SATAR: A Self-supervised Approach to Twitter Account Representation Learning and its Application in Bot Detection

Shangbin Feng
Xi'an Jiaotong University
Xi'an, China
wind_binteng@stu.xjtu.edu.cn

Herun Wan
Xi'an Jiaotong University
Xi'an, China
wanherun@stu.xjtu.edu.cn

Ningnan Wang
Xi'an Jiaotong University
Xi'an, China
mrwangyou@stu.xjtu.edu.cn

Jundong Li
University of Virginia
Charlottesville, USA
jundong@virginia.edu

Minnan Luo
Xi'an Jiaotong University
Xi'an, China
minnluo@xjtu.edu.cn

ABSTRACT

Twitter has become a major social media platform since its launching in 2006, while complaints about bot accounts have increased recently. Although extensive research efforts have been made, the state-of-the-art bot detection methods fall short of generalizability and adaptability. Specifically, previous bot detectors leverage only a small fraction of user information and are often trained on datasets that only cover few types of bots. As a result, they fail to generalize to real-world scenarios on the Twittersphere where different types of bots co-exist. Additionally, bots in Twitter are constantly evolving to evade detection. Previous efforts, although effective once in their context, fail to adapt to new generations of Twitter bots. To address the two challenges of Twitter bot detection, we propose SATAR, a self-supervised representation learning framework of Twitter users, and apply it to the task of bot detection. In particular, SATAR generalizes by jointly leveraging the semantics, property and neighborhood information of a specific user. Meanwhile, SATAR adapts by pre-training on a massive number of self-supervised users and fine-tuning on detailed bot detection scenarios. Extensive experiments demonstrate that SATAR outperforms competitive baselines on different bot detection datasets of varying information completeness and collection time. SATAR is also proved to generalize in real-world scenarios and adapt to evolving generations of social media bots.

1 INTRODUCTION

Twitter is a popular online social media platform which was released in 2006. Individuals can sign up for a Twitter account to view and publish content of their interests. As reported by Statista[1], the number of daily active Twitter users in the United States is over 35 million in the second quarter of 2020[2]. Twitter has become not only an essential social platform in people's daily life but also an information publishing venue. The open nature and widespread popularity of Twitter have made itself an ideal target of exploitation from automated programs, also known as bots. These bot accounts are often operated to achieve malicious goals. Bots have been actively involved in many important events, including the elections in the United States and Europe [11, 18]. Bots are also responsible for spreading fake news and propagating extreme ideology [2]. These malicious bots try to hide their automated nature by imitating the behaviors of normal users. Across the whole Twittersphere, it is reported that bot accounts for 9% to 15% of total active users [41]. Since bots jeopardize user experience in Twitter and may even induce undesirable social effects, many research efforts have been devoted to Twitter bot detection.

The first work to detect automated accounts in social media dates back to 2010 [41]. Early studies conducted feature engineering and adopted traditional classification algorithms. Three categories of features were considered: (1) user property features [9]; (2) features derived from tweets [30]; and (3) features extracted from neighborhood information [39]. Later, researchers began to propose neural network based bot detection frameworks. Wei *et al.* [38] adopted long short-term memory to extract semantics information from tweets. Kudugunta *et al.* [23] proposed a method that combined feature engineering and neural network models. Heuristic methods for bot detection were also put forward recently. Minnich *et al.* [31] proposed a bot detection method based on anomaly detection. Cresci *et al.* [4] encoded tweets into a string to find out the difference between human and bots in tweeting behaviors.

Despite early successes, ever-shifting social media brought two new challenges to the task of bot detection: generalization and adaptation. The challenge of generalization in social media bot detection demands bot detectors to simultaneously identify bots that attack in many different ways and exploit diversified features on Twitter. Cresci *et al.* [5] points out that Twitter bots attack in different ways such as retweet frauds, malicious hashtag promotion and URL spamming. They also imitate the tweeting behaviour of different types of genuine users, fill out profile items differently and follow each other to boost their follower count. Since Twitter bots are indeed becoming more diversified, a robust Twitter bot detector should therefore address the challenge of generalization to induce real-world impact. However, previous bot detection methods fail to generalize since they only leverage limited user information and are trained on datasets with few types of bots.

Apart from that, the challenge of adaptation in bot detection demands bot detectors to maintain desirable performance in different times and catch up with rapid bot evolution. Cresci *et al.* [3]'s investigation shows that bots in the past used to be simple and easily identified, possessing too little profile and friend information to be genuine. However, more recently evolved bots have large

---

numbers of friends and followers, use stolen profile pictures and intersperse malicious tweets with neutral ones. These newly evolved bots often evade existing detection measures, thus a robust bot detector should address the challenge of adaptation to put an end to the arms race between bot evolution and bot detection research. However, previous bot detection measures rely heavily on feature engineering and are not designed to adapt to emerging trends in bot evolution.

In light of the two challenges of Twitter bot detection, we propose a novel framework SATAR (**S**elf-supervised **A**pproach to **T**witter **A**ccount **R**epresentation learning). SATAR adopts self-supervised learning to obtain user representation and identify bots on social media. Specifically, SATAR jointly encodes tweet, property and neighborhood information of users without feature engineering to promote bot detection generalization. SATAR follows a pre-training and fine-tuning learning schema to adapt to different generations of bots. Our main contributions are summarized as follows:

- We propose a novel framework SATAR to conduct generalizable and adaptable Twitter bot detection. SATAR is an end-to-end framework that jointly uses semantic, property and neighborhood information of users without feature engineering.
- To the best of our knowledge, this paper is the first work to introduce self-supervised representation learning to improve the performance of bot detection.
- We conduct extensive experiments on three real-world datasets to evaluate SATAR and competitive baselines. SATAR outperforms baselines on all three datasets and is proved to generalize and adapt through further exploration.

In the following, we first review related work in Section 2 and define the task of Twitter bot detection in Section 3. Next, we propose SATAR in Section 4, following with extensive experiments in Section 5. Finally, we conclude the whole paper in Section 6.

2 RELATED WORK

In this section, we briefly review the related literature on self-supervised learning and Twitter bot detection.

