features and auto learned features, and made balance between them by using our decision making component.

5.3 Ablation Study

We conducted an ablation study to understand the contribution of each type of the features. Table 5 presents the experimental results when we remove one of the five feature types from our proposed model (*JntL*). For example, *-SentEmb* means excluding sentence level embeddings from *JntL*. We notice that (1) all types of features/embeddings positively contributed to the final prediction performance. (2) Even if we exclude traditional features (handcrafted features), ablation results still outperform all the baselines. The results reflect the success and effectiveness of our joint learning approach, where multi-perspective information based auto-learning provides a unique scalable advantage. GAFMTF features contribute the most, while WordEmb features contribute the least. This result is mainly because Sentence level embedding already captures part of the content information, while GAFMTF and IPTEmb encode temporal behaviors a lot differently. Automatically learned features easily scale better and thus provide helpful support to handcrafted features. Future work can be to explore other ways to learn scalable features through deep learning frameworks automatically.

6 Conclusion

In this paper, we aimed to detect various types of malicious bots altogether and distinguish them against legitimate users. In particular, we combined 15 publicly available Twitter datasets. We grouped accounts into two classes: (1) malicious bots; and (2) legitimate accounts. Then, we proposed a novel joint learning framework based on handcrafted features and auto learned features toward detecting malicious bots. Experimental results showed that our framework outperformed all baselines, achieving 0.901 accuracy, improving 3% against the best baseline. The ablation study provided supporting information indicating all parts of our framework non-trivially contributed to performance improvement.

Acknowledgements

This work was supported in part by NSF grant CNS-1755536, AWS Cloud Credits for Research, and Google Cloud.

References

1. Adewole, K.S., Anuar, N.B., Kamsin, A., Varathan, K.D., Razak, S.A.: Malicious accounts: dark of the social networks. Journal of Network and Computer Applications **79**, 41–67 (2017)
2. Albadi, N., Kurdi, M., Mishra, S.: Hateful people or hateful bots? detection and characterization of bots spreading religious hatred in arabic social media. CSCW (2019)
3. Alfifi, M., Caverlee, J.: Badly evolved? exploring long-surviving suspicious users on twitter. In: International Conference on Social Informatics (2017)
4. Alfifi, M., Kaghazgaran, P., Caverlee, J., Morstatter, F.: Measuring the impact of isis social media strategy. In: MIS2 (2018)
5. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
6. Beskow, D.M., Carley, K.M.: Bot conversations are different: leveraging network metrics for bot detection in twitter. In: ASONAM (2018)
7. Bhat, S.Y., Abulaish, M.: Community-based features for identifying spammers in online social networks. In: ASONAM (2013)
8. Campello, R.J., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining. pp. 160–172. Springer (2013)
9. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., Kurzweil, R.: Universal sentence encoder for English. In: EMNLP (2018)
10. Chavoshi, N., Hamooni, H., Mueen, A.: Temporal patterns in bot activities. In: WWW (2017)
11. Chavoshi, N., Mueen, A.: Model bots, not humans on social media. In: ASONAM (2018)
12. Conroy, N.J., Rubin, V.L., Chen, Y.: Automatic deception detection: Methods for finding fake news. In: Proceedings of the 78th ASIS&T Annual Meeting (2015)
13. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: Fame for sale: Efficient detection of fake twitter followers. Decision Support Systems **80**, 56–71 (2015)
14. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In: WWW (2017)
15. Cresci, S., Lillo, F., Regoli, D., Tardelli, S., Tesconi, M.: $ fake: Evidence of spam and bot activity in stock microblogs on twitter. In: ICWSM (2018)
16. Cresci, S., Lillo, F., Regoli, D., Tardelli, S., Tesconi, M.: Cashtag piggybacking: uncovering spam and bot activity in stock microblogs on twitter. ACM Transactions on the Web (TWEB) **13**(2), 1–27 (2019)
17. Cresci, S., Petrocchi, M., Spognardi, A., Tognazzi, S.: Better safe than sorry: An adversarial approach to improve social bot detection. In: WebSci (2019)
18. Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F.: Botornot: A system to evaluate social bots. In: WWW (2016)
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
20. Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., Cohen, W.: Tweet2vec: Character-based distributed representations for social media. In: ACL (2016)

