Detecting Various Types of Malicious Users at One Sitting in Online Social Networks

by

Guanyi Mou

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

June 2019

APPROVED:

Professor Kyumin Lee, Major Thesis Advisor

Professor Mohamed Y. Eltabakh, Thesis Reader

Abstract

Online social networks (OSNs) have long been suffering from various types of malicious users such as spammers and bots. Over several years, researchers proposed multiple approaches to identify different types of them toward lowering their impact into the OSNs. However, their strategies mostly focused on some specific types of malicious users (e.g., spammers, bots), or they less paid attention to newly emerging malicious users. To overcome the limitation of the prior work, in this study, we proposed a novel method to detect various types of malicious users at one sitting. In particular, we (i) combine publicly available Twitter user datasets and categorize these accounts into two groups (e.g., legitimate account, and malicious account); and (ii) propose a robust deep learning framework which jointly learns various features and detects malicious accounts. Our experimental results show that our proposed models outperform stat-of-the-art baselines, effectively detecting various types of malicious users at one sitting. Under the training data reduction scenario, our models consistently achieve high accuracy. Our source code and dataset are available at an anonymized URL.

Contents

List of Figures

List of Tables

1 Introduction

Online social networks (OSNs) such as Twitter, Facebook, and Weibo have long been suffering from various types of malicious users (e.g., spammers, crowdturfers, fake followers, and social spambots). These malicious users have misused the power of OSNs, acted as content polluters, continuously caused significant disturbance to the overall online social environment, and shaped unhealthy trends, bias and misbelief in societies. Their accounts[1] have made severe impact and damage to the OSNs by causing inconveniences, intensifying contradictions and aggravating prejudices.

In the recent years, OSN service providers established policies for warning, blocking, and even suspending malicious users[2]. Researchers have proposed approaches to detect specific types of malicious users [7, 8, 12, 19, 22, 28, 35, 43]. Even though their proposed approaches identified some of these malicious accounts well, we are still facing new challenges by new malicious accounts such as hashtag promoters, social spambots especially political bots and even extremists such as ISIS recruiters. According to our study (which will be described in the following sections), some of these malicious accounts are still alive for years without any further punishment or proper treatment. These new types of malicious users should be banned out. There are also accounts intentionally disseminating hate speeches and rumors/fake news [25, 42, 51]. In addition, researchers rarely studied various types of these malicious users altogether.

To fill this gap, in this thesis, we are interested in studying various types of malicious users at one sitting. However, there are a few challenges. First, how to collect information of various types of malicious accounts? If we collect some of malicious accounts in different timing, it may be hard to detect them (e.g., collect

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

[2] An example: `https://help.twitter.com/en/rules-and-policies/twitter-rules`

information of some users 2 years ago and collect information of the other users today). Can we propose a unified framework which can effectively detect all types of malicious accounts at one sitting?

By keeping these challenges in mind, in this thesis, we combine publicly available multiple Twitter datasets, which contain accounts of content polluters, fake followers, traditional spambots, social spambots and legitimate users. We categorize these accounts to malicious accounts and legitimate accounts, and then compare how they are similar or different. Finally, we propose a unified framework based on various features to distinguish between malicious accounts and legitimate accounts.

In this thesis, we make the following contributions:

- We propose a novel joint learning framework that is capable of identifying various types of malicious accounts at one sitting. It combines profile and activity features, LIWC-based personality features, and two CNN model based features in order to jointly learn various types of user perspectives together.
- Our proposed models outperform 8 state-of-the-art baselines.
- Our study show that our proposed framework is robust even with less sufficient data and still outperform the state-of-the-art baselines.

2 Related Work

In this section, we summarize some of the prior works related to malicious accounts in OSNs, attention mechanisms and convolutional neural networks(CNNs).

2.1 Malicious Users

Researchers focused on analyzing and detecting content polluters and spammers in OSNs [4, 9, 24, 40]. They investigated and analyzed these users, and then proposed

various methods to detect them. Lee et al. [35] did a long-term study of content polluters, and further proposed a well-built classifier for detecting content polluters.

Other researchers studied how to detect bots, especially malicious spambots, from different perspectives, including classifications based on temporal patterns of behaviors [11, 12], sentiment analysis [23], networks [8] and many other methods [24, 33, 41]. DARPA held a twitter bot challenge [44] for better understanding and detecting bots. Davis et al. [22] proposed a framework "BotOrNot" which was trained on a dataset of malicious users. Their model produced an output called "Botness-score" between 0 and 1 to determine whether an account is a malicious bot with a confidence rate. Adewole et al. [1] made a thorough review of 65 bot detection papers. By doing comparisons and contradictions, it concluded with many useful observations, including techniques and features.

Alfifi et al. [2, 3] studied the behavior of long-lived eventually suspended accounts in social media (especially ISIS related accounts) through a comprehensive investigation of Arabic Twitter. They were granted full access to Arabic Twitter accounts, even including data of those suspended accounts from Twitter. They then researched on those long-lived, short-lived, legitimate users and pro-ISIS users. Benigni et al. [7] also tried to fight against extremists such as ISIS by building frameworks to detect their supporting community. Recently there is also a rising trend in fake news detection and rumor detection. There have been traditional models [16, 39, 45] as well models made use of deep learning frameworks [28, 38, 43].

Cresci et al. [18, 19, 20, 21] proposed a DNA inspired model which produced a relatively good result without much detailed information from users. Lee et al. [36] built an unsupervised machine learning model and used a hierarchical design to detect malicious users. They gathered their ground truth by using user account status (i.e., suspended by Twitter or not). Therefore, they might overlook many

malicious users who were not yet(or even never) suspended by Twitter. Viswanath et al. [48] also tried to identify malicious users. They focused on building a model to detect "attacks" which are generated by workers in black markets. Although many papers focused on different types of users and relied on various OSN platforms, there is little work on identifying various types of malicious users altogether by extracting their common characteristics.

2.2 Attention Mechanisms

Recently there is a rising trend in applying attention mechanisms in many different research domains. Born to resolve problems in remembering long term memory in neural machine translation(NMT) of natural language processing (NLP) domain [5], it soon became popular in other domains such as computer vision(CV) [50]. Many researchers also explored deeply in many variations of attention mechanisms [10, 37, 47]. In this thesis, we apply soft self-attention [14] which uses dot product to improve performance of our model and make our model less complicated.