2.1 Twitter Bot Detection

Traditional bot detection methods mainly focused on extracting basic features from user information. Among them, Gao *et al.* [19] used text shingling and incremental clustering to merge spam messages into campaigns for real-time classification. Lee *et al.* [26] proposed to use the redirection of URLs in tweets and Thomas *et al.* [36] focused on classification of mentioned websites . Other features are also adopted such as information on the user profile [25], social networks [31] and timeline of accounts [4]. Yang *et al.* [39] designed several new features to counter the evolution of modern Twitter bots. Cresci *et al.* [6] proposed that confrontation between bot detectors and bot operators is a never-ending arms race. It is also argued that we should refrain from methods that rely on posterior observations.

Neural networks are also adopted to detect Twitter bots because of their strong learning capability. Wei *et al.* [38] employed recurrent neural networks to efficiently capture features across tweets.

Kudugunta *et al.* [23] divided user features into account-level features, such as follower count, and tweet-level features, such as the number of hashtags. Both kinds of features and semantic information are used to set up an LSTM-based bot detection framework. Stanton *et al.* [34] utilized generative adversarial network for spam detection to avoid annotation costs and inaccuracies. Alhosseini *et al.* [1] proposed a model based on graph convolutional networks for spam bot detection to leverage both node features and neighborhood information. However, these supervised methods rely heavily on annotated data while relevant datasets are typically limited in size. We draw from self-supervised learning to leverage large quantities of unlabeled data.

2.2 Self-Supervised Learning

In order to use unsupervised dataset in a supervised manner, self-supervised learning frames a special learning task, predicting a subset of entities' information using the rest. As a promising learning paradigm, self-supervised learning has drawn massive attention for its fantastic data efficiency and generalization ability, with many state-of-the-art models following this paradigm [27]. Doersch *et al.* [16] combined several self-supervised tasks to jointly train a network. Zhai *et al.* [42] proposed that semi-supervised learning can benefit from self-supervised learning.

Self-supervised learning has been used in different domains, such as natural language processing [12, 43], computer vision [24, 32] and graph analysis [20, 22]. In natural language processing, self-supervised tasks are designed based on following words [33] or the whole sentence [29]. Masked language models are also adopted to better attend to the content in general [12]. In computer vision, adjacent pixels [32, 37] and the full image [14, 15] are used for pretext tasks similarly. In graph analysis, self-supervised tasks are designed based on edge attributes [8, 35] or node attributes [13].

3 PROBLEM DEFINITION

Let U be a Twitter user, consisting of three aspects of user information: semantic T, property P and neighborhood N. Let $T = \{t_i\}_{i=1}^{M}$ be a user's semantic information of M tweets. Each tweet $t_i = \{w_1^i, \cdots, w_{Q_i}^i\}$ contains Q_i words. Let $P = \{p_i\}_{i=1}^{R}$ be a user's property information with a total of R properties. Each property p_i could be numerical such as follower count or categorical such as whether the user is verified. Let $N = \{N^f, N^t\}$, where $N^f = \{N_1^f, \cdots, N_u^f\}$ are u followings of the user and $N^t = \{N_1^t, \cdots, N_v^t\}$ are v followers. Similar to previous research [23, 40], we treat Twitter bot detection as a binary classification problem, where each user could either be human ($y = 0$) or bot ($y = 1$). Formally, we can define the Twitter bot detection task as follows:

> **Problem: Twitter Bot Detection** Given a Twitter user U and its information T, P and N, learn a bot detection function $f : f(U(T, P, N)) \rightarrow \hat{y}$, such that $\hat{y}$ approximates ground truth y to maximize prediction accuracy.
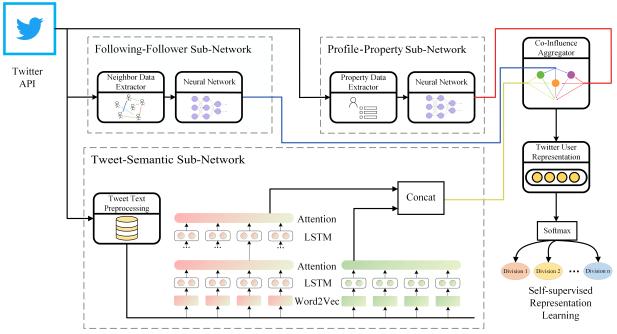
Figure 1: Overview of our proposed framework SATAR.

4 SATAR METHODOLOGY

In this section, we present the details of the proposed Twitter user representation learning framework named as SATAR (**S**elf-supervised **A**pproach to **T**witter **A**ccount **R**epresentation learning).

4.1 Overview

Figure 1 illustrates the proposed framework SATAR. It consists of four major components: (1) a tweet-semantic sub-network, (2) a profile-property sub-network, (3) a following-follower sub-network and (4) a Co-Influence aggregator. Specifically, we use the Twitter API[3] to obtain relevant data regarding a user's semantic, property and neighborhood information. The tweet-semantic sub-network encodes a Twitter user's textual information into r_s with hierarchical RNNs of different depth accompanied by the attention mechanism. The profile-property sub-network encodes a Twitter user's profile properties into r_p with property data encoding and fully connected layers. The following-follower sub-network encodes a Twitter user's neighborhood relationships into r_n with neighborhood information extractor and fully connected layers. Finally, a non-linear Co-Influence aggregator takes the correlation between three aforementioned components into account, generating a representation vector that fully embodies the social status of a specific Twitter user. A softmax layer is then applied for user classification and enables model learning.

4.2 Tweet-Semantic Sub-Network

In this paper, we exploit user semantic information at two different levels, tweet-level and word-level, to capture the tweet content of users. Specifically, words in a user's tweets could be fitted into two hierarchical structures. For tweet-level characterization, as defined in Section 3, w_i^j denotes the i-th word in the j-th tweet of the user timeline, and t_j represents the j-th tweet of a specific user. We also concatenate temporally adjacent tweets: $\{w_1, \cdots, w_K\} = \{w_1^1, \cdots, w_{Q_1}^1, w_1^2, \cdots, w_{Q_M}^M\}$, where the total word count $K = \sum_{i=1}^{M} Q_i$. Thus for word-level characterization, w_k denotes the k-th word in the user's tweet history with temporally adjacent tweets concatenated to form a sequence. It is noteworthy that the underlying words are identical between tweet-level and word-level, but their annotations differ according to the user's tweeting behaviors. To jointly leverage user tweet information on

these two different levels, we propose tweet-level and word-level encoders of hierarchical RNNs to model tweet text sequences respectively and derive an overall semantic representation for Twitter users. The overall semantic representation for Twitter users are concatenated with results of tweet-level and word-level:

$$r_s = concatenation(r_s^t; r_s^w). \tag{1}$$

where r_s^t and r_s^w are representations of tweets on tweet-level and word-level.

