RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter

Michele Mazza
IIT-CNR, Italy
michele.mazza@iit.cnr.it

Stefano Cresci*
IIT-CNR, Italy
stefano.cresci@iit.cnr.it

Marco Avvenuti
University of Pisa, Italy
marco.avvenuti@unipi.it

Walter Quattrociocchi
Ca' Foscari University of Venice, Italy
w.quattrociocchi@unive.it

Maurizio Tesconi
IIT-CNR, Italy
maurizio.tesconi@iit.cnr.it

ABSTRACT

Within OSNs, many of our supposedly online friends may instead be fake accounts called *social bots*, part of large groups that purposely re-share targeted content. Here, we study retweeting behaviors on Twitter, with the ultimate goal of detecting retweeting social bots. We collect a dataset of 10M retweets. We design a novel visualization that we leverage to highlight benign and malicious patterns of retweeting activity. In this way, we uncover a "normal" retweeting pattern that is peculiar of human-operated accounts, and 3 suspicious patterns related to bot activities. Then, we propose a bot detection technique that stems from the previous exploration of retweeting behaviors. Our technique, called Retweet-Buster (RTbust), leverages unsupervised feature extraction and clustering. An LSTM autoencoder converts the retweet time series into compact and informative latent feature vectors, which are then clustered with a hierarchical density-based algorithm. Accounts belonging to large clusters characterized by malicious retweeting patterns are labeled as bots. RTbust obtains excellent detection results, with $F1 = 0.87$, whereas competitors achieve $F1 \leq 0.76$. Finally, we apply RTbust to a large dataset of retweets, uncovering 2 previously unknown active botnets with hundreds of accounts.

CCS CONCEPTS

• **Information systems** → *Social networking sites.*

KEYWORDS

Social bots, retweet patterns, OSN security, Twitter

ACM Reference Format:
Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. 2019. RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter. In *11th ACM Conference on Web Science (WebSci '19), June 30-July 3, 2019, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292522.3326015

*This is the corresponding author.

WebSci '19, June 30-July 3, 2019, Boston, MA, USA

ACM ISBN 978-1-4503-6202-3/19/06…$15.00
https://doi.org/10.1145/3292522.3326015

1 INTRODUCTION

In 2016 the Oxford dictionary elected "Post-Truth" as the word of the year and in 2017 Collins dictionary did the same for "Fake News". In 2017 the World Economic Forum raised a warning on the potential distortion effect of social media on user perceptions of reality[1]. Recent studies, targeting Facebook, showed the tendency of the users to interact with information adhering to their preferred narrative [3, 13] and to ignore dissenting information [45]. Confirmation bias seems to account for user decisions about consuming and spreading content and at the same time, aggregation of favored information within those communities reinforces selective exposure and group polarization [38]. As we interact with our peers, we are exposed to the effects of echo chambers, which may result in polarization and hate speech [14]. In this scenario the role of bots is not clear, both for the difficulties in quantifying their impact as well as for the accuracy of detection algorithms. However, any of our supposedly online friends may instead be fake, automated accounts (*social bots*), and a part of large coordinated groups that purposely (re-)share targeted content [16]. Along this challenge, in this work we propose a new method for the detection of groups of bots, accounting for their statistical traces induced by their coordinated behavior on Twitter.

Specifically, we focus on re-sharing (i.e., retweeting) patterns with the ultimate goal of detecting retweeting social bots. Indeed, by artificially boosting retweet counts, these bots may affect users' popularity or influence, thus "reshaping political debates [40]. They can defraud businesses and ruin reputations"[2]. Moreover, large retweet counts indicate influential users, and can be monetized. Thus, there are strong economic and sociopolitical incentives for malicious bots to tamper with retweets [18, 41]. Hence, fast and accurate removal of these bots might be crucial for ensuring the healthiness of our online social ecosystems.

Both researchers and OSN administrators have recently been very active towards the detection of social bots, and many different techniques have been proposed to this end. Unfortunately, the same also applies to malicious bot developers. In fact, as soon as new detection techniques are deployed, bot developers tweak the characteristics of their accounts, thus allowing them to evade detection [8]. This "never-ending clash" led to the current situation where social bots are so sophisticatedly engineered as to mimic legitimate accounts, becoming almost indistinguishable from them [42]. A

[1] http://reports.weforum.org/global-risks-2017/acknowledgements/
[2] https://www.nytimes.com/interactive/2018/01/27/technology/social-media-bots.html

straightforward consequence of this situation is that standard machine learning classification approaches, where each account is analyzed individually, are no longer profitable. Instead, the scientific frontier of bot detection now focuses on groups of suspicious accounts as a whole. Analyzing groups has the advantage that, no matter how sophisticated a single bot can be, a large enough group of them will still leave traces of automation, since they do share a common goal (e.g., increasing someone's popularity) [8]. For the same reasons, unsupervised approaches are preferred over supervised ones [44].

The possibility to exploit more data for the analysis, opened up by the approaches that target groups rather than individual accounts, is however counterbalanced by the difficulties in collecting and processing that much data. For instance, the behavior-based technique described in [7, 9] requires collecting and comparing data of the Twitter timelines of all analyzed accounts. Similarly, graph-based techniques such as [26] require building and analyzing the social graph of a large group of accounts. The amount of data and computational power needed to complete these analyses, inevitably limits the large-scale applicability of these techniques. In short, the next generation of bot detection techniques should strike the balance between accuracy and data/algorithmic efficiency.

Contributions. We propose a novel technique for the detection of retweeting social bots, called Retweet-Buster (RTbust), having all the desirable features previously discussed. Our technique only requires the timestamps of retweets (and of the retweeted tweets) for each analyzed account, thus avoiding the need for full user timelines or social graphs. Then, it compares the temporal patterns of retweeting activity of large groups of users. Leveraging previous findings in the field, RTbust looks for groups of accounts with distinctive and synchronized patterns. Evaluation results on a large dataset of retweets demonstrate excellent bot detection performances, with $F1 = 0.87$, whereas competitors achieve $F1 \leq 0.76$. Summarizing, our detailed contributions are as follows.

- We analyze retweeting behaviors of a large set of users by introducing a simple – yet effective – visualization, which we then leverage to highlight benign and malicious patterns of retweeting activity.
- We design a group-analysis technique that is capable of detecting accounts having the same retweeting patterns. Accounts belonging to a large group characterized by the same malicious patterns are labeled as bots.
- We compare detection results of our technique with those obtained by baselines and other state-of-the-art techniques, demonstrating the effectiveness of our approach.
- By applying our technique to a large dataset of retweets, we uncover 2 previously unknown active botnets.

2 RELATED WORK IN BOT DETECTION

The vast majority of previous attempts at bot detection are based on supervised machine learning [8]. The first challenge in developing a supervised detector is related to the availability of a ground truth (i.e., labeled) dataset, to be used in the learning phase of the classifier. In most cases, a real ground truth is lacking and the labels are simply given by human operators that manually analyze the data. Critical issues arise since there is no "standard" definition of what a social bot is [34, 44]. Moreover, humans have been proven to largely fail at spotting sophisticated bots, with only $\simeq$ 24% bots correctly labeled as such [8]. As anticipated, these criticalities support the development of unsupervised techniques.