2.3 Convolutional Neural Networks

Traditionally, Convolutional Neural Networks(CNNs) were well applied in research domains such as image processing [32] and computer vision [34]. Many researchers have shown quite satisfying results in these areas. Recently, it has also been applied to Natural Language Processing(NLP) problems such as sentence modeling [29], classification [30] and prediction [15]. For example, Kim [30] proposed a CNN for sentence classification. They used word level embedding to transfer text content into a matrix which fed to the CNN framework for classification tasks. Kalchbrenner et al. [29] explored CNN's performance and tried different adjustments on many datasets. Wang et al. [49] built an event adversarial neural network for multi-modal

Table 1: Statistics of three datasets.

Dataset	User Type	Size	UserId	Profile	Tweet	Network	Label
Honeypot Dataset	Content Polluter(CP)	22,223	✓	✓	✓	✓	✓
	Legitimate User(LU)	19,276	✓	✓	✓	✓	✓
DNA Inspired Dataset	Fake Follower (FF)	3351	✓	✓	✓	✗	✓
	Normal User(NU)	3,474	✓	✓	✓	✗	✓
	Social Spambot1(SSB1)	991	✓	✓	✓	✗	✓
	Social Spambot2(SSB2)	3,457	✓	✓	✓	✗	✓
	Social Spambot3(SSB3)	464	✓	✓	✓	✗	✓
	Traditional Spambot1(TSB1)	1,000	✓	✓	✓	✗	✓
	Traditional Spambot2(TSB2)	100	✓	✓	✗	✗	✓
	Traditional Spambot3(TSB3)	403	✓	✓	✗	✗	✓
	Traditional Spambot4(TSB4)	1,128	✓	✓	✗	✗	✓
BotOrNot Dataset	Bots(BoNB)	826	✓	✗	✗	✗	✓
	Human(BoNH)	1,747	✓	✗	✗	✗	✓

Table 2: Statistics of the combined dataset.

Dataset	Total Size	Type	Cur. Status	Anal. & Exp.
MA	33,943	CP	✓	✓
		FF	✓	✓
		SSB1	✓	✓
		SSB2	✓	✓
		SSB3	✓	✓
		TSB1	✓	✓
		TSB2	✓	✗
		TSB3	✓	✗
		TSB4	✓	✗
		BoNB	✓	✗
LA	24,497	LU	✓	✓
		NU	✓	✓
		BoNH	✓	✗

fake news detection, which also used a Text-CNN structure. Chavoshi et al. [13] proposed another CNN model which made use of the inter-posting timestamps(IPT) rather than the text content themselves.

3 Dataset

We were interested in collecting various types of malicious accounts so that we could learn their common characteristics which are different from legitimate accounts. In-

stead of collecting data from Twitter by ourselves, we decided to download publicly available Twitter datasets in order to avoid two problems: (1) we may collect biased sample data which was collected at the same time via the same method; and (2) labeling each type of malicious accounts would be time-consuming and getting objective labels are still challenging.

In particular, we downloaded the following three Twitter datasets: HoneyPot Dataset [35], DNA Inspired Dataset [19], and BotOrNot Dataset [46]. We describe the general information about datasets in Table 1. We treat the content polluters, fake followers, social spam bots 1,2,3, traditional spam bots 1,2,3,4, BotOrNot dataset's bot group, as malicious accounts/users (MA), and those legitimate users, BotOrNot dataset's human group, together with normal users as legitimate accounts/users (LA).

After downloading these datasets, we faced another problem that these datasets were not consistent with each other in terms of available information of users in each dataset. We had to make our choice of either selecting part of features or part of these dataset users.

As shown in Table 2, we used all the datasets consisting of 33,943 malicious accounts and 24,497 legitimate accounts to check a current status of each account via Twitter API, which only requires the user's Twitter ID and his/her label. In the stage of analysis and experiment, we used datasets containing user profiles, tweets, and labels (i.e., we excluded TSB2, TSB3, TSB4, BoNB and BoNH).

Current Status of the Malicious Accounts. In July, 2018, we checked a current status of each malicious account via Twitter API to understand whether the account has been suspended, deleted, protected or alive. Table 3 presents proportion of deleted, protected, suspended and alive accounts in each type. Overall, 81.5% malicious accounts were deleted or suspended. However, the remaining 18.5% ac-

Table 3: Current status of malicious accounts.

Types	Deleted	Protected	Suspended	Alive
CP	20,452 (92.0%)	254 (1.1%)	71 (0.3%)	1,446 (6.5%)
FF	2,140 (63.9%)	184 (5.5%)	35 (1.0%)	991 (29.6%)
SSB1	834 (84.2%)	20 (2.0%)	6 (0.6%)	131 (13.2%)
SSB2	906 (26.2%)	393 (11.4%)	172 (5.0%)	1,986 (57.5%)
SSB3	375 (80.8%)	15 (3.2%)	3 (0.7%)	71 (15.3%)
TSB1	925 (92.5%)	10 (1.0%)	5 (0.5%)	60 (6.0%)
TSB2	27 (27.0%)	12 (12.0%)	4 (4.0%)	57 (57.0%)
TSB3	192 (47.6%)	36 (8.9%)	14 (3.5%)	161 (40.0%)
TSB4	1,019 (90.3%)	16 (1.4%)	6 (0.5%)	87 (7.7%)
BoNB	448 (54.2%)	49 (5.9%)	20 (2.4%)	309 (37.4%)
Overall	27,318 (80.5%)	989 (2.9%)	336 (1.0%)	5,299 (15.6%)

counts are either alive for a few years or changed their profiles in a protected mode. It indicates we need a better detection method to identified those alive and protected malicious accounts.

4 Our Framework

In this section, we describe our proposed framework, which aims to detect various types of malicious accounts at one sitting. Our framework in Figure 1 consists of three parts: data integration, data grouping, and model training.

The data integration module is in charge of combining datasets from different sources, extracting their shared features, and transforming various expressions into unified ones. During these procedures, some accounts were filtered out as they failed to satisfy some essential criteria such as missing some important data/features. In the figure, Filter1 filters out those broken data and inconsistent data, and Filter2 filters out overlapped data and features that are not important to our model.

In the data grouping part, we split those remaining users into two groups: legitimate accounts, and malicious accounts.