Tweet-Level Encoder. The tweet-level encoder follows a bottom-up approach. For the j-th tweet of a specific user, we first embed words in it with an embedding layer:

$$x_i^j = emb(w_i^j), 1 \leqslant i \leqslant Q_j, 1 \leqslant j \leqslant M, \tag{2}$$

where Q_j is the length of the j-th tweet, and we use Word2Vec [29] as the embedding layer $emb(\cdot)$. To encode the tweet, a bidirectional RNN processes the tweet in a forward pass and a backward pass. For the forward pass, a sequence of forward hidden states is generated for the j-th tweet:

$$\overrightarrow{h}_j^t = \left[\overrightarrow{h}_{j,1}^t, \overrightarrow{h}_{j,2}^t, \cdots, \overrightarrow{h}_{j,Q_j}^t \right], \tag{3}$$

where the hidden representation for each step is generated by

$$\overrightarrow{h}_{j,i}^t = RNN\left(\overrightarrow{h}_{j,i-1}^t, x_i^j \right). \tag{4}$$

Here we use LSTM [21] as $RNN(\cdot)$, which is widely adopted to model long-term dependencies in a sequence. For the backward pass, a sequence of backward hidden states is generated similarly:

$$\overleftarrow{h}_j^t = \left[\overleftarrow{h}_{j,1}^t, \overleftarrow{h}_{j,2}^t, \cdots, \overleftarrow{h}_{j,Q_j}^t \right]. \tag{5}$$

We concatenate the forward and backward results to form a sequence of word representations in the j-th tweet:

$$h_j^t = \left[h_{j,1}^t, h_{j,2}^t, \cdots, h_{j,Q_j}^t \right], \tag{6}$$

where $h_{j,i}^t = \left[\overrightarrow{h}_{j,i}^t; \overleftarrow{h}_{j,i}^t \right]$. Since words in a tweet vary in their contribution to the tweet's overall semantic meaning, the attention mechanism is adopted to aggregate word hidden representations into a tweet vector. Specifically,

$$\alpha_{j,i}^t = \frac{exp(u_{j,i}^t \cdot v_l^t)}{\sum_{i'} exp(u_{j,i'}^t \cdot v_l^t)}, \tag{7}$$

where $u_{j,i}^t = tanh(W_l^t h_{j,i}^t + b_l^t)$ transforms vectors for each word and v_l^t, W_l^t and b_l^t are learnable parameters. $\alpha_{j,i}^t$ represents the weight of the i-th word in the j-th tweet. Finally, the representation of the j-th tweet can be obtained as follows:

$$v_j^t = \sum_i \alpha_{j,i}^t h_{j,i}^t. \tag{8}$$

After deriving a vector for each tweet, the tweet-level encoder applies RNN similarly to tweet representations $\{v_j^t\}_{j=1}^M$, generating a forward and a backward sequence. We concatenate the forward and backward results to form a sequence of tweet representations:

---

[3]https://developer.twitter.com/en/products/twitter-api/early-access

$$h^t = \left[h_1^t, h_2^t, \cdots, h_M^t \right], \tag{9}$$

where $h_i^t = \left[\overrightarrow{h}_i^t ; \overleftarrow{h}_i^t \right]$. An attention layer is applied to model the influence each tweet has on the overall semantics of the user:

$$\alpha_i^t = \frac{exp(u_i^t \cdot v_h^t)}{\sum_{i'} exp(u_{i'}^t \cdot v_h^t)}, \tag{10}$$

where $u_i^t = tanh(W_h^t h_j^t + b_h^t)$ transforms vectors for each tweet and v_h^t, W_h^t and b_h^t are learnable parameters. α_i^t represents the weight of the i-th tweet. Finally, the representation of a user's tweet semantics from a tweet-oriented perspective can be obtained as follows:

$$r_s^t = \sum_i \alpha_i^t h_i^t. \tag{11}$$

Word-Level Encoder. The word-level encoder concatenates temporally adjacent tweets into a long sequence of words. For the i-th word of the sequence, we first embed it with the embedding layer identical to the tweet-level encoder:

$$x_i = emb(w_i), 1 \le i \le K, \tag{12}$$

where K is the total word count in the temporally concatenated tweets. A bidirectional RNN with attention is adopted to encode the concatenated sequence. For the forward pass, we have:

$$\overrightarrow{h}^w = \left[\overrightarrow{h}_1^w, \overrightarrow{h}_2^w, \cdots, \overrightarrow{h}_K^w \right], \tag{13}$$

where $\overrightarrow{h}_i^w = RNN(\overrightarrow{h}_{i-1}^w, x_i)$ and LSTM is adopted for $RNN(\cdot)$ regarding its particular length. For the backward pass, we have:

$$\overleftarrow{h}^w = \left[\overleftarrow{h}_1^w, \overleftarrow{h}_2^w, \cdots, \overleftarrow{h}_K^w \right], \tag{14}$$

where $\overleftarrow{h}_i^w = RNN(\overleftarrow{h}_{i+1}^w, x_i)$. Then we concatenate the forward and backward results to form a sequence of word representations in the user's tweet history:

$$h^w = \left[h_1^w, h_2^w, \cdots, h_K^w \right], \tag{15}$$

where $h_i^w = \left[\overrightarrow{h}_i^w ; \overleftarrow{h}_i^w \right]$. Then the attention mechanism is applied:

$$\alpha_i^w = \frac{exp(u_i^w \cdot v^w)}{\sum_{i'} exp(u_{i'}^w \cdot v^w)}, \tag{16}$$

where $u_i^w = tanh(W^w h_i^w + b^w)$, v^w, W^w and b^w are learnable parameters, α_i^w represents the weight of the i-th word in the concatenated sequence. Finally, the representation of a user's tweet semantics from a word-oriented perspective is as follows:

$$r_s^w = \sum_i \alpha_i^w h_i^w. \tag{17}$$

4.3 Profile-Property Sub-Network

To avoid the undesirable bias incorporated in feature engineering, the profile-property sub-network utilizes profile properties that could be directly retrieved from the Twitter API. Different encoding strategies are adopted for different types of property data:

- There are 15 true-or-false property items in total. We use 1 for true and 0 for false. e.g. "profile uses background image".
- There are 5 numerical property items in total. We apply z-score normalization to numerical properties over the whole dataset. e.g. "favorites count".
- There is one special property item: "location". We divide locations geographically into different countries and apply one-hot encoding.