Regarding features to exploit for the detection, 3 classes have been mainly considered: (i) profile features [12, 28]; (ii) features extracted from the posts, such as posting behavior and content of posted messages [5, 30, 36]; and (iii) features derived from the social or interaction graph of the accounts [26, 27, 31]. The classes of features exploited by the detection technique have a strong impact on both the performances of the detector as well as its efficiency [6]. For instance, in Twitter it has been demonstrated that those features that mostly contribute towards the predictive power of bot detectors (e.g., measures of centrality in the social graph), are also the most costly ones.

The difficulties in detecting sophisticated bots with supervised approaches that are based on the analysis of individual accounts, recently gave rise to a new research trend that aims to analyze groups of accounts as a whole [8]. This new approach to bot detection is proving particularly effective at detecting coordinated and synchronized bots, such as those targeted in our work. For instance, the technique discussed in [7, 9] associates each account to a sequence of characters that encodes its behavioral information. Such sequences are then compared between one another to find anomalous similarities among sequences of a subgroup of accounts. The similarity is computed by measuring the longest common subsequence shared by all the accounts of the group. Accounts that share a suspiciously long subsequence are then labeled as bots. Instead, the family of systems described in [26, 31, 32] build a bipartite graph of accounts and their interactions with content (e.g., retweets to some other tweets) or with other accounts (e.g., becoming followers of other accounts). Then, they aim to detect anomalously dense blocks in the graph, which might be representative of coordinated and synchronized attacks. A possible drawback of group approaches is that they exacerbate challenges related to data and algorithmic costs, since they typically involve a large number of comparisons between all accounts in a given group [9, 43].

Lastly, a trailblazing direction of research involves the application of adversarial machine learning to bot detection. Until now, bot detection has mostly followed a *reactive* schema, where countermeasures are taken only after having witnessed evidence of bot mischiefs [42]. Instead, in [42] is proposed an adversarial framework enabling *proactive* analyses. The framework has been later instantiated in [10] by using evolutionary algorithms to test current detection techniques against a wider range of unseen, sophisticated bots. Another preliminary work in adversarial bot detection is described in [21]. Among the positive outcomes of adversarial approaches to bot detection, is a more rapid understanding of the drawbacks of current detectors and the opportunity to gain insights into new features for achieving more robust and more reliable detectors.

3 DATA COLLECTION AND ANNOTATION

Our dataset for this study is composed of *all* Italian retweets shared in a 2 weeks time span – specifically, between 18 June 2018 and 1 July 2018 (inclusive). Overall, our dataset comprises 9,989,819 retweets, shared by 1,446,250 distinct users. Thus, on average, each

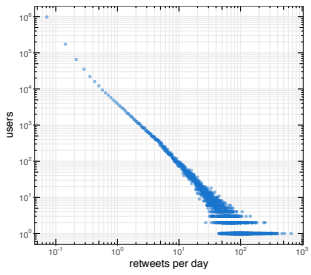


Figure 1: Daily retweets per user.

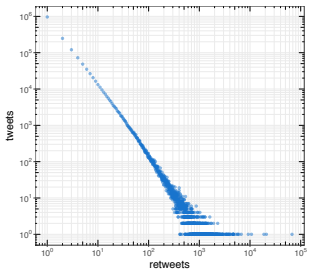


Figure 2: Total retweets per tweet.

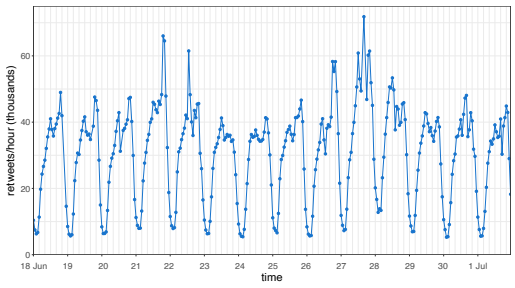


Figure 3: Hourly volume of retweets across the 2 considered weeks.

user in our dataset retweeted 7 times per day, in line with recent statistics reporting between 2 and 50 daily retweets for legitimate users [17]. However, our dataset also includes many "extreme" users, as visible in Figure 1 showing the distribution of retweets per day per user, which features a typical heavy-tailed shape. The 9,989,819 retweets are related to 1,691,865 distinct original tweets, which are also included in the dataset. Figure 2 shows the distribution of retweets per original tweet, while Figure 3 shows the hourly volume of retweets across the 2 considered weeks.

For each tweet, retweet, and user in our dataset we have access to all Twitter metadata fields, as provided by Twitter APIs. To collect this dataset, we resorted to Twitter Premium Search API[3] with the following query parameters: `lang:IT` and `is:retweet`. Here, the exploitation of the Premium Search API is important since it allowed us to build a *complete* dataset of retweets. In fact, the Standard Search API[4] used in the majority of previous works, does not guarantee completeness, meaning that not all tweets matching the query criteria are returned. Notably, although our dataset for this study is limited to tweets in the Italian language, both the data collection approach and the analytical process described in the remainder of the paper, are totally language independent. The language of collected tweets can be easily changed with the `lang:` parameter, and our analyses only exploit timestamps.

After data collection, we performed data filtering and annotation. A manual inspection of those users exhibiting the largest number of retweets per day quickly revealed their automated nature. However, it turned out that all such accounts, despite being bots, were not malicious ones. Their automated nature was manifest, they did not try to disguise as human-operated accounts, and they were not acting coordinatedly nor trying to inflate the popularity of some specific content. The presence of this kind of bots is well known [19]. They do not pose a threat to OSNs and, in fact, some of them are even benign [2, 39]. Thus, we excluded such accounts from future analyses, since they are not the target of our work. Similarly, we also excluded accounts featuring a very small number of retweets. Operationally, we set our filtering thresholds by leveraging statistics in [17] – that is, we retained only those users with a mean number of retweets per day ≥ 2 and ≤ 50. In this way, we ended up with 63,762 distinct users exhibiting human-like retweeting behaviors. The goal of our next analyses is to tell apart the sophisticated human-like bots from the real human-operated accounts.

Although our detection technique is unsupervised and hence it does not require a labeled ground truth, we nonetheless carried out manual annotation of a small subset of our dataset. This is useful in order to evaluate the extent to which our technique is capable of correctly spotting the bots. We thus annotated $\simeq 1,000$ accounts from our dataset, following the latest annotation guidelines for datasets containing social bots [8]. We ended up with an almost balanced annotated dataset[5], comprising 51% bots and 49% legitimate (i.e., human-operated) accounts.