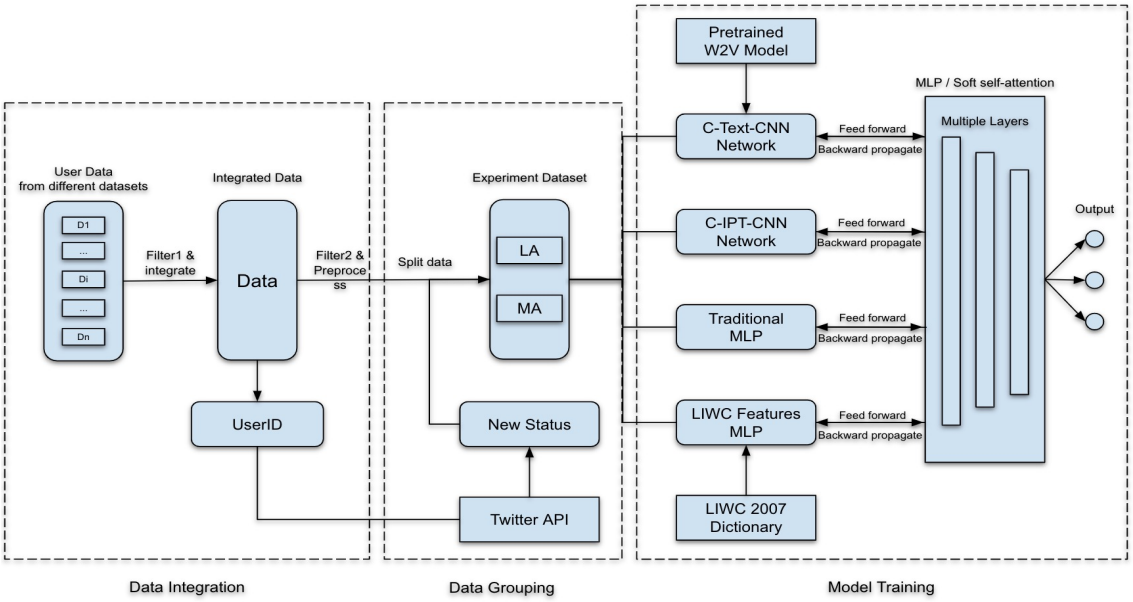

Figure 1: Overall framework.

After the datasets are thoroughly cleaned and grouped, we feed each type of different features to their corresponding neural network structures. Then, we combine their intermediate results to form a new fully connected layer. We try to take the advantages of these different features together with their structures by jointly learning them through feed forwards and back propagations altogether rather than learn them individually and then combine their results.

In the training part, we combine four types of features: (i) traditional features (i.e., profile features and activity features); (ii) LIWC features generated from each user's tweet content to understand the user's personality; (iii) Temporal behavior related features generated by our customized inter-posting-time CNN (C-IPT-CNN); and (iv) Text/content related features generated by our customized Text-CNN (C-Text-CNN). We will describe the four types of features in detail in the following section. After the layer concatenation, these features will be mapped to target classes through either classical multi-layer perceptrons (MLP, shown in the overall

framework as an example) or soft self-attention layer.

A CNN consists of an ordered sequence of layers. Types of these layers are usually Convolutional layers, Pooling layers, and Fully Connected layers. In most of cases, the results generated by Convolution layers and Pooling layers are treated as independent dimensions/perspectives of an original graph. Then, Fully Connected layers show up as the last several layers, mapping those independent features to target/expected results. The number of layers, types of layers, output size of layers are up to the designer of the CNN framework. A matrix as the input layer is fed to the CNN framework and the final output usually represents what researchers need from that matrix. For example, in classification problems, the output layer could constitute CNN's final prediction of input matrix/instance. Thus the size of the output is exactly the size of the classes. In the following subsections, we describe detailed information regarding our proposed two CNN networks to generate independent dimensions/features.

4.1 Our C-IPT-CNN Network

Inspired from Chavoshi et al. [13], who designed the original IPT-CNN, we propose customized IPT-CNN (C-IPT-CNN) as shown in Figure 2.

Input matrix of C-IPT-CNN. Given all timestamps of tweets posted by a user, we sort them in the ascending order and represent them as follows:

$$t_i, 1 \leq i \leq n \tag{1}$$

where n is the total number of tweets posted by the user.

Then, we calculate the time difference (i.e., inter-posting time) between a pair of timestamps with respect of time lag, and apply the log function to decrease its

range as follows:

$$p_i = \begin{cases} \log_{10}(t_{i+lag} - t_i) + 1, & t_{i+lag} > t_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where p_i is the time difference in terms of the second as a unit, and lag is the number of hops to choose following timestamp. For example, if $lag = 1$, we will measure p_i based on t_i and t_{i+1}. Similarly, if $lag = 2$, we will measure p_i based on t_i and t_{i+2} pair. In this study, we choose lag as 1, 2, and 3.

Now, we have three sequences of tuples each of which was generated by each lag value that we chose. Each tuple in a sequence contains two consecutive time difference: (p_i, p_{i+1}). Given each tuple in a sequence, we map it to a 2D plane, considering p_i as a value of x-axis and p_{i+1} as a value of y-axis. In this thesis, we set up the boundaries of x-axis and y-axis as $min = 0$ and $max = 10$. All possible extreme values that were initially out of the boundary is forced to map to their closest limit. Then we manually divide the chosen area into 32*32 grids. Since there are three sequences (again, we chose three lag values), we have 3*32*32 grids. Thus, we have a $3 \times 32 \times 32$ matrix, where each element is the proportion of counts in the plane. Such a matrix is the input matrix of C-IPT-CNN. Notice that our expression is different from the original one as we handle time difference differently, and we also incorporate a smoothing constant 1 in Eq. 2.