It is noteworthy that the follower count of a specific user would not be included in the property vector, which would be part of the self-supervised learning schema presented in Section 4.6.

The encoded property items are concatenated to form a raw property vector u_p, which is then transformed to produce the Twitter user's property representation r_p:

$$r_p = ReLU(FC_p(u_p)), \tag{18}$$

where $FC_p(\cdot)$ is a fully connected layer and $ReLU(\cdot)$ is a nonlinearity adopted as the activation function.

4.4 Following-Follower Sub-Network

For user followings, according to Twitter mechanism, their tweets will appear in the timeline and the following behaviors often demonstrate interest in their tweet content. Thus we propose u_n^f to model the following relationships:

$$u_n^f = \frac{1}{\sum_{u \in N^f} TF(u)} \sum_{u \in N^f} TF(u) r_s(u), \tag{19}$$

where N^f denotes the following set of a Twitter user, $TF(u)$ denotes the tweet frequency of user u and $r_s(u)$ is the semantic representation of user u generated by the tweet-semantic sub-network. Tweet frequency TF is approximated by a user's total tweet count divided by account active time, which is the time period between a user's registration and its last update. Note that $\frac{TF(u)}{\sum_{u' \in N^f} TF(u')}$ represents the proportion that user u appears in one's timeline, thus u_n^f serves as a weighted sum of followings' semantics information according to their relative tweeting frequency.

For followers, as the average quality of followers of an account defines its social status and the quality could be evaluated by its properties, we propose to model the follower relationships as follows:

$$u_n^t = \frac{1}{|N^t|} \sum_{u \in N^t} r_p(u), \tag{20}$$

where N^t denotes the follower set of a Twitter user, $| \cdot |$ denotes the cardinality of a set and $r_p(u)$ is the property representation of user u generated by the profile-property sub-network.

The following-follower sub-network then produces a raw hidden vector for neighborhood information $u_n = concatenation(u_n^f; u_n^t)$. The intermediate vector is then transformed to produce the Twitter

user's neighborhood representation r_n:

$$r_n = ReLU(FC_n(u_n)), \qquad (21)$$

where $FC_n(\cdot)$ is a fully connected layer and $ReLU(\cdot)$ is the adopted activation function.

4.5 Co-Influence Aggregator

So far, we have obtained the representation vectors regarding three and all three aspects of a Twitter user, namely r_s, r_p and r_n for tweet semantics, user property and follow relationships. A good bot detector should be comprehensive and robust to tamper. In other words, independently considering each aspect of user information would inevitably jeopardize the robustness of the bot detector. Co-attention has been a successful mechanism at handling correlation between two sequences, but it is not designed for mutual influence between multiple representation vectors. Thus we propose a Co-Influence aggregator to take the mutual correlation between tweet semantics, user property and follow relationships into consideration.

Firstly, the affinity index between a pair of aspects is derived:

$$
\begin{aligned}
F_{sp} &= tanh(r_s^T W_{sp} r_p), \\
F_{pn} &= tanh(r_p^T W_{pn} r_n), \\
F_{ns} &= tanh(r_n^T W_{ns} r_s),
\end{aligned}
\qquad (22)
$$

where W_{sp}, W_{pn} and W_{ns} are learnable parameters of the aggregator. A hidden representation for each aspect which incorporates relevant information from the other two aspects are derived:

$$
\begin{aligned}
h^s &= tanh(W_s r_s + F_{sp}(W_p r_p) + F_{ns}(W_n r_n)), \\
h^p &= tanh(W_p r_p + F_{sp}(W_s r_s) + F_{pn}(W_n r_n)), \\
h^n &= tanh(W_n r_n + F_{ns}(W_s r_s) + F_{pn}(W_p r_p)),
\end{aligned}
\qquad (23)
$$

where W_s, W_p and W_n are learnable parameters of the aggregator. Finally, the proposed framework SATAR produces the Twitter user representation r as follows:

$$r = tanh(W_V \cdot concatenation(h^s; h^p; h^n)), \qquad (24)$$

where W_V is a learnable parameter of the aggregator.

4.6 Self-Supervised Learning and Optimization

Twitter user representation learning attempts to model a specific user with a distributed representation. We adopt **follower count** as the self-supervised signal for SATAR training. Specifically, a user's follower count is separated into several categories based on its numerical scale and the overall follower count distribution. We train the representation learning framework SATAR to classify each user into such categories, obtaining user representation in the process. We believe that **follower count** would be an ideal self-supervised training signal due to the following reasons:

- Self-supervised training with follower count is task-agnostic. Whether it is bot detection, content recommendation or online campaign modeling, follower count relates to all tasks on social media without being specific to any of them.
- Follower count is most representative of a Twitter user. There is no better choice to describe a Twitter user more efficiently and accurately, especially when follower count also involves the evaluation of other users.

Algorithm 1: SATAR Learning Algorithm

Input: Twitter user dataset TU, each user $u \in TU$ has tweets T, properties P and neighbors N

Output: SATAR-optimized parameters θ

Initialize θ;

for *each user $u \in TU$* **do**
  Initialize $r_n(u)$;
  $u.y \leftarrow$ self-supervised label assignment according to user u's follower count;
end

while θ *has not converged* **do**
  for *each user $u \in TU$* **do**
    $r_s(u) \leftarrow$ Equation (2 - 1) with $u.T$;
    $r_p(u) \leftarrow$ Equation (18) with $u.P$;
    $r(u) \leftarrow$ Equation (22 - 24) with $r_s(u)$, $r_p(u)$ and $r_n(u)$;
    $L_u \leftarrow$ Equation (25 - 26) with $r(u)$ and $u.y$;
  end
  $\theta \leftarrow$ BackPropagate(L_u);
  for *each user $u \in TU$* **do**
    $r_n(u) \leftarrow$ Equation (19 - 21) with $u.N$;
  end
end

- Follower count is more robust to large-scale tamper. Although it is possible to purchase fake followers, according to Cresci *et al.* [7]'s investigation, an increase of 1,000 followers often costs from 13 to 19 U.S. dollars. As a result, it is costly to significantly alter the magnitude of a user's follower count, let alone launch a campaign with many active bots.