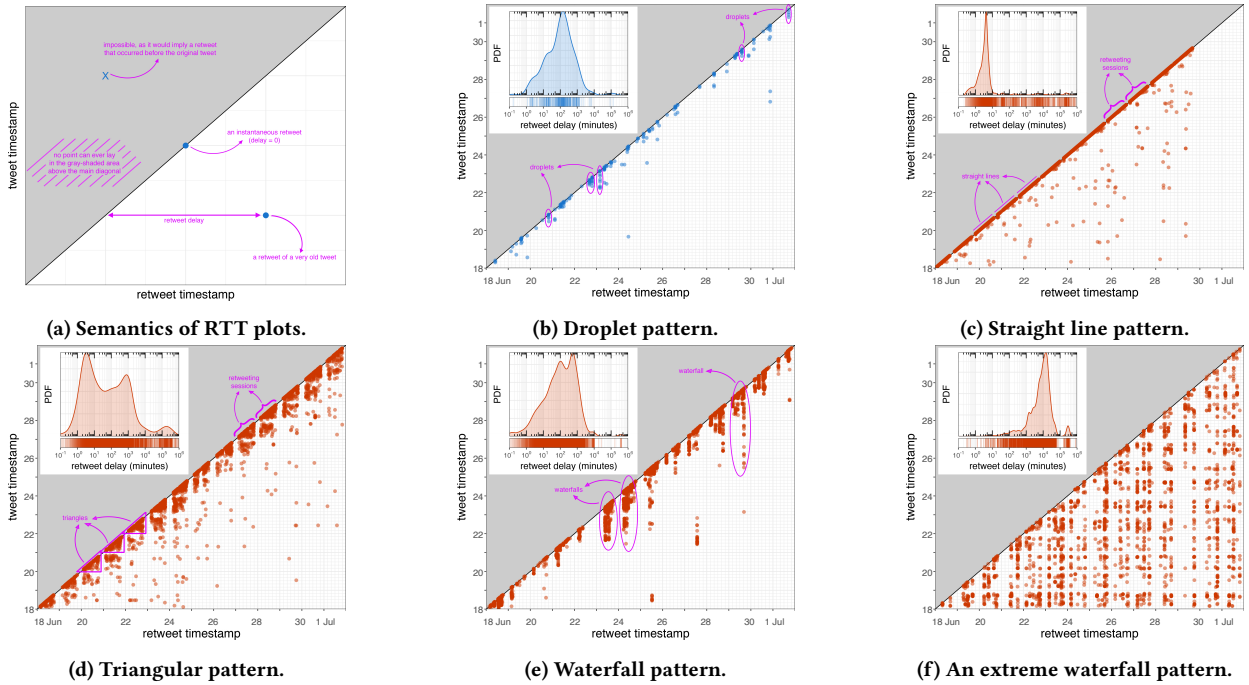
4 PATTERNS OF RETWEETING ACTIVITY

Here, we investigate the temporal dynamics of retweeting activity of all the 47,947 accounts in our dataset. By also leveraging class labels of the annotated accounts, we aim at highlighting retweeting behaviors that are indicative of normal versus suspicious activity.

To ease the exploration of a user's retweeting activity, we propose a compact – yet informative – scatterplot visualization called ReTweet-Tweet (RTT). Given a user and the list of all his retweets, RTT plots the timestamp of each retweet (x axis) against the timestamp of the corresponding original tweet (y axis). By the definition of RTT, points laying in different areas of the plot imply different retweeting behaviors. Figure 4a shows the semantics of this visualization, by means of a visual explanation of the most common behaviors caught by the RTT plot. In detail, no point can ever appear above the main diagonal of the plot, since that would break causality (i.e., it would correspond to a retweet that anticipates the original tweet, which is clearly impossible). Points laying near to the main diagonal represent retweets occurring rapidly after the publication time of the original tweet. Conversely, points that lay far from the diagonal imply a large temporal distance between a tweet and its retweet (i.e., retweets of very old tweets), which is an uncommon behavior.

Normal behaviors – droplet pattern. Figure 4b shows the typical RTT plot of a "normal", legitimate user. As shown, the vast majority of points concentrates slightly below the main diagonal. This is the expected behavior on the past-paced Twitter OSN, since retweets typically occur with short delay (e.g., between 1 and 10 minutes) from the original tweets [20]. Occasionally, legitimate users retweet a sequence of tweets with increasing delays. In RTT

[3] https://developer.twitter.com/en/docs/tweets/search/api-reference/premium-search.html
[4] https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html
[5] The dataset is avaible at https://doi.org/10.5281/zenodo.2653137



(a) Semantics of RTT plots. (b) Droplet pattern. (c) Straight line pattern.

(d) Triangular pattern. (e) Waterfall pattern. (f) An extreme waterfall pattern.

Figure 4: RTT **plots depicting a normal tweeting behavior (blue colored) and many suspicious ones (red colored). The insets of** RTT **plots show the empirical probability density function (PDF) and a rug plot of retweet delays.**

plots, this retweeting behavior creates a few vertically-stacked points that we refer to as *droplets*. We found this pattern to be frequent among legitimate users. In fact, the droplet pattern occurs each time a user retweets a Twitter feed (e.g., search results) or a user timeline. Since feeds and timelines are rendered – both in the Web application and via APIs – in reverse chronological order, retweeting a sequence of such tweets results in increasing retweet delays, just because tweets retweeted last are actually the oldest ones. Apart from these observations, no other clear pattern emerges in the RTT plot of Figure 4b. The possible presence of distinctive patterns in RTT plots is particularly relevant since it implies some form of regularity in a user's retweeting activity. In turn, striking regularities are typically caused by automated actions – that is, they are representative of bot behaviors, as previous literature already highlighted [5, 9, 31].

Suspicious behaviors – straight line pattern. The RTT plot in Figure 4c shows different retweeting behaviors. Almost all points in the plot are laying precisely above the main diagonal, meaning that the user almost always retweets in a matter of a few seconds from the original tweets. The user's activity is also clearly split into different sessions, separated by small gaps denoting inactivity.

Suspicious behaviors – triangular pattern. Figure 4d shows yet another suspicious pattern. This time the activity sessions are very regular, with always roughly the same length and the same inactivity time that separates subsequent sessions. Moreover, sessions seem to create a peculiar *triangular* pattern below the diagonal of the RTT plot. This means that, within a given session, the user retweets past tweets only up to a fixed point in time that roughly corresponds to the starting time of the session.

Suspicious behaviors – waterfall pattern. Finally, Figure 4e shows the behavior of a user whose retweets go way back in time. In figure, this is represented by points forming solid vertical lines. These lines are significantly longer than those representing droplets in Figure 4b and are probably caused by systematic retweeting of a Twitter feed or timeline in reverse chronological order. Since many vertical lines are present in the RTT plots showing this retweeting behavior, we call this a *waterfall* pattern. Apart from the vertical lines, mild signs of retweeting sessions are also visible in the plot. Although the presence of the *waterfall* pattern shown in Figure 4e is already indicative of automated behaviors, for some users this behavior is truly extreme, as shown in Figure 4f.