21. Ferrara, E.: Measuring social spam and the effect of bots on information diffusion in social media. In: Complex Spreading Phenomena in Social Systems, pp. 229–255. Springer (2018)
22. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: ICML (2016)
23. Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., Crowcroft, J.: Of bots and humans (on twitter). In: ASONAM (2017)
24. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: ACL (Jul 2018)
25. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP (2014)
26. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
27. Ko, R.: Social media is full of bots spreading covid-19 anxiety. don't fall for it. https://www.sciencealert.com/bots-are-causing-anxiety-by-spreading-coronavirus-misinformation (2020)
28. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Information Sciences **467**, 312–322 (2018)
29. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. Information Sciences **467**, 312–322 (2018)
30. Lee, K., Eoff, B.D., Caverlee, J.: Seven months with the devils: A long-term study of content polluters on twitter. In: ICWSM (2011)
31. Ma, J., Gao, W., Wei, Z., Lu, Y., Wong, K.F.: Detect rumors using time series of social context information on microblogging websites. In: CIKM (2015)
32. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML (2013)
33. Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., Tesconi, M.: Rtbust: exploiting temporal patterns for botnet detection on twitter. In: WEBSCI (2019)
34. Miller, Z., Dickinson, B., Deitrick, W., Hu, W., Wang, A.H.: Twitter spammer detection using data stream clustering. Information Sciences **260**, 64–73 (2014)
35. Morstatter, F., Wu, L., Nazer, T.H., Carley, K.M., Liu, H.: A new approach to bot detection: striking the balance between precision and recall. In: ASONAM (2016)
36. Ruths, D.: The misinformation machine. Science **363**(6425), 348–348 (2019)
37. Shao, C., Ciampaglia, G.L., Varol, O., Yang, K.C., Flammini, A., Menczer, F.: The spread of low-credibility content by social bots. Nature communications **9**(1), 4787 (2018)
38. Subrahmanian, V., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., Menczer, F.: The darpa twitter bot challenge. Computer **49**(6), 38–46 (2016)
39. Varol, O., Ferrara, E., Davis, C.A., Menczer, F., Flammini, A.: Online human-bot interactions: Detection, estimation, and characterization. In: ICWSM (2017)
40. Wang, Z., Oates, T.: Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In: AAAI-W (2015)
41. Yang, C., Harkreader, R., Gu, G.: Empirical evaluation and new design for fighting evolving twitter spammers. IEEE Transactions on Information Forensics and Security **8**(8), 1280–1293 (2013)
42. Yang, K.C., Varol, O., Davis, C.A., Ferrara, E., Flammini, A., Menczer, F.: Arming the public with artificial intelligence to counter social bots. Human Behavior and Emerging Technologies **1**(1), 48–61 (2019)
43. Yang, K.C., Varol, O., Hui, P.M., Menczer, F.: Scalable and generalizable social bot detection through data selection. In: AAAI (2020)
44. Young, L.Y.: The effect of moderator bots on abusive language use. In: ICPRAI (2018)

A Appendix

A.1 Account Status

As we keep the original information of *Lee'11*, *Cresci'15* and *Cresci'17*, we checked the current status of those malicious bots as shown in Table 6. Overall 68.3% malicious bots are still alive on Twitter, some of which lived more than ten years. This fact indicates that there is a great room to improve the current Twitter's bot detection system.

A.2 Source Dataset Details

We list the original user types that each dataset contains as follows:

Lee'11 [30]: content polluters, and legitimate users
Cresci'15 [13]: various kinds of fake accounts
Cresci'17 [14]: traditional & social spambots, and legitimate users
Cresci'18 [15, 16]: stock related bots, and legitimate users
RTBust'19 [33]: retweet bots, and legitimate users
Gilani'17 [23]: bots, and legitimate users
Varol'17 [39]: bots, and legitimate users
Midterm'18 [43]: political bots, and legitimate users
Botwiki'19 [43]: social and bots
Political'19 [42]: political bots
Pronbots'19 [42]: bots advertising scam sites
Vendor'19 [42]: fake followers
Verified'19 [42]: verified legitimate users
Celebrity'19 [42]: celebrity accounts (legitimate)
Botometer'19 [42]: bots and legitimate users

We grouped legitimate users, verified accounts and celebrity accounts as legitimate, while other types of accounts as malicious bots.

Table 6. Recent status of malicious accounts.

Source	Deleted	Suspended	Alive	Sum
Lee'11	1,417 (8.2%)	3,868 (22.4%)	11,956 (69.3%)	17,241
Cresci'15	282 (6.0%)	2,336 (49.9%)	2,067 (44.1%)	4,685
Cresci'17	344 (6.3%)	443 (8.1%)	4,678 (85.6%)	5465
Overall	2,043 (7.5%)	6,647 (24.3%)	18,701 (68.3%)	27,391

A.3 Detailed Baseline Descriptions

Lee'11 [30]. Authors proposed handcrafted features extracted from user profiles, posting contents and the change of following/follower list over time. We built their best Random Forest model without the network features.
Kim'14 [25]. This is a convolutional text classification architecture that achieved comparable performance against state-of-the-art models. Its hyper-parameters

are stable across different domains. We applied his work in using the tweets posted by each user for classifying the accounts.

Tweet2Vec'16 [20]. Tweet2Vec was proposed as a general-purpose tweet embedding framework, trained with neural networks for the hashtag prediction subtask. This work generates domain-specific feature representations of tweets. We constructed a bot detection model, following the proposed architecture, where the embedding layer is followed with fully connected layers.

Chavoshi'18 [11]. Authors proposed a method for mapping the posting timestamp pairs into 2D images to make better use of the temporal posting behavior information of each account. Convolutional neural networks can be applied for the downstream bot detection task.

Kudugunta'19 [28]. This is a framework using LSTM for learning content features and then combine them with several handcrafted features.

RTBust'19 [33]. RTBust is a framework using temporal retweet/tweet patterns for bot detection. Such a framework captures the information in tweet/retweet sequences and extracted features using the variational autoencoder (VAE) [26]. Then the feature embedding generated by the encoders is fed into HDBSCAN [8], an unsupervised clustering method. Outliers are treated as malicious bots.

Yang'19 [43]. Random Forest built on various authors' proposed features.