Specifically, assuming that a user could be categorized into D classes based on its follower count, a softmax layer is applied to the representation of the user r:

$$\hat{y} = softmax(W_f r + b_f), \qquad (25)$$

where $\hat{y} = [\hat{y_1}, \hat{y_2}, \cdots, \hat{y_D}]$ is the predicted probability vector for each class, W_f and b_f are learnable parameters. $y = [y_1, y_2, \cdots, y_D]$ denotes the self-supervised ground-truth for such classification in one-hot encoding. We minimize the cross-entropy loss function as follows:

$$L(\theta) = -\sum_{1 \leqslant i \leqslant D} y_i log(\hat{y_i}), \qquad (26)$$

where θ denotes the parameters in the proposed framework SATAR.

Algorithm 1 presents the overall training schema of our proposed Twitter account representation learning framework SATAR.

5 EXPERIMENTS

In this section, we conduct extensive experiments with in-depth analysis on three real-world bot detection datasets.

5.1 Experiment Settings

In this section, we provide information about datasets, bot detection baselines and evaluation metrics adopted in the experiments.

Datasets. We make use of three datasets, `TwiBot-20`, `cresci-17` and `PAN-19`. As Twitter bots bear different purposes and evolve

Table 1: Components of Twitter user information used by each bot detection method.

	Lee *et al.* [25]	Yang *et al.* [40]	Kudugunta *et al.* [23]	Wei *et al.* [38]	Miller *et al.* [30]	Cresci *et al.* [4]	Botometer [10]	Alhosseini *et al.* [1]	SATAR$_{FC}$	SATAR$_{FT}$
Semantic	✓		✓	✓	✓	✓	✓		✓	✓
Property	✓	✓	✓		✓		✓	✓	✓	✓
Neighbor							✓	✓	✓	✓

Table 2: Performance comparison for bot detection methods. "/" denotes insufficient user information to support the baseline.

		Lee *et al.* [25]	Yang *et al.* [40]	Kudugunta *et al.* [23]	Wei *et al.* [38]	Miller *et al.* [30]	Cresci *et al.* [4]	Botometer [10]	Alhosseini *et al.* [1]	SATAR$_{FC}$	SATAR$_{FT}$
TwiBot-20	Acc	0.7456	0.8191	0.8174	0.7126	0.4801	0.4793	0.5584	0.6813	0.7838	**0.8412**
	F1	0.7823	0.8546	0.7517	0.7533	0.6266	0.1072	0.4892	0.7318	0.8084	**0.8642**
	MCC	0.4879	0.6643	0.6710	0.4193	-0.1372	0.0839	0.1558	0.3543	0.5637	**0.6863**
Cresci-17	Acc	0.9750	0.9847	0.9799	0.9670	0.5204	0.4029	0.9597	/	0.9622	**0.9871**
	F1	0.9826	0.9893	0.9641	0.9768	0.4737	0.2923	0.9731	/	0.9737	**0.9910**
	MCC	0.9387	0.9625	0.9501	0.9200	0.1573	0.2255	0.8926	/	0.9069	**0.9685**
PAN-19	Acc	/	/	/	0.9464	/	0.8797	/	/	0.8728	**0.9509**
	F1	/	/	/	0.9448	/	0.8701	/	/	0.8729	**0.9510**
	MCC	/	/	/	0.8948	/	0.7685	/	/	0.7456	**0.9018**

Table 3: Overview of three adopted bot detection datasets.

Dataset	User Count	Human Count	Bot Count
TwiBot-20	229,573	5,237	6,589
Cresci-17	9,813	2,764	7,049
PAN-19	11,378	5,765	5,613

(a) train on politics domain

(b) train on business domain

(c) train on entertainment domain

(d) train on sports domain

Figure 2: Train SATAR and two competitive baselines on one domain of TwiBot-20 and test on the other three domains.

rapidly, these high quality datasets are adopted to provide a comprehensive evaluation and verify the generalizability and adaptability of baselines and our proposed method.

- `TwiBot-20` [17] is a comprehensive sample of the current Twittersphere to evaluate whether bot detection methods can generalize in real-world scenarios. Users in `TwiBot-20` could be generally split into four interest domains: politics, business, entertainment and sports. As of user information, `TwiBot-20` contains semantic, property and neighborhood information of Twitter users.
- `cresci-17` [5] is a public dataset with 4 components: genuine accounts, social spambots, traditional spambots and fake followers. We merge the four parts and utilize `cresci-17` as a whole. `cresci-17` contains semantic and property information.
- `PAN-19` [4] is a dataset of a Bots and Gender Profiling shared task in the PAN workshop at CLEF 2019. It is used for bots and gender profiling and only contains user semantic information.

A summary of these three datasets is presented in Table 3. We randomly conduct a 7:2:1 partition for three datasets as training, validation and test set. Such a partition is shared across all experiments in Section 5.2, Section 5.3 and Section 5.4. We choose these three benchmarks out of numerous bot detection datasets due to their larger size, collection time span and superior annotation quality.
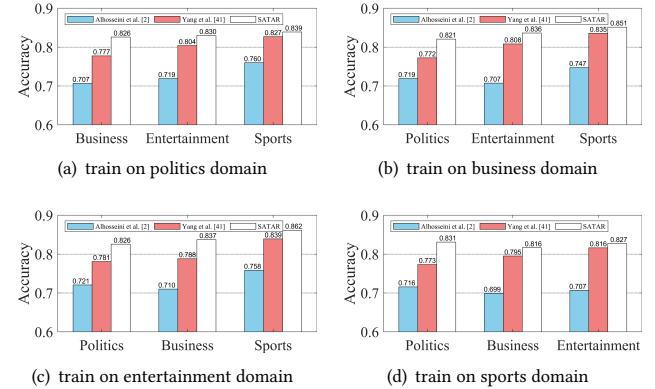
Baseline Methods. We compare SATAR with the following bot detection methods as baselines:

- Lee *et al.* [25]: Lee *et al.* use random forest classifier with several Twitter user features. e.g. the longevity of the account.
- Yang *et al.* [40]: Yang *et al.* use random forest with minimal account metadata and 12 derived features.
- Kudugunta *et al.* [23]: Kudugunta *et al.* propose an architecture that uses both tweet content and the metadata.
- Wei *et al.* [38]: Wei *et al.* use word embeddings and a three-layer BiLSTM to encode tweets. A fully connected softmax layer is adopted for binary classification.
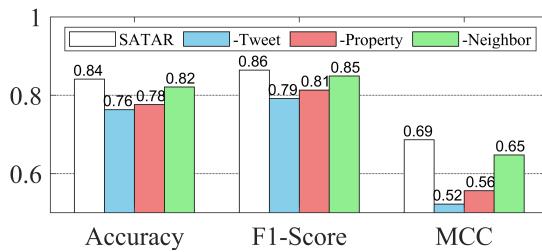
Figure 3: Ablation study that removes the semantic, property and neighborhood sub-networks from SATAR.