Design of C-IPT-CNN. The input matrix will go through four convolution layers and three pooling layers. We flatten the result of the last pooling layer, and exclude all fully connected layers of the original IPT-CNN design. Then, the result will be concatenated with other network results to form a more extended fully connected layer to do the joint learning as we described in the overall framework. The weights

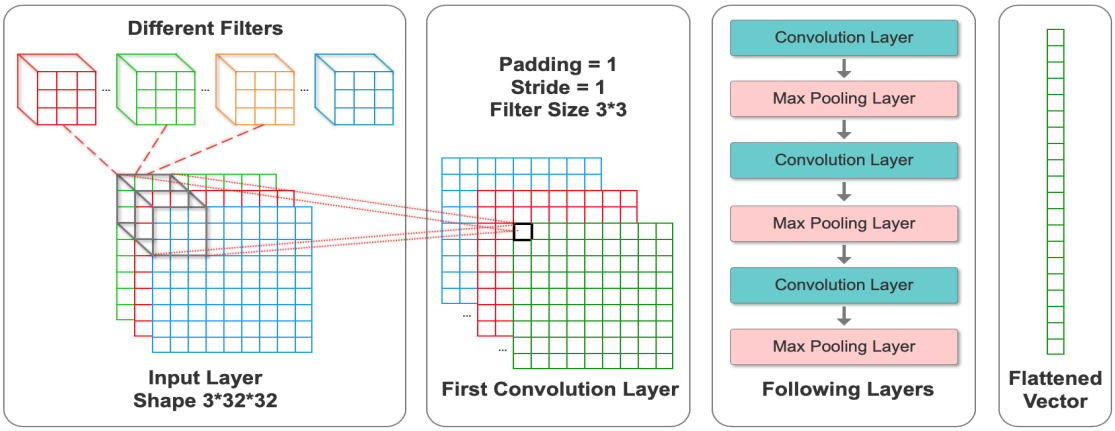

Figure 2: Our C-IPT-CNN network structure.

of each layer will be updated by the back propagation method.

4.2 Our C-Text-CNN

Input matrix of C-Text-CNN. Inspired from the original Text-CNN [30] and Kalchbrenner et al. [29], we propose customized Text-CNN (C-Text-CNN) presented in Figure 3. First, we process our tweets by tokenizing them, filtering out useless characters and meaningless words, and lowercasing remaining words. Second, we pad two unique words "TWEETSTART" and "TWEETEND" to each non-trivial tweet. Then, we concatenate all the tweets together for each user. Now each user is represented as a long paragraph. The next step is to translate/convert the paragraph into meaningful embedding vectors. In particular, we used a pre-trained word to vector (W2V) model on Twitter[3] to translate the paragraphs into matrices. Notice that to let the embedding make sense for unrecognized words, "TWEETSTART" and "TWEETEND", three extra dimensions were expanded to the original embedding vector. It made the length of each word's embedding vector to be changed from 400 to 403. We did extra padding for those matrices to make sure all users'

[3] `https://fredericgodin.com/software/`

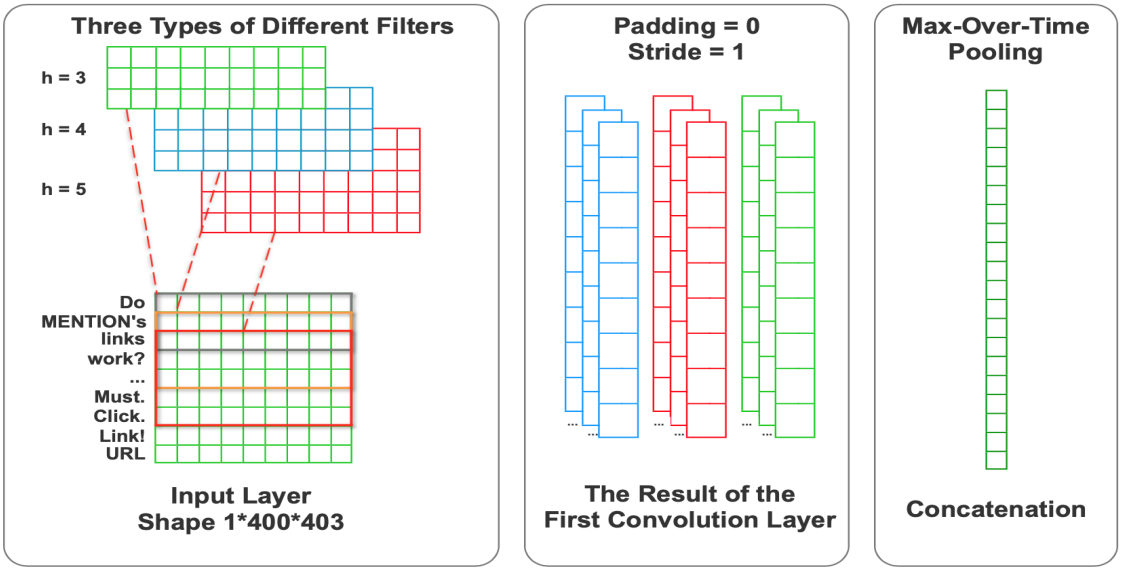

Figure 3: Our C-Text-CNN network structure.

embedding matrices have the same height (i.e., the same number of words), which will fit into our C-Text-CNN model well.

Design of C-Text-CNN. Our C-Text-CNN contains one Convolutional layer, and one Pooling layer. Similar to C-IPT-CNN, the result of the pooling layer is flattened, and then is concatenated with other features for the joint learning framework. Instead of using other pre-trained models such as Google's "negative-300", we used a domain specified W2V model which was trained on a Twitter dataset to represent each word correctly in the Twitter domain.

5 Features

In this section, first we describe a list of features that our framework extracts and uses. Then, we analyze whether the traditional features can clearly distinguish between malicious accounts and legitimate accounts. Table 4 presents the list of our features.

Table 4: Features and their notations.

Feature Type	Notation	Description
Traditional Features	t/d	|Average tweets posted per day|
	dd	|days since account creation|
	ut/t	Unique tweet ratio
	h/t	|hashtags posted per tweet|
	uh/t	|unique hashtags posted per tweet|
	m/t	|mentions posted per tweet|
	um/t	|unique mentions posted per tweet|
	l/t	|links (URLs) posted per tweet|
	ul/t	|unique links (URLs) posted per tweet|
	$len(sn)$	Length of screen name
	$len(des)$	Length of description
	$fing/d$	|new followings per day|
	fer/d	|new followers per day|
	ff	Following follower ratio
	cr	tweet compression ratio
LIWC features	-	64 LIWC features
CNN features	-	C-IPT-CNN and C-Text-CNN features

5.1 Traditional Features

We first extracted 15 traditional features like profile features and activity features from each user's profile and tweets. Many baseline models made use of these features and claimed that they achieved rather satisfying results. We concatenate these features with other features generated from different perspectives and apply joint learning methods in our framework. For unique tweets, we first translate all links to the same word "URL" and then count the number of unique tweets based on these transformed data. For compression ratio of user tweets, we used Python's zip package with its default zip setting.