As highlighted by the previous analyses, the RTT plot is useful for studying the retweeting behavior of a given user. It is a valuable tool *per se*, that can empower human analysts for getting insights into the behaviors of Twitter accounts. Moving forward, in the next section we build on the results of these analyses by designing a fully automatic technique for spotting malicious retweeting bots.

5 INTRODUCING RETWEET-BUSTER

As highlighted in the previous sections, unsupervised group-based approaches currently represent the most promising research direction for social bot detection [8, 44]. Thus, in this section we propose an unsupervised technique that leverages the temporal distribution of the retweets of large groups of users, in order to detect malicious retweeting bots. Since we aim to design an unsupervised technique, we can not simply learn a detector for spotting the suspicious patterns identified in Section 4. Instead, our goal

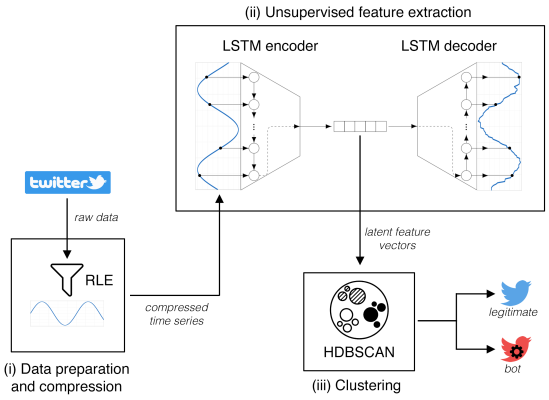


Figure 5: Main logical components of RTBust.

is to develop a technique that is capable of spotting such suspicious patterns, *as well as possible other patterns* that might exist in a dataset describing temporal retweeting activities. In other words, the technique should be able to automatically identify meaningful patterns from the data, rather than be able to recognize only those patterns that we manually labeled as suspicious.

Thus, building on these considerations, we propose a retweeting bot detection technique, called Retweet-Buster (RTBust), based on automatic unsupervised feature extraction. Then, in order to implement a group-analysis technique, such automatically-learned features are passed to a density-based clustering algorithm. The rationale for clustering stems from previous research in human and bot behaviors, in that humans have been proven to exhibit much more behavioral heterogeneity than automated accounts [5, 11]. As a consequence, we expect that the heterogeneous humans will not be sufficiently "dense" to be clustered. In fact, they will act pretty much like a background noise in the features space. Conversely, groups of coordinated and synchronized bots will be organized in much denser groups in the features space, thus resulting in nice density-based clusters. In detail, our proposed RTBust technique is organized in 3 main steps, as also shown in Figure 5: (i) time series data preparation and compression; (ii) unsupervised feature extraction; and (iii) user clustering.

Data preparation and compression. Let t_{REF} be a reference timestamp set at the start of the analysis time window. In our case t_{REF} corresponds to 17 June, 2018 at 00:00:00. Then, $t(x)$ denotes the publication timestamp of tweet ID = x. The raw data exploited to carry out bot detection with RTBust is the same used in the RTT plots of Section 4, that is tweet and retweet timestamps. In RTBust such timestamps are organized as a retweet time series $R_i = \{r_{i,0}, r_{i,1}, \ldots, r_{i,n}\}$ for each user u_i, where,

$$r_{i,j} = \begin{cases} |t(x) - t_{REF}| & \text{if } u_i \text{ retweeted tweet ID} = x \text{ at time } t_j \\ 0 & \text{if } u_i \text{ did not retweet at time } t_j \end{cases}$$

The temporal granularity of our time series is that of seconds, which is the same granularity as Twitter timestamps. That is, we have one $r_{i,j}$ observation per second per user. Hence, our time series have a very fine temporal grain. However, overall they are very sparse because users retweet, on average, only once in a few minutes.

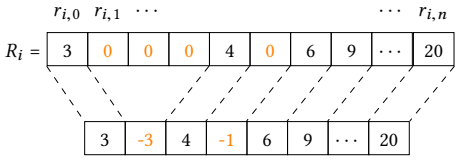


Figure 6: Excerpt of a retweet time series compressed with RLE. Observations affected by RLE are orange-colored.

At this step, we want to maximize the informativeness of our time series, while minimizing the amount of data to store and process. To beat this trade off, we employ a modified version of the run-length encoding (RLE) compression scheme. RLE is a simple and widely-used lossless sequence compression scheme that encodes consecutive equal data values with one single <*value, length*> tuple. In our case, we use RLE to reduce the sparsity in our time series by compressing the many consecutive zeros that represent no retweeting activity. Since we only need to compress zeros, we do not need to use the typical RLE <*value, length*> tuples. In fact, just the *length* will suffice. We have however to make a distinction between RLE lengths and the $|t(x) - t_{REF}|$ observations that represent retweets. Since by definition, $r_{i,j}$ observations that represent retweets are always positive, one straightforward way to make such distinction is to substitute consecutive zeros with the opposite of RLE lengths. Figure 6 graphically summarizes this data preparation and compression step. As a result of this step, we have one single compact time series for each user. Furthermore, such time series efficiently encodes multiple information, such as inactivity periods as well as tweets and retweets timestamps.

Unsupervised feature extraction. So far, we obtained a compact representation of the temporal retweeting behaviors of our users. Now, we need to turn this representation into a limited number of highly-informative features. Specifically, we have 3 goals for this step: (i) obtain features for a subsequent machine learning task (i.e., density-based clustering) in an unsupervised way; (ii) obtain fixed-length feature vectors, whereas user time series have variable length; (iii) maximize the amount of information in our features, while minimizing the number of features.

Basically, all our goals can be achieved with a well thought out application of dimensionality reduction techniques to our run-length-encoded time series. Specifically, we take a feature projection approach, where we aim to transform our compressed time series into lower-dimensional feature vectors. Feature projection can be achieved via a large number of different techniques. Here, we propose a solution based on *variational autoencoders*, a particular type of deep neural networks, because of their favorable characteristics [29]. In Section 6 we also show comparisons with other dimensionality reduction and feature projection techniques (e.g., PCA, TICA).

Variational autoencoders (VAEs) are an unsupervised learning technique that leverages neural networks for learning a probabilistic representation (i.e., learning features) of the input. As shown in Figure 5, a VAE is composed of 2 main modules: an encoder network and a decoder network. The encoder takes the original input and learns a compressed knowledge representation. Dually, the decoder

takes the compressed representation and tries to reconstruct the original input at its best. During the (unsupervised) training phase of the network, the encoder learns to create a meaningful and informative compressed representation of the input, so that the decoder can accurately reconstruct it. Notably, if the input dimensions are largely independent of one another, the compression performed by the encoder results in a loss of valuable information and the subsequent reconstruction becomes a very difficult task. However, if some sort of structure (i.e., patterns) exists in the input data, such as the peculiar patterns that we uncovered in Section 4, that structure can be learned by the encoder and consequently leveraged when converting the input into its compressed representation. In other words, this encoding learns latent features of the input data, which is precisely what we require from this unsupervised feature extraction step. Because deep neural networks are capable of learning nonlinear relationships, this encoding can be thought of as a more powerful generalization of PCA [35]. In our work, the decoder is only used for training the VAE, since we are not really interested in reconstructing our time series. Instead, once trained, the encoder becomes our unsupervised feature extractor. The dimension of the latent feature vector generated by the encoder is a parameter of the VAE, with which we extensively experiment in the next section. The lower the dimension, the more compressed is the representation of the input. However, a low-dimensional representation mitigates possible issues during the subsequent clustering step, caused by the curse of dimensionality [15].