- Miller *et al.* [30]: Miller *et al.* extract 107 features from a user's tweet and property information. Bot users are conceived as abnormal outliers and modified stream clustering algorithm is adopted to identify Twitter bots.
- Cresci *et al.* [4]: Cresci *et al.* utilize strings to represent the sequence of a user's online actions. Each action type can be encoded with a character. By identifying the group of accounts that share the longest common substring, a set of bot accounts are obtained.
- Botometer [10]: Botometer is a publicly available service that leverages more than one thousand features to classify an account.
- Alhosseini *et al.* [1]: Alhosseini *et al.* utilize graph convolutional network to detect Twitter bots. It uses following information and user features to learn representations and classify Twitter users.

For the following SATAR-based bot detection methods, the self-supervised representation learning step adopts the Pareto Principle[5] as a self-supervised classification task, where the framework learns to predict whether a Twitter user's follower count is among the top 20% or the bottom 80%. It is an instance of the self-supervised representation learning strategy in Section 4.6.

- SATAR$_{FC}$: The proposed representation learning framework SATAR is firstly trained with self-supervised user classification tasks based on their follower count, then the final softmax layer is reinitialized and trained on the task of bot detection.
- SATAR$_{FT}$: The proposed representation learning framework SATAR is firstly trained using self-supervised users, then the final softmax layer is reinitialized and fine-tuning is performed on the whole framework using the training set of bot detection.

Evaluation Metrics. We adopt Accuracy, F1-score and MCC [28] as evaluation metrics of different bot detection methods. Accuracy is a straightforward indicator of classifier correctness, while F1-score and MCC are more balanced evaluation metrics.

5.2 Bot Detection Performance

Table 1 identifies the user information that each compared method uses. Table 2 reports bot detection performance of different methods on three datasets. Table 2 demonstrates that:

- SATAR based methods achieve competitive performance compared with other baselines, which demonstrates that SATAR is generally effective in Twitter bot detection. SATAR$_{FT}$ outperforms SATAR$_{FC}$, which demonstrates the efficacy of the pre-training and fine-tuning approach.

---

[5]https://en.wikipedia.org/wiki/Pareto_principle

- SATAR$_{FT}$ generalizes to real-world scenarios because it outperforms the state-of-the-art methods on the comprehensive and representative dataset `TwiBot-20`, which imitates the real-world Twittersphere. Meanwhile, SATAR$_{FT}$ adapts to evolving generations of bots because it achieves the best performance on all three datasets with varying collection time from 2017 to 2020. Section 5.3 and Section 5.4 will provide further analysis to demonstrate that SATAR successfully addresses the challenges of generalization and adaptation, while critical components and design choices of SATAR are the reasons behind its success.
- For methods mainly based on LSTM, we see that Kudugunta *et al.* [23] outperforms Wei *et al.* [38]. It indicates that Kudugunta *et al.* [23] can better capture bots by incorporating property items. SATAR$_{FT}$ leverages even more user information than Kudugunta *et al.* [23] and achieves better performance, which suggests that bot detection methods should incorporate more aspects of user information.
- Feature-engineering based methods, such as Yang *et al.* [40], perform well on `cresci-17` but inferior to SATAR$_{FT}$ on `TwiBot-20`. This shows that traditional bot detection methods that emphasize feature engineering fail to adapt to new generations of bots.
- Both Alhosseini *et al.* [1] and SATAR use neighborhood information. SATAR based methods outperform Alhosseini *et al.* [1], which shows that SATAR better utilizes user neighbors that put Twitter users into their social context.

5.3 SATAR Generalization Study

The challenge of generalization in social media bot detection demands bot detectors to simultaneously identify bots that attack in many different ways and exploit diversified user information. To prove that SATAR generalizes, we examine SATAR and competitive baselines' performance on `TwiBot-20`. As demonstrated in Table 2, SATAR outperforms all baselines on `TwiBot-20`. Given the fact that `TwiBot-20` contains diversified bots and human which imitates the real-world Twittersphere, SATAR is demonstrated to best generalize in real-world scenarios.

To further prove SATAR's generalizability, we train SATAR and two competitive baselines, Alhosseini *et al.* [1] and Yang *et al.* [40], on one of the four user domains and test on the others. The results are presented in Figure 2. It is illustrated that SATAR could better capture other types of bots even when not explicitly trained on them, which further establishes the claim that SATAR successfully generalizes to diversified bots that co-exist on social media.

SATAR is designed to generalize by jointly leveraging all three aspects of user information, namely semantic, property and neighborhood information. To figure out whether our proposal of using as much user information as possible has lead to the generalizability of SATAR, we conduct ablation study that removes one aspect of user information at a time. The results are demonstrated in Figure 3.