5.2 LIWC Related Personality Features

LIWC[4] is a well-known 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.

5.3 CNN Related Features

We build two CNN models (i.e., C-IPT-CNN and C-Text-CNN) and extract CNN related features to capture different characteristics of Twitter users. The number of features to be extracted depends on a result of hyper-parameter tuning for each CNN.

5.4 Malicious Accounts vs. Legitimate Accounts in Traditional Features

We turn to compare malicious accounts and legitimate in terms of traditional features. Do they have different characteristics? Can the traditional features distinguish between them? If all the traditional features are already good enough in separating different groups of users, then it may not be necessary to add additional feature into our framework. To answer these questions, we visualize traditional feature test results in the form of histograms in Figure 4. For better comprehensive visualization, the actual scope of the y-axis is compressed in the log of the actual

[4] `http://liwc.wpengine.com/`

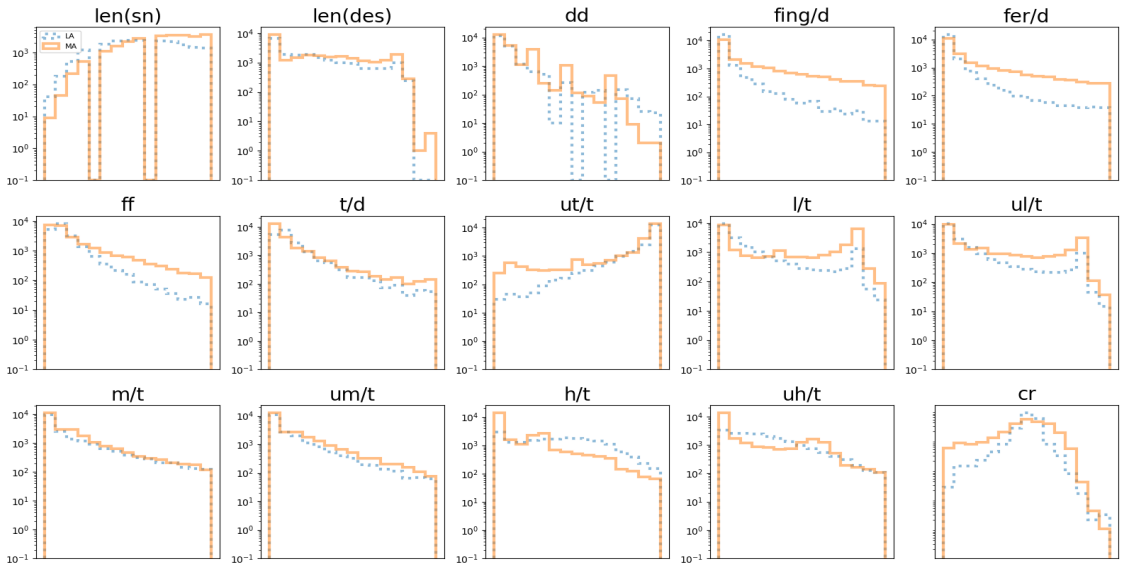

Figure 4: Histograms displaying malicious accounts and legitimate accounts in terms of traditional features. The full name of each histogram name is described in Table 4.

number of users. There is also a long tail on the original histograms where only a small proportion of users are distributed in a rather long range of domain. We intentionally keep 99% of the users' data in the graph by removing 1% tail users in the histograms to see clear macro-scale views. As the exact x-axis value does not matter much, we hide them in the histograms.

We observe that some traditional features can partially tell some difference between groups of accounts; however, there is still much overlap among them. Especially, length of screen name (*len(des)*), the number of average tweets posted per day (t/d), the number of mentions posted per tweet (m/t), and the number of unique mentions posted per tweet (um/t) are less effective to distinguish between malicious accounts and legitimate accounts. This result indicates that our approach to combine various features makes sense.

Table 5: Experimental dataset.

Type	Size
Malicious Accounts (MA)	25,543
Legitimate Accounts (LA)	19,652

6 Experiment

6.1 Experimental Setting

Given 33,943 malicious accounts and 24,497 legitimate accounts, we conducted data filtering described in Section 4. In particular, we filtered out accounts each of which posted less than 5 tweets, and the number of aggregated words in the tweets is less than 10. Finally, 25,543 malicious accounts and 19,652 legitimate accounts remained as shown in Table 5. We randomly split the dataset with 80% training, 10% validation, and 10% testing.

We choose the following state-of-the-art baselines based on various features: (i) logistic regression(LG), decision tree(DT), support vector machine(SVM) [17] based on traditional and LIWC features; (ii) Bagging methods, such as random forest(RF) [6, 27] and boosting techniques such as AdaBoost(AdaB) [26] based on traditional and LIWC features; (iii) Original Text-CNN (CNN1) [29, 30] to model tweet information; (iv) Original IPT-CNN (CNN2) [13] to model a user's temporal posting behavior patterns; and (v) Multilayer perceptron (MLP) based on traditional and LIWC features.

We implement our proposed framework in two variations called *Co1* and *Co2*. *Co1* is a joint learning model attached with one single soft self-attention. *Co2* is another joint learning model attached with a MLP of 3 hidden layers (as shown in Figure 1), and the dropout rate is applied to prevent over-fitting. Cross entropy is applied as the default loss function for all the models.

Table 6: Experimental result for the binary classification.

	LA			MA			Overall
Model	Pre.	Rec.	F1	Pre.	Rec.	F1	Acc.
LR	0.837	0.851	0.844	0.884	0.873	0.878	0.863
DT	0.884	0.874	0.879	0.904	0.912	0.908	0.896
RF	0.928	0.911	0.920	0.933	0.946	0.939	0.931
AdaB	0.902	0.915	0.909	0.934	0.924	0.929	0.920
SVM	0.873	0.887	0.880	0.912	0.901	0.907	0.895
MLP	0.863	0.891	0.877	0.914	0.891	0.902	0.891
CNN2	0.826	0.800	0.813	0.850	0.871	0.860	0.840
CNN1	0.912	**0.923**	0.918	**0.940**	0.932	0.936	0.928
Co1	**0.939**	0.917	**0.928**	0.937	**0.954**	**0.946**	**0.938**
Co2	**0.935**	**0.925**	**0.930**	**0.943**	**0.951**	**0.947**	**0.940**

We conducted a thorough grid search for all possible hyper-parameters/options of all models, and used the validation set to verify and pick those best combinations. We measure Precision (Pre), Recall (Rec), F1 and Accuracy (Acc) as evaluation metrics.