Regarding the deep learning architecture used for implementing the VAE, we relied on a long short-term memory (LSTM) network. LSTMs are well-known and widely used recurrent neural networks. Because of their memory states, they are well-suited for performing tasks on time series, which are often characterized by temporal correlations and by lags of unknown duration between relevant events [23]. Specifically, LSTMs have been proven very accurate at extracting patterns in input space, where input data spans over long sequences. Furthermore, they are also suitable for dealing with sequences of variable length, such as our time series, while still being able to produce fixed-lengths data representations.

User clustering. Now that our users are represented by their latent feature vectors computed by the LSTM encoder, we can apply density-based clustering in order to check for common retweeting behaviors. If large clusters of users are found, then we might have detected a coordinated and synchronized group of accounts, possibly constituting a retweeting botnet. We base our clustering step on a recent efficient algorithm that combines density- and hierarchical-based clustering: HDBSCAN [4]. Among the advantages of this algorithm is its effectiveness in finding clusters with variable degrees of density, a much desirable feature when dealing with noisy real-world data. In addition, HDBSCAN also proved about twice as fast as its predecessor DBSCAN. Regarding algorithm parameters, HDBSCAN removed the need to specify a global density threshold (the ϵ parameter in DBSCAN) by employing an optimization strategy for finding the best cluster stability [4]. The only mandatory parameter to set is the minimum cardinality of the clusters found by HDBSCAN. We experiment with this parameter in the next section. Concluding, after running our density-based clusering step, we label as bots all those accounts that end up clustered. Conversely, we label as legitimate all those account that are treated as noise (i.e., that are not clustered) by HDBSCAN.

6 EXPERIMENTS AND RESULTS

Here we present results obtained while searching for the best set of parameters of RTbust, as well as overall bot detection results.

Evaluation methodology. In all the experiments described in this section we analyze all 63,762 accounts of our dataset. Then, we evaluate the effectiveness of different techniques, possibly executed with different configurations of parameters, in correctly classifying the $\simeq 1,000$ annotated accounts. Thus, the bot detection task is framed as a binary classification task, with the 2 classes being: *bot* (positive class) and *human* (negative class). For presenting evaluation results we rely on 5 standard, well-known metrics used for evaluating binary machine learning classifiers, specifically: precision, recall, accuracy, F1-Score ($F1$), and Matthews correlation coefficient (MCC) [37].

Comparisons. While evaluating the bot detection results of RTbust, we also perform extensive comparisons with other baselines and state-of-the-art techniques for social bot detection.

In Section 5 we grounded the unsupervised feature extraction step of RTbust on a variational autoencoder. However, a number of other techniques could be used to achieve the same goal. Thus, for the sake of experimentation we implemented 2 more versions of RTbust that are based respectively on principal component analysis (PCA) and time independent component analysis (TICA) for feature extraction [25]. PCA is a well-known linear statistical technique for dimensionality reduction. By choosing an appropriate number of principal components found by PCA, it is possible to greatly reduce the dimensionality of a problem, while minimizing the loss of information. Similarly, TICA is another dimensionality reduction technique that is particularly suitable for dealing with time series. While PCA finds high-variance linear combinations of the input dimensions, TICA aims to find high-autocorrelation linear combinations. Thus, when using PCA and TICA for feature extraction, the features resulting from PCA are likely to convey different information with respect to those obtained from TICA.

As typically done for many machine learning tasks, we also experiment with a small set of handcrafted features. These features take into account characteristics of the retweet time series of the users (e.g., inter-retweet times, retweeting delays and sessions), as well as other characteristics not available from retweet time series, but that might still contribute to bot detection (e.g., the distribution of retweeted accounts). Table 1 lists the 12 handcrafted features. Notably, some of them – such as inter-retweet times – have been largely used in previous retweeter bot detection systems [24, 43]. In our subsequent experiments, these handcrafted features are tested in RTbust in place of those automatically extracted by PCA, TICA, and the VAE.

Regarding comparisons with other state-of-the-art bot detection techniques, we experiment with the supervised version of the *Social fingerprinting* technique proposed in [7, 9] and with the unsupervised *HoloScope* technique proposed in [31]. Both techniques are designed to detect synchronized and coordinated groups of bots, and have been briefly described in Section 2. In addition, we also

	feature	description
1	RT users entropy	Shannon entropy of the distribution of retweeted users
2	RT days entropy	Shannon entropy of the distribution of the publication days of retweeted tweets
3	RT rate	number of retweets per time unit
4	daily mean RTs	daily mean of the number of retweets
5	RT days	number of distinct days during which the user retweeted at least once
6	minimum IRT	minimum of the inter-retweet times
7	mean IRT	mean of the inter-retweet times
8	stdev IRT	standard deviation of the inter-retweet times
9	minimum RT delay	minimum of the retweet delays
10	mean RT delay	mean of the retweet delays
11	stdev RT delay	standard deviation of the retweet delays
12	RT sessions	number of detected retweeting sessions

Table 1: List of handcrafted features computed for each user.

Figure 7: Dependence of clustering stability and bot detection performances on the number of features extracted by PCA, TICA and the VAE. Best results are obtained with the VAE, starting from 8 features (dashed vertical line).

experiment with the *Botometer* system [12]. *Botometer* is a publicly-available service[6] to evaluate the similarity of a Twitter account with the known characteristics of social bots. It leverages an off-the-shelf supervised machine learning classifier that exploits more than 1,000 features of the accounts under investigation. Similarly to the majority of existing systems, *Botometer* performs account-by-account analyses, rather than group analyses.

Finally, as a simple comparison baseline we also try classifying as bots all those accounts whose retweet rate (i.e., retweets per unit of time) is higher than a fixed threshold. We set this threshold as the third quartile of the distribution of retweet rates for our 63,762 accounts. This comparison helps understand if simply looking at the number of retweets is enough for gaining insights into the nature (bot/legitimate) of an account.

Parameters configuration. The most important parameter of RTbust is the dimension of the latent feature vectors generated by the VAE. It affects both the quality of the subsequent clustering step, which in turn affects bot detection performances, as well as the time/computation needed to complete it. The same also applies to the number of projected dimensions obtained with PCA or TICA. Thus, we designed an experiment to evaluate the performances of RTbust, implemented with VAE, PCA, and TICA for feature extraction, in relation to the number of considered latent features.