Results in Figure 3 show that removing any aspect of information from SATAR would result in a considerable loss in performance, limiting SATAR's ability to generalize to different types of bots. It indicates that SATAR's strategy of leveraging more aspects of information is crucial in its generalization.
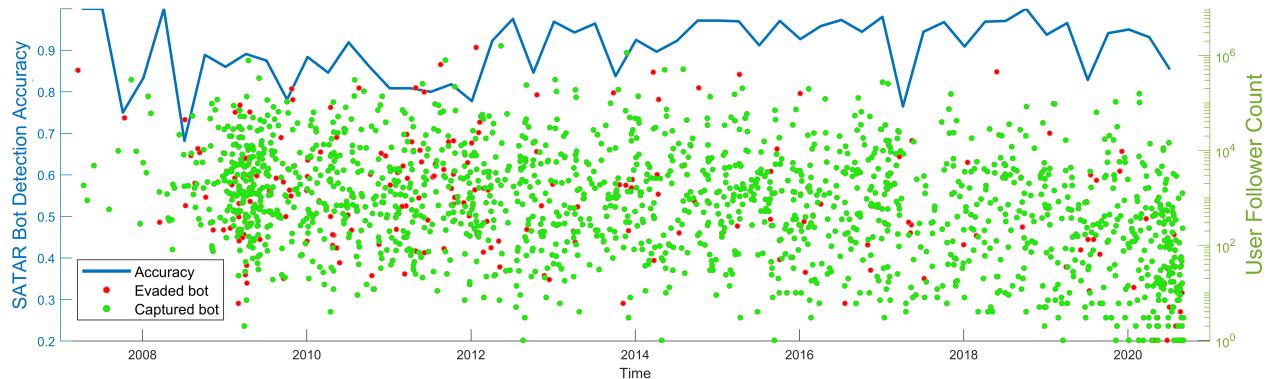
Figure 4: SATAR's prediction of specific users in TwiBot-20. Scattered points demonstrate SATAR's prediction for specific users and the line indicates SATAR's overall accuracy of capturing bots registered in a 3-month time span.
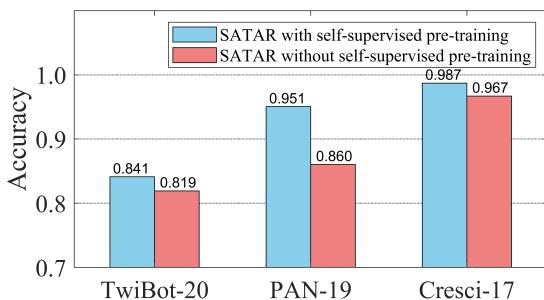

Figure 5: Ablation study removing the self-supervised pre-training step from SATAR and train on the three datasets.

5.4 SATAR Adaptation Study

The challenge of adaptation in bot detection demands bot detectors to maintain desirable performance in different times and catch up with rapid bot evolution. To prove that SATAR adapts, we examine SATAR and competitive baselines' performance on three datasets, since they are released in 2017, 2019 and 2020 respectively and could well characterize the bot evolution. Results in Table 2 demonstrate that SATAR reaches state-of-the-art performance on all three datasets, which indicates that SATAR is more successful at adapting to the bot evolution than existing baselines.

To further prove SATAR's ability to adapt, we examine SATAR's prediction of users in dataset `TwiBot-20`'s validation set and test set. We present SATAR's prediction results of specific users and SATAR's accuracy in any 3-month time span of user registration time in Figure 4. It is illustrated that SATAR maintains a steady detection accuracy for users created from 2007 to 2020, which further establishes the claim that SATAR successfully adapts to the everlasting bot evolution.

SATAR is designed to adapt by pre-training on mass self-supervised users and fine-tuning on specific bot detection scenarios. To figure out whether this pre-training and fine-tuning schema has enabled SATAR to adapt to newly evolved bots, we conduct ablation study to remove the self-supervised pre-training step. SATAR's performance on different datasets are illustrated in Figure 5.

Figure 5 shows that SATAR's performance increases with the adoption of the self-supervised pre-training step, and such trend is especially salient on the dataset PAN-19 with less user information. It indicates that SATAR's ability to adapt indeed comes from the innovative strategy to use follower count as a self-supervised signal for user representation pre-training.

5.5 Representation Learning Study

SATAR improves representation learning for Twitter users. Extrinsic evaluation has proven that SATAR representations are of desirable quality. We further conduct intrinsic evaluation by comparing SATAR representations with Alhosseini *et al.* [1] and Yang *et al.* [40], which also provide user representations. We cluster representations using k-means with $k = 2$, and calculate the homogeneity score, which is the extent to which clusters contain a single class. Higher homogeneity score indicates that users with the same label are more likely to be close to each other.

Figure 6 visualizes representations of users in a subgraph of `TwiBot-20`. Figure 6(a) is the t-SNE plot of SATAR representations, which shows moderate collocation for groups of bot and human, while Figure 6(b) and (c) show little collocation. Quantitatively, SATAR achieves the highest homogeneity score, which indicates that SATAR produces user representations of higher quality.

5.6 Case Study

To further understand how SATAR identifies bots, we study a specific case of several bots. We use the affinity index values in Equation (22) to quantitatively analyze SATAR's decision making. Figure 7 shows the detailed information of the sampled users:

- SATAR identifies user B and E through their repeated or similar tweets that signal automation. For example, user B has affinity values of $F_{sp} = -0.9989$, $F_{pn} = 0.0017$ and $F_{ns} = 0.6376$. Absolute values of F_{sp} and F_{ns} are significantly greater than F_{pn}, which demonstrates that semantic information is the dominant factor for SATAR's decision in this case.
- SATAR identifies user C and D through their properties. Abnormal characteristics such as too many followings and default
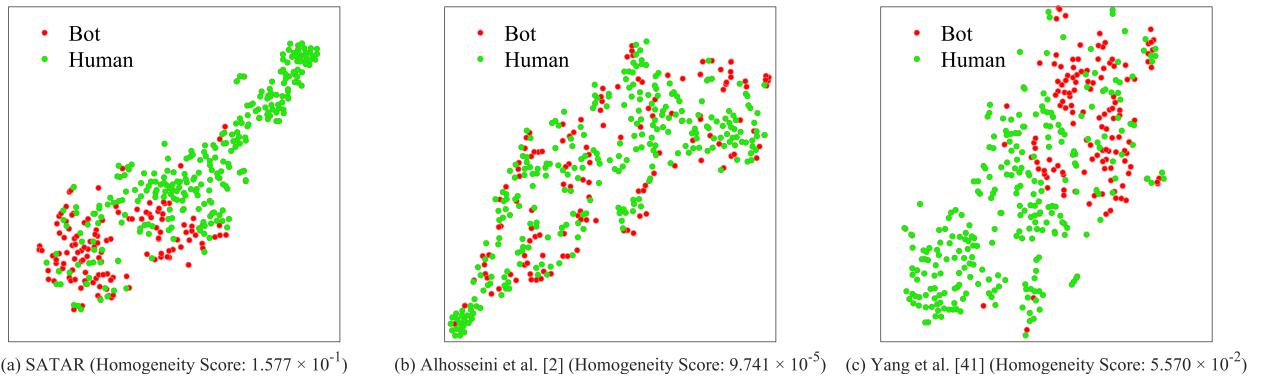
(a) SATAR (Homogeneity Score: 1.577×10^{-1})   (b) Alhosseini et al. [2] (Homogeneity Score: 9.741×10^{-5})   (c) Yang et al. [41] (Homogeneity Score: 5.570×10^{-2})