Q1: Baselines vs. Our Models?

We compared our models against 8 baselines. Table 6 present the experimental results. The best result of each column is marked in bold red and the second best is marked in bold black. Both of our models outperformed the baselines, and achieved the highest F1 in both legitimate accounts and malicious accounts. It indicates our framework is stable in both types. Overall, our *Co2* performed the best, achieving 0.940 accuracy. This promising result confirms that it is possible to detect various types of malicious accounts at one sitting with a high accuracy. Our framework also performed better than Twitter security system which only identified 81.5% malicious accounts correctly as presented in Section 3.

We also conducted an additional experiment called a ternary classification task to show that our models can handle multi-class classification tasks. In particular, in the ternary classification task, we treat all social spambot accounts as newly

Table 7: Experiment result for the ternary classification.

3class	LA			TM			NM			Overall
Measure	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	ACC.
LR	0.846	0.840	0.843	0.853	0.829	0.841	0.863	0.986	0.921	0.851
DT	0.899	0.889	0.894	0.897	0.907	0.902	0.982	0.984	0.983	0.908
RF	0.920	0.914	0.917	0.917	0.927	0.922	**1**	0.982	0.991	0.927
AdaB	0.910	0.911	0.911	0.916	0.916	0.916	0.990	0.990	0.990	0.922
SVM	0.882	0.873	0.878	0.881	0.890	0.885	0.994	0.990	0.992	0.894
MLP	0.875	0.880	0.878	0.887	0.882	0.884	0.990	**0.992**	0.991	0.893
CNN2	0.835	0.806	0.820	0.821	0.849	0.835	0.996	0.988	0.992	0.846
CNN1	0.913	**0.925**	0.919	**0.926**	0.916	0.921	0.996	0.988	0.992	0.928
Co1	**0.936**	0.915	**0.926**	0.921	**0.942**	**0.931**	0.994	**0.992**	**0.993**	**0.936**
Co2	**0.929**	**0.939**	**0.934**	**0.942**	**0.932**	**0.937**	**0.998**	**0.992**	**0.995**	**0.942**

Table 8: Experimental results for Joint learning vs. separate learning.

	LA			MA			Overall
Model	Pre.	Rec.	F1	Pre.	Rec.	F1	Acc.
Se+LR	0.921	**0.921**	0.921	**0.939**	0.939	0.939	0.931
Co1	**0.939**	0.917	**0.928**	0.937	**0.954**	**0.946**	**0.938**
Co2	**0.935**	**0.925**	**0.930**	**0.943**	**0.951**	**0.947**	**0.940**

emerging and evolving malicious group (NM), the rest of the malicious accounts as traditional malicious group (TM), and keep the same legitimate accounts as LA. Table 7 shows prediction results. Our models still achieved the highest F1 in all groups, and our *Co2* performed even better, achieving 0.942 accuracy.

6.2 Experimental Results

In this section, we describe experimental results in a Q&A style as follows:

Q2: How does our joint learning framework perform differently from separate learning?

To answer this question, we did another experiment by training on each feature model separately rather than joint learning them together. Then, we applied straight forward "voting" to predict the final result of the separate learning method (Se+LR). In particular, The C-Text-CNN on content/Tweets, the C-IPT-CNN on temporal

Table 9: Detailed classification results (actual vs. predicted).

Model	Actual vs. Predicted	LA	MA
Co1	LA	1,801 (91.7%)	163(8.3%)
	MA	117(4.6%)	2,439 (95.4%)
Co2	LA	1847 (94.0%)	117(6.0%)
	MA	156 (6.1%)	2400 (93.9%)

behaviors, the MLP on traditional and LIWC features are independently well-tuned. Then, their prediction vector(the last result layer) was concatenated into one. Then, we ensembled the predictions together through classic machine learning methods. We used logistic regression to generate the final prediction (actually RF, SVM, MLP, DT, and AdaB were all tested, but we only report the best model's result here). The results are shown in Table 8. We observe that joint learning methods performed better than separate learning model. It may mean that ensemble methods in the separate learning naturally only focused on groups of features separately and easily overlooked the possible hidden correlations/dependencies between different types of features, whereas the joint learning took care of a variety of different groups of features and the correlations of features altogether.

Q3: What is the distribution of users that our models misclassified?

We present the detailed prediction results in Table 9. Each row is the original label, and each column is the predicted label. We observe that each of *Co1* and *Co2* have their own advantage over the other. *Co1* did better in terms of correctly identifying malicious users, whereas *Co2* did better in terms of correctly identifying legitimate users. There might be some chance of further improving the classification performance by combining the two models.

Q4: What is the original groups that those misclassified accounts belong to?

Table 10 shows the original groups of misclassified accounts. In this analysis, we

Table 10: The original groups of misclassified accounts.

Group	CP	TSB	SSB	FF	LA
Co1 Mistake	42(47.19%)	0(0%)	3(3.37%)	5(5.62%)	39(43.82%)
Co2 Mistake	55(58.51%)	2(2.13%)	3(3.19%)	6(6.38%)	28(29.79%)

focused on 5 account groups: social spambots (SSB), traditional spambots (TSB), content polluters (CP), fake followers (FF), and all legitimate accounts (LA). Each row represents all accounts that were mislabeled by a particular model, and each column represents the original group those accounts belong to. CP and LA contributed to the dominating part of mistakes as they were harder to detect compared with two types of malicious bots. Intuitively speaking, this might be due to the reason that bots behave in a fixed pattern and other malicious users are more flexible.

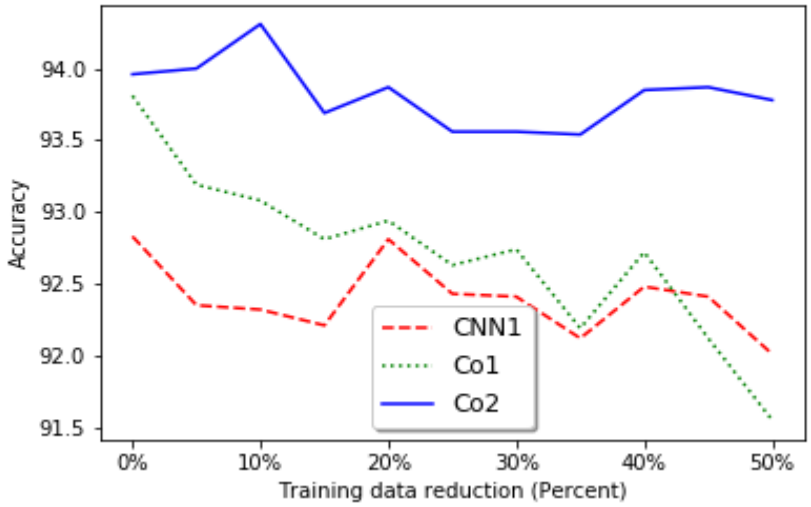

Figure 5: Performance of our models against CNN1 when training data reduction happened.