Figure 7 shows results for this experiment, in terms of the overall $F1$ obtained for the classification of bots and humans, and of the total number of clusters found by HDBSCAN. Regarding the number of clusters, all 3 implementations show the same qualitative behavior. When considering a very limited number of features, HDBSCAN finds a large number of clusters. Increasing the number of features results in fewer clusters, up to a point (size of latent feature vector = 10) where the number of clusters stops decreasing. Regarding bot detection performance, Figure 7 shows a very different behavior between the VAE implementation of RTbust, with respect to the PCA and TICA ones. Features extracted by PCA and TICA seem to provide significantly less information for bot detection with respect to those extracted by the VAE, as demonstrated by a lower $F1$. Moreover, increasing the number of considered PCA and TICA features seems of no help, since the $F1$ remains almost the same for each latent feature vector size. Instead, the VAE implementation shows a behavior that is consistent with the results obtained for the number of clusters. The $F1$ rapidly increases when considering a larger number of features, up to a point (size of latent feature vector = 8) where it reaches and maintains its maximum (apart from a few fluctuations). Overall, these results demonstrate that the VAE is much more powerful than PCA and TICA for the unsupervised feature extraction from our retweet time series. Furthermore, as little as 8 VAE features seem to be enough for stabilizing the clustering and achieving good bot detection results.

We repeated this experiment by also varying the minimum cluster cardinality (i.e., the only HDBSCAN parameter). We omit detailed results due to space limitations, however we report finding the best results with clusters larger than 10 accounts, a threshold that is both intuitive (i.e., it is not very meaningful to look for minuscule botnets) and operationally effective.

Quantitative evaluation of bot detection. Next, we present detailed bot detection results for all the considered techniques. For the VAE, PCA and TICA implementations of RTbust, we use only 8 features, leveraging results of our previous experiment.

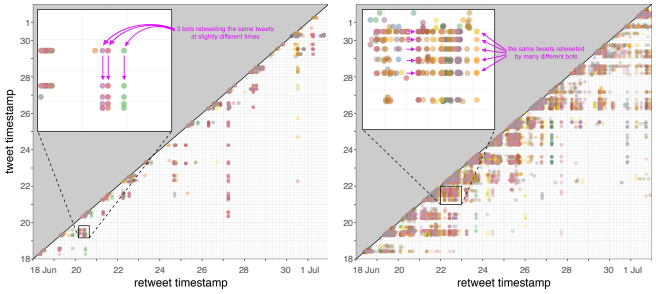
Table 2 shows retweeter bot detection results. The best detection performances ($F1 = 0.87$) are achieved by the proposed RTbust technique using the VAE for unsupervised feature extraction. In fact, it beats all other competitors in each evaluation metric, with the only exception of the *recall* metric. All other implementations of RTbust achieve worse results, with $F1 \leq 0.67$. As anticipated, this is a strong point in favor of the VAE for extracting informative features from our retweet time series. The second best overall results are obtained by the *Social fingerprinting*, achieving an encouraging $F1 = 0.76$. Instead, the other state-of-the-art techniques obtain much worse results, with *Botometer*'s $F1 = 0.43$ and *HoloScope*'s $F1 = 0.01$. Interestingly, the majority of evaluated techniques achieve their worst results in the *precision* metric, meaning that many legitimate accounts are misclassified as bots. This results is in contrast with previous results in bot detection [8]. However, previous works mainly experimented with supervised techniques, while here we mainly explore unsupervised ones. Thus, combined

[6] https://botometer.iuni.iu.edu

technique	type	features	evaluation metrics				
			precision	*recall*	*accuracy*	*F1*	*MCC*
baseline							
retweet rate	–	1	0.3534	0.3585	0.3440	0.3559	−0.3124
comparisons							
Botometer [12]	supervised	> 1, 000	0.6951	0.3098	0.5830	0.4286	0.2051
HoloScope [31]	unsupervised	–	0.2857	0.0049	0.4908	0.0096	−0.0410
Social fingerprinting [7, 9]	supervised	–	0.6562	0.8978	0.7114	0.7582	0.4536
our contributions							
RTbust (handcrafted features)	unsupervised	12	0.5284	0.7707	0.5364	0.6270	0.0767
RTbust (PCA)	unsupervised	8	0.5111	**0.9512**	0.5154	0.6649	0.0446
RTbust (TICA)	unsupervised	8	0.5228	**0.9512**	0.5364	0.6747	0.1168
RTbust (VAE)	unsupervised	8	**0.9304**	0.8146	**0.8755**	**0.8687**	**0.7572**

Table 2: Retweeter bot detection results of RTbust and comparison with a baseline and other state-of-the-art techniques. Best results in each evaluation metric are shown in bold.

Figure 8: Visual exploration of RTbust (VAE) bot detection results, and comparison with t-SNE.



(a) *Cars* botnet. **(b) *Singer* botnet.**

Figure 9: RTT plots of the 2 botnets discovered with RTbust.

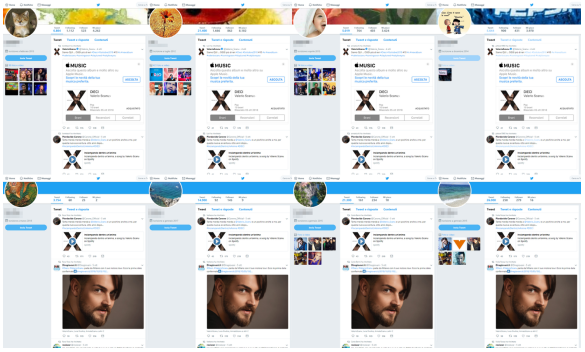


Figure 10: A subset of accounts from the *singer* botnet.

results of our study and previous ones might suggest that supervised approaches to social bot detection are more prone to type II errors (false negatives), while unsupervised approaches are more prone to type I errors (false positives).

In Figure 8, we use t-SNE [33] for plotting our annotated accounts in a bi-dimensional space, where each account is colored according to the RTbust (VAE) cluster it belongs to, and 2 different symbols represent bots and legitimate accounts according to our ground truth. Since in RTbust each clustered account is labeled as bot while unclustered ones are labeled as humans, by comparing colors and symbols of the accounts in figure, it is possible to visually assess the quality of bot classification. Furthermore, Figure 8 also allows a comparison between our clustering and that resulting from t-SNE. Regarding bot detection, we can see few clustered legitimate accounts. Similarly, only a minority of bots are grey-colored (i.e., unclustered). Concerning the comparison between t-SNE and RTbust, the 2 algorithms produce rather different clusters. For instance, multiple RTbust clusters collapse in a single t-SNE cluster and vice versa. However, the overall distinction between bots and humans seem to hold for both RTbust and t-SNE. The majority of legitimate accounts are both grey-colored and spread across the central region of Figure 8. In other words, they do not belong to any cluster neither for t-SNE nor for RTbust. Dually, the majority of bots appear as colored and positioned near other bots. Thus, both t-SNE and RTbust seem to recognize bot similarities although organizing the bots differently, despite the very different inner functioning of the 2 techniques. Given the usefulness and widespread application of t-SNE, these similarities support results of RTbust (VAE).