Figure 6: 2D t-SNE plot of the user representation vectors of SATAR, Alhosseini *et al.* [1] and Yang *et al.* [40].
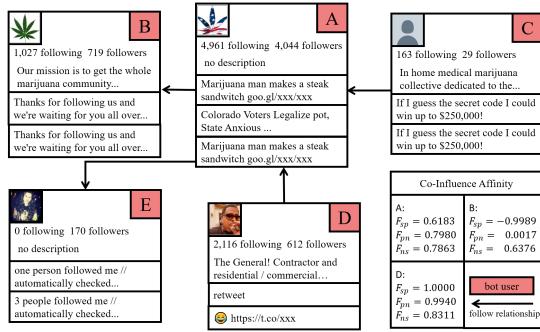

Figure 7: A sample bot cluster to explain SATAR's decision.

background image are detected by SATAR. User D has larger absolute values for F_{sp} and F_{pn} than F_{ns}, which shows that property information is critical in SATAR's judgement.

- SATAR captures that user A has four bots as neighbors, which is unlikely for genuine users. User A has larger absolute values for F_{ns} and F_{pn} than F_{sp}, which also bears out the claim that user A's abnormal neighborhood has led to SATAR's decision.

The case study in Figure 7 demonstrates that SATAR identifies bot users by jointly evaluating their semantic, property and neighborhood information. Affinity values of our proposed Co-Influence aggregator provides explanation to SATAR's decisions.

6 CONCLUSION AND FUTURE WORK

Social media bot detection is attracting growing attention. We proposed SATAR, a self-supervised approach to Twitter account representation learning and applied it to the task of bot detection. SATAR aims to tackle the challenges of generalizing in real-world scenarios and adapting to bot evolution, where previous efforts failed. We conducted extensive experiments to demonstrate the efficacy of SATAR-based bot detection in comparison to competitive baselines. Further exploration proved that SATAR also succeeded in generalizing on the real Twittersphere and adapting to different generations of Twitter bots. In the future, we plan to apply the SATAR representation learning framework to other tasks in the social media domain such as fake news detection and content recommendation.

REFERENCES

[1] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. 2019. Detect Me If You Can: Spam Bot Detection Using Inductive Representation Learning. In *Companion Proceedings of The 2019 World Wide Web Conference*. 148–153.

[2] Jonathon M Berger and Jonathon Morgan. 2015. The ISIS Twitter Census: Defining and describing the population of ISIS supporters on Twitter. *The Brookings project on US relations with the Islamic world* 3, 20 (2015), 4–1.

[3] Stefano Cresci. 2020. A Decade of Social Bot Detection. *Commun. ACM* 63, 10 (Sept. 2020), 72–83. https://doi.org/10.1145/3409116

[4] 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.

[5] 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*. 963–972.

[6] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. 2018. From reaction to proaction: Unexplored ways to the detection of evolving spambots. In *Companion Proceedings of the The Web Conference 2018*. 1469–1470.

[7] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. 2015. Fame for sale: Efficient detection of fake Twitter followers. *Decis. Support Syst.* 80 (2015), 56–71.

[8] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[9] Eleonora D'Andrea, Pietro Ducange, Beatrice Lazzerini, and Francesco Marcelloni. 2015. Real-time detection of traffic from twitter stream analysis. *IEEE transactions on intelligent transportation systems* 16, 4 (2015), 2269–2283.

[10] 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*. 273–274.

[11] Ashok Deb, Luca Luceri, Adam Badaway, and Emilio Ferrara. 2019. Perils and Challenges of Social Media and Election Manipulation Analysis: The 2018 US Midterms. In *Companion Proceedings of The 2019 World Wide Web Conference* (San Francisco, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 237–247. https://doi.org/10.1145/3308560.3316486

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Ming Ding, Jie Tang, and Jie Zhang. 2018. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 913–922.

[14] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[16] Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 2051–2060.

[17] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark. *arXiv preprint arXiv:2106.13088* (2021).

[18] Emilio Ferrara. 2017. Disinformation and Social Bot Operations in the Run Up to the 2017 French Presidential Election. *CoRR* abs/1707.00086 (2017). arXiv:1707.00086 http://arxiv.org/abs/1707.00086

[19] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. 2012. Towards online spam filtering in social networks.. In *NDSS*, Vol. 12. 1–16.

[20] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[22] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[23] Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467 (2018), 312–322.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning representations for automatic colorization. In *European conference on computer vision*. Springer, 577–593.

[25] Kyumin Lee, Brian Eoff, and James Caverlee. 2011. Seven months with the devils: A long-term study of content polluters on twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 5.

[26] Sangho Lee and Jong Kim. 2013. Warningbird: A near real-time detection system for suspicious urls in twitter stream. *IEEE transactions on dependable and secure computing* 10, 3 (2013), 183–195.

[27] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218* 1, 2 (2020).

[28] Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405, 2 (1975), 442–451.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013), 3111–3119.

[30] 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.

[31] Amanda Minnich, Nikan Chavoshi, Danai Koutra, and Abdullah Mueen. 2017. BotWalk: Efficient adaptive exploration of Twitter bot networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 467–474.

[32] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328* (2016).

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

[34] Gray Stanton and Athirai A Irissappane. 2019. GANs for semi-supervised opinion spam detection. *arXiv preprint arXiv:1903.08289* (2019).

[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[36] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. 2011. Design and evaluation of a real-time url spam filtering service. In *2011 IEEE symposium on security and privacy*. IEEE, 447–462.

[37] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *International Conference on Machine Learning*. PMLR, 1747–1756.

[38] Feng Wei and Uyen Trang Nguyen. 2019. Twitter Bot Detection Using Bidirectional Long Short-term Memory Neural Networks and Word Embeddings. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 101–109.

[39] Chao Yang, Robert Harkreader, and Guofei Gu. 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security* 8, 8 (2013), 1280–1293.

[40] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. 2020. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1096–1103.

[41] Sarita Yardi, Daniel Romero, Grant Schoenebeck, et al. 2010. Detecting spam in a twitter network. *First Monday* (2010).

[42] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. 2019. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*. 1476–1485.

[43] Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566* (2019).