Q5:How robust is our framework?

We specially designed another experiment to show the robustness of our models. We randomly dropped out some accounts of the training data and checked performance of the models. For comparison, we chose CNN1 as a baseline model

as it achieved a good performance in the previous experiment. We kept all hyper-parameters learned from the previous experiment (using full training data), and gradually decreased a size of the training data with a step size of 5%. Figure 5 shows the experimental results related to training data reduction. We observe that regardless of the training data reduction, our *Co2* consistently achieved the highest accuracy and was robust. It significantly outperformed CNN1 even when the number of training accounts reduced to 50%. The *Co1* seems to rely on more training data, but still outperformed CNN1 until up to 40% training data reduction.

Q6: Is there any other room to further improve?

In this study, our combined dataset does not contain social network information. If each user's social network information is available, we could potentially apply network embedding technique such as Graph Convolutional Network (GCN) [31] to learn distinguishing patterns between malicious accounts and legitimate accounts. In addition, since our framework is flexible, other researchers can try more advanced attention mechanisms such as multi-head attention.

6.3 Case Study: Examples of Misclassified Accounts

Overall, our *Co1* and *Co2* models commonly misclassified 190 accounts: actually 100 legitimate accounts and 90 malicious accounts. Most of those malicious accounts are still alive, and most of those legitimate users are already deleted by the users. Now, we conduct a micro-scale case study to understand why our models misclassified some users/accounts. In particular, we analyze one example of misclassified legitimate account and one example of misclassified malicious account. To protect the privacy of the accounts/users, we intentionally blurred/blocked some information.

Legitimate user. Figure 6 shows an example of the misclassified legitimate ac-

Figure 6: Example of a misclassified legitimate account's profile and tweets.

count. There are several reasons why our models were not able to correctly identify its true label:

- This user mainly wrote tweets in French rather than English. However, the dominating part of our training data was written in English, and thus the pre-trained W2V model may not be working well on the contents it posted. This problem could be solved by introducing/applying better embedding models that incorporate multiple languages.
- Although the user is a person, the user is also a manager of a company. Thus the

profile and tweets/retweets are mostly related to the company. The user does not have a personalized profile photo but rather the scene of buildings. The user also posted more tweets than the average of legitimate users. So there is some chance that our models treated the account as a malicious one.

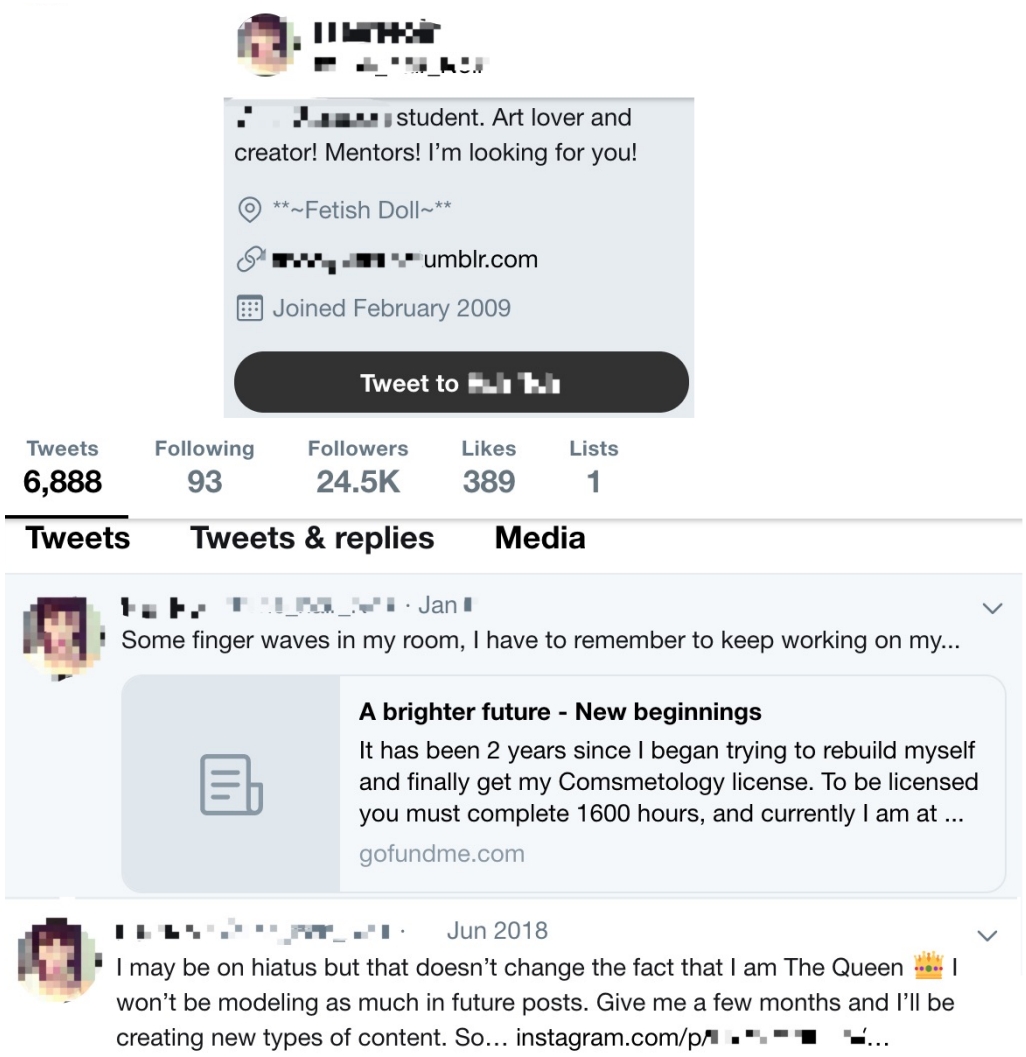

Figure 7: Example of a misclassified malicious account's profile and tweets.