Qualitative evaluation of bot detection. Given the good results achieved by RTbust (VAE) at spotting the annotated bots in

our dataset, we now analyze also those non-annotated accounts that have been labeled as bots by the same technique. This manual and qualitative validation might serve as an additional evaluation of the outcomes of RTbust.

In particular, RTbust found 2 notable clusters of non-annotated accounts. The smallest of such clusters counts 44 accounts. A manual inspection of these accounts revealed that they actually belong to a small botnet with some peculiar characteristics. First and foremost, they almost only retweet 3 accounts (*@peugeotitalia*, *@citroenitalia* and *@motorionline*). Thus, this botnet has a specific focus on *cars*. Moreover, they only post retweets or images, no account of the botnet has a real profile picture, and they are loosely synchronized, meaning that they tend to retweet the same tweets but at different moments in time. In order to get further insights into their behavior, we produced a combined RTT visualization of all the accounts of the botnet, as shown in Figure 9a. Each point in figure is a retweet made by one of the botnet's account and colors represent different bots. Interestingly, clear patterns emerge in some portions of the RTT plot, as seen in the inset of Figure 9a. Such patterns testify the synchronized retweeting activity of the botnet.

The other notable cluster of bot accounts found by RTbust is composed of almost 300 accounts that almost exclusively retweet 2 accounts (*@Valerio_Scanu* and *@ArmataScanu*). Similarly to the previous smaller botnet, also this larger one seems to retweet with a specific focus – that is, publicizing tweets related to the Italian pop singer *Valerio Scanu*[7]. Figure 9b shows the RTT plot of this botnet, which exhibits a mixture of the triangular and waterfall patterns that we described in Section 4. Figure 10 also displays the appearance of the profiles of some of these accounts. As seen, some accounts of the botnet are tightly synchronized, while the botnet as a whole is loosely synchronized.

This qualitative evaluation revealed that, not only did RTbust achieve good results on the annotated accounts, but it also allowed to discover 2 previously unknown active retweeting botnets.

7 DISCUSSION

Visualizing suspicious behaviors. In this work we provided several contributions. The first of such contributions is the development of the RTT visualization. In figures 4 and 9 we showed that RTT plots represent a useful tool for analyzing the retweeting behavior of an account, or of a group of accounts. Human analysts can gain valuable insights into the behaviors of suspicious accounts by leveraging RTT plots. The possibility to leverage RTT plots might be particularly valuable in future data annotation tasks, considering the current difficulties faced by human annotators in correctly labeling social bots [8]. Another favorable application scenario is related to the banning of bots from OSNs. Indeed, despite the recent advances in machine learning-based detectors, manual verification of accounts to assess their degree of automation is still carried out by OSN administrators [16]. To this end, RTT plots might contribute to speed-up the process and to reduce possible human mistakes.

Notably, a few previous works also proposed some simple visualizations with the goal of highlighting suspicious behaviors in OSNs [18, 26, 27]. However, previous visualizations for spotting suspicious behaviors require much more data than that needed by RTT plots. For example, the visualizations used in [26] are based on the full social graph of considered accounts. Similarly, the visualizations proposed in [18] require building retweet threads/cascades, which in turn require friend/followers information of each retweeter. Conversely, with RTT plots we can spot suspicious retweeting behaviors with as little information as retweet-tweet timestamps.

Generalizability and robustness. Our second, and largest, contribution is the development of the Retweet-Buster (RTbust) bot detection technique. RTbust is an unsupervised technique that is capable of automatically spotting meaningful patterns in retweet data. Because of this feature, RTbust is potentially capable of detecting retweeting bots exhibiting a behavior that has not been witnessed before. We believe this to be an important feature, considering that we still lack a "standard" and "well-agreed" definition of what a social bot is [34, 44] – and consequently, of its expected behaviors [1]. Hence, the capability to spot a broad set of different behaviors, surely comes in handy. Moreover, it has been now largely demonstrated that social bots do *evolve* to escape detection techniques [8, 42]. Thus, the generalizability of RTbust and its robustness to evasion is a much desirable feature because it allows us to better withstand the next evolution of social bots.

Explainability. Model and decision explainability has now become one of the major practical and ethical concerns around AI [22]. To this regard, one of the possible drawbacks of RTbust lies in the difficulty to "explain" the reasons for labeling an account as bot or legitimate. This is largely due to the difficulty in interpreting the latent features used for clustering, which in turn depend on the adoption of the "black-box" variational autoencoder for unsupervised feature extraction. This issue is not peculiar of our technique, but it is rather a well-known limitation of all deep learning techniques [22]. Here, to mitigate this issue we again propose to resort to RTT plots. As demonstrated in Figure 9, our visualization can be profitably used also *after* the application of RTbust, with the goal of understanding the characteristics of those accounts that have been labeled as bots.

8 CONCLUSIONS

In this work, we investigated patterns of retweeting activity on Twitter, with the specific goal of detecting malicious retweeting bots. To this end, our work provides several contributions.

Firstly, we proposed a novel visualization technique called ReTweet-Tweet (RTT). As thoroughly shown, RTT plots are an effective and efficient mean to gain valuable insights into the retweeting behaviors of Twitter accounts. By leveraging RTT plots we analyzed the "normal" retweeting behavior of legitimate users, and we uncovered 3 suspicious behaviors that are caused by automated retweeting – and thus, that are representative of bot activities. Furthermore, we discussed how RTT plots can empower human analysts when manually annotating a dataset comprising social bots, as well as OSN administrators looking for evidence of automation when deciding about banning accounts from social platforms. Finally, we highlighted that RTT plots can also be used to explain decisions of black-box bot detectors, thus contributing towards explainable and interpretable AI.

Next, we designed an unsupervised group-analysis technique, called Retweet-Buster (RTbust), for detecting retweeting social

[7] https://en.wikipedia.org/wiki/Valerio_Scanu

bots. The core of RTbust is an LSTM variational autoencoder that we trained to extract a minimum number of highly informative latent features from the retweet time series of each account. The decision about an account (whether it is bot or legitimate) is based on the outcome of a hierarchical density-based clustering algorithm. In detail, accounts belonging to large clusters are labeled as bots, while unclustered accounts are labeled as legitimate. We compared different implementations of RTbust with baselines and state-of-the-art social bot detection techniques. RTbust outperformed all competitors achieving $F1 = 0.87$, in contrast with $F1 \leq 0.76$ of other techniques. By applying RTbust to a large dataset of retweets, we also discovered 2 previously unknown active botnets comprising hundreds of accounts.