Malicious user. We show an example of the mislabeled malicious account in Figure 7. Although this user has an extremely unbalanced following/follower ratio and embedded URLs in almost every tweet, there are some reasons that such a user escaped our detection:

- This fake account typed in many words before the URLs and the contents he/she posted involved in many different topics. This may make our models got confused. A better way would be to incorporate linked content as well to expand information regarding which page the user referred to.
- This account filled in almost all profile information including fake location information. Social network related features may help correctly identifying the user's true label.

7 Conclusion

In this thesis, we aimed to detect various types of malicious accounts at one sitting. In particular, we combined publicly available three Twitter datasets, which consisted of various types of malicious accounts and legitimate accounts. We grouped accounts into two classes (malicious and legitimate). Then, we proposed a novel joint learning framework based on traditional features, LIWC-based features, C-IPT-CNN features and C-Text-CNN features toward detecting malicious accounts. Experimental results showed that Our framework outperforms all baselines, achieving 0.94 accuracy. We further analyzed why our models performed well and showed how robust our model is under the training data reduction scenario. In the future, we are interested in extending our framework by incorporating network embedding features (e.g., GCN) and advanced attention mechanism (e.g., multi-head attention) to further improve the performance of our models.

References

[1] Kayode Sakariyah Adewole, Nor Badrul Anuar, Amirrudin Kamsin, Kasturi Dewi Varathan, and Syed Abdul Razak. 2017. Malicious accounts: dark of the social networks. *Journal of Network and Computer Applications* 79 (2017), 41–67.

[2] Majid Alfifi and James Caverlee. 2017. Badly Evolved? Exploring Long-Surviving Suspicious Users on Twitter. In *International Conference on Social Informatics*. Springer, 218–233.

[3] Majid Alfifi, Parisa Kaghazgaran, James Caverlee, and Fred Morstatter. 2018. Measuring the Impact of ISIS Social Media Strategy.

[4] Abdullah Almaatouq, Erez Shmueli, Mariam Nouh, Ahmad Alabdulkareem, Vivek K Singh, Mansour Alsaleh, Abdulrahman Alarifi, Anas Alfaris, et al. 2016. If it looks like a spammer and behaves like a spammer, it must be a spammer: analysis and detection of microblogging spam accounts. *International Journal of Information Security* 15, 5 (2016), 475–491.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[6] Iñigo Barandiaran. 1998. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence* 20, 8 (1998).

[7] Matthew C Benigni, Kenneth Joseph, and Kathleen M Carley. 2017. Online extremism and the communities that sustain it: Detecting the ISIS supporting community on Twitter. *PloS one* 12, 12 (2017), e0181405.

[8] David M Beskow and Kathleen M Carley. 2018. Bot conversations are different: leveraging network metrics for bot detection in twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 825–832.

[9] Sajid Yousuf Bhat and Muhammad Abulaish. 2013. Community-based features for identifying spammers in online social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*. IEEE, 100–107.

[10] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906* (2017).

[11] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. 2016. DeBot: Twitter Bot Detection via Warped Correlation.. In *ICDM*. 817–822.

[12] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. 2017. Temporal patterns in bot activities. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 1601–1606.

[13] Nikan Chavoshi and Abdullah Mueen. 2018. Model Bots, not Humans on Social Media. 178–185. `https://doi.org/10.1109/ASONAM.2018.8508279`

[14] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733* (2016).

[15] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.

[16] Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*. American Society for Information Science, 82.

[17] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[18] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2016. DNA-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems* 31, 5 (2016), 58–64.

[19] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 963–972.

[20] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing* 15, 4 (2017), 561–576.

[21] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. 2019. Better Safe Than Sorry: An Adversarial Approach to Improve Social Bot Detection. *arXiv preprint arXiv:1904.05132* (2019).

[22] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 273–274.

[23] John P Dickerson, Vadim Kagan, and VS Subrahmanian. 2014. Using sentiment to detect bots on twitter: Are humans more opinionated than bots?. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, 620–627.

[24] Emilio Ferrara. 2018. Measuring social spam and the effect of bots on information diffusion in social media. In *Complex Spreading Phenomena in Social Systems*. Springer, 229–255.

[25] William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*. 1163–1168.

[26] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. Citeseer, 148–156.

[27] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[28] Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 795–816.

[29] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[30] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[31] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[33] Sneha Kudugunta and Emilio Ferrara. 2018. Deep Neural Networks for Bot Detection. *CoRR* abs/1802.04289 (2018). arXiv:1802.04289 `http://arxiv.org/abs/1802.04289`

[34] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.

[35] Kyumin Lee, Brian David Eoff, and James Caverlee. 2011. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter.. In *ICWSM*. 185–192.

[36] Sangho Lee and Jong Kim. 2014. Early filtering of ephemeral malicious accounts on Twitter. *Computer Communications* 54 (2014), 48–57.

[37] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[38] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting Rumors from Microblogs with Recurrent Neural Networks.. In *IJCAI*. 3818–3824.

[39] Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1751–1754.

[40] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. 2014. Twitter spammer detection using data stream clustering. *Information Sciences* 260 (2014), 64–73.

[41] Fred Morstatter, Liang Wu, Tahora H Nazer, Kathleen M Carley, and Huan Liu. 2016. A new approach to bot detection: striking the balance between precision and recall. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, 533–540.

[42] Julio CS Reis, André Correia, Fabrício Murai, Adriano Veloso, Fabrício Benevenuto, and Erik Cambria. 2019. Supervised Learning for Fake News Detection. *IEEE Intelligent Systems* 34, 2 (2019), 76–81.

[43] Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 797–806.

[44] VS Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. The DARPA Twitter bot challenge. *Computer* 49, 6 (2016), 38–46.

[45] Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506* (2017).

[46] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *Eleventh international AAAI conference on web and social media*.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[48] Bimal Viswanath, Muhammad Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. 2014. Towards Detecting Anomalous User Behavior in Online Social Networks.. In *USENIX Security Symposium*. 223–238.

[49] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 849–857.

[50] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. 2048–2057.

[51] Zen Yoshida and Masayoshi Aritsugi. 2019. Rumor Detection in Twitter with Social Graph Structures. In *Third International Congress on Information and Communication Technology*. Springer, 589–598.