For future work we plan to improve the decisions taken by RTbust. In fact, here we adopted a rather naive solution revolving around accounts being clustered or not. However, we can augment the final decision step of RTbust by considering additional information, such as the hierarchy of clusters produced by the clustering algorithm and other internal and external clustering validation measures. In this way, we could prune some clusters or some accounts belonging to a cluster, thus improving *precision*. Similarly, we could expand some clusters by adding borderline accounts, thus possibly also improving *recall*, which currently is the bottleneck of RTbust. Other than improving RTbust, we also plan to apply it *in the wild* for discovering and analyzing active botnets.

REFERENCES

[1] Abdullah Almaatouq, Ahmad Alabdulkareem, Mariam Nouh, Erez Shmueli, Mansour Alsaleh, Vivek K Singh, Abdulrahman Alarifi, Anas Alfaris, and Alex Sandy Pentland. 2014. Twitter: who gets caught? observed trends in social micro-blogging spam. In *ACM WebSci*.

[2] Marco Avvenuti, Salvatore Bellomo, Stefano Cresci, Mariantonietta Noemi La Polla, and Maurizio Tesconi. 2017. Hybrid crowdsensing: A novel paradigm to combine the strengths of opportunistic and participatory crowdsensing. In *ACM WWW Companion*.

[3] Alessandro Bessi, Mauro Coletto, George Alexandru Davidescu, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. 2015. Science vs conspiracy: Collective narratives in the age of misinformation. *PloS one* 10, 2 (2015).

[4] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *PAKDD*.

[5] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. 2016. DeBot: Twitter Bot Detection via Warped Correlation. In *IEEE ICDM*.

[6] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems* 80 (2015).

[7] 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).

[8] 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 *ACM WWW Companion*.

[9] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2018. Social Fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE TDSC* 15, 4 (2018).

[10] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. 2019. On the capability of evolved spambots to evade detection via genetic engineering. *Online Social Networks and Media* 9 (2019).

[11] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. Exploiting digital DNA for the analysis of similarities in Twitter behaviours. In *IEEE DSAA*.

[12] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. BotOrNot: A System to Evaluate Social Bots. In *ACM WWW Companion*.

[13] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *PNAS* 113, 3 (2016).

[14] Michela Del Vicario, Gianna Vivaldo, Alessandro Bessi, Fabiana Zollo, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. 2016. Echo chambers: Emotional contagion and group polarization on facebook. *Scientific reports* 6 (2016).

[15] Pedro Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10 (2012).

[16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM* 59, 7 (2016).

[17] Syeda Nadia Firdaus, Chen Ding, and Alireza Sadeghian. 2018. Retweet: A popular information diffusion mechanism–A survey paper. *Online Social Networks and Media* 6 (2018).

[18] Maria Giatsoglou, Despoina Chatzakou, Neil Shah, Christos Faloutsos, and Athena Vakali. 2015. Retweeting activity on twitter: Signs of deception. In *PAKDD*.

[19] Zafar Gilani, Ekaterina Kochmar, and Jon Crowcroft. 2017. Classification of twitter accounts into automated agents and human users. In *IEEE/ACM ASONAM*.

[20] Manuel Gomez-Rodriguez, Krishna P Gummadi, and Bernhard Schoelkopf. 2014. Quantifying Information Overload in Social Media and Its Impact on Social Contagions. In *AAAI ICWSM*.

[21] Christian Grimme, Dennis Assenmacher, and Lena Adam. 2018. Changing Perspectives: Is It Sufficient to Detect Social Bots?. In *SCSM*.

[22] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM CSUR* 51, 5 (2018).

[23] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. 2016. Robust online time series prediction with recurrent neural networks. In *IEEE DSAA*.

[24] Sonu Gupta, Ponnurangam Kumaraguru, and Tanmoy Chakraborty. 2019. MalReG: Detecting and Analyzing Malicious Retweeter Groups. In *ACM CoDS-COMAD*.

[25] Carlos X Hernández, Hannah K Wayment-Steele, Mohammad M Sultan, Brooke E Husic, and Vijay S Pande. 2018. Variational encoding of complex dynamics. *Physical Review E* 97, 6 (2018).

[26] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Catching Synchronized Behaviors in Large Networks: A Graph Mining Approach. *ACM TKDD* 10, 4 (2016).

[27] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring lockstep behavior from connectivity pattern in large graphs. *KAIS* 48, 2 (2016).

[28] Christian Kater and Robert Jäschke. 2016. You shall not pass: detecting malicious users at registration time. In *ACM WebSci Workshops*.

[29] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *IEEE ICML*.

[30] Sangho Lee and Jong Kim. 2014. Early filtering of ephemeral malicious accounts on Twitter. *Computer Communications* 54 (2014).

[31] Shenghua Liu, Bryan Hooi, and Christos Faloutsos. 2017. HoloScope: Topology-and-Spike Aware Fraud Detection. In *ACM CIKM*.

[32] Shenghua Liu, Bryan Hooi, and Christos Faloutsos. 2018. A Contrast Metric for Fraud Detection in Rich Graphs. *IEEE TKDE* (2018).

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* (2008).

[34] Gregory Maus. 2017. A Typology of Socialbots (Abbrev.). In *ACM WebSci*.

[35] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J Kennedy. 2017. Relational autoencoder for feature extraction. In *IEEE IJCNN*.

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

[37] David Martin Ward Powers. 2011. Evaluation: from Precision, Recall and F-Measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technologies* 2, 1 (2011).

[38] Walter Quattrociocchi. 2017. Inside the echo chamber. *Scientific American* 316, 4 (2017).

[39] Saiph Savage, Andres Monroy-Hernandez, and Tobias Höllerer. 2016. Botivist: Calling volunteers to action using online bots. In *ACM CSCW*.

[40] Massimo Stella, Emilio Ferrara, and Manlio De Domenico. 2018. Bots increase exposure to negative and inflammatory content in online social systems. *PNAS* 115, 49 (2018).

[41] L Steward, Ahmer Arif, and Kate Starbird. 2018. Examining Trolls and Polarization with a Retweet Network. In *ACM WSDM Workshops*.

[42] Stefano Tognazzi, Stefano Cresci, Marinella Petrocchi, and Angelo Spognardi. 2018. From Reaction to Proaction: Unexplored Ways to the Detection of Evolving Spambots. In *ACM WWW Companion*.

[43] Nguyen Vo, Kyumin Lee, Cheng Cao, Thanh Tran, and Hongkyu Choi. 2017. Revealing and detecting malicious retweeter groups. In *IEEE/ACM ASONAM*.

[44] Kai-Cheng Yang, Onur Varol, Clayton A Davis, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2019. Arming the public with AI to counter social bots. *Human Behavior and Emerging Technologies* (2019).

[45] Fabiana Zollo, Alessandro Bessi, Michela Del Vicario, Antonio Scala, Guido Caldarelli, Louis Shekhtman, Shlomo Havlin, and Walter Quattrociocchi. 2017. Debunking in a world of tribes. *PloS one* 12, 7 (2017).