Its All in a Name: Detecting and Labeling Bots by Their Name

2 authors:

David Beskow
Carnegie Mellon University
21 PUBLICATIONS **33** CITATIONS

SEE PROFILE

Kathleen M Carley
Carnegie Mellon University
686 PUBLICATIONS **18,395** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Socio-Cultural Cognitive Mapping - A Latent Space Approach View project

Network Flow View project

Its All in a Name: Detecting and Labeling Bots by Their Name

David M. Beskow and Kathleen M. Carley

School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213, USA
Email: dbeskow@andrew.cmu.edu and kathleen.carley@cs.cmu.edu

Abstract. Automated social media *bots* have existed almost as long as the social media environments they inhabit. Their emergence has triggered numerous research efforts to develop increasingly sophisticated means to detect these accounts. These efforts have resulted in a *cat and mouse* cycle in which detection algorithms evolve trying to keep up with ever evolving *bots*. As part of this continued evolution, our research proposes a multi-model 'tool-box' approach in order to conduct detection at various *tiers* of data granularity. To support this toolbox approach this research also uses random string detection applied to user names to filter twitter streams for bot accounts and use this as labeled training data for follow on research.

1 Introduction

Automated social media accounts, often called "bots", are increasingly used on many social media sites. Ever since social media sites built Application Programming Interfaces (API's) that allow their platforms to integrate with other platforms and applications, various actors have developed computer routines that conduct a variety of automated tasks on the respective social media ecosystems. While some bots are designed for positive purposes [14], many others range from nuisance (i.e. a spam bot) to propaganda [19], suppression of dissent [26], and network infiltration/manipulation [12,3] . They have recently gained widespread notoriety due to their use in several major international events, including the British Referendum known as "Brexit" [15], the American 2016 Presidential Elections [7], the aftermath of the 2017 Charlottesville protests [13], the German Presidential Elections [21], the conflict in Yemen [18], and recently in the Malaysian presidential elections [2].

As these bots have proliferated and their use is being discussed broadly in the media and political bodies, researchers have increasingly developed methods to detect these accounts. The same openness and ease of use of the social media API's that facilitates the creation and use of automated accounts also facilitates the collection of data used to detect them. As detection efforts proliferate, bot engineers change and adapt in order to survive and succeed in a dynamic environment. The requirement for higher accuracy in the midst of a changing *signal*

motivates our efforts to improve not only the models that detect bots, but the labeled data that is used to train them.

This paper lays the foundation for a *tiered* supervised machine learning approach to bot detection and characterization. This approach acknowledges that there are discrete levels or 'tiers' of data granularity, and seeks to develop supervised machine learning models for each *tier* of data on social media platforms. The resulting bot detection 'toolbox' ensures that researchers have requisite models for their specific data granularity. Some research aims at understanding bot behavior in large conversations (analyze overall bot presence in the Twitter conversation surrounding the 2018 mid-term elections), while other research aims at characterizing a handful of accounts (an depth analysis of the top 10 most influential followers of the NATO Twitter account). A toolbox approach provides distinct models to support both requirements, allowing researchers to analyze the proverbial *forest* with one model and *trees* with a distinct but related model.

Table 1: Four *tiers* of Twitter data collection to support account classification

Tier	Description	Focus	Collect/process Time per 250 Accounts	# of Data Entities (i.e. tweets)
Tier 0	Tweet text or user name	Semantics	N/A**	1
Tier 1	*User* Object + 1 Tweet object	Account Meta-data	∼ 1.9 seconds	2
Tier 2	*User* Object + Timeline	Temporal patterns	∼ 3.7 minutes	200+
Tier 3	*User* Object + Timeline + Friends Timeline	Network patterns	∼ 20 hours	50,000+

To support the development of this toolbox, our research has identified several tiers of Twitter data collection and developed related machine learning feature space and models. The constraints of data availability and rate limiting associated with the Twitter API [1] artificially create these tiers, which are summarized in Table 1. *Tier 0* involves just a single entity, most often either a single *status* or *screen name*. *Tier 1* is the *tweet* object (and associated *user* object), and is the most common data granularity collected by researchers. *Tier 2* adds a user *timeline* object (up to last 3,200 tweets) for every account, and *Tier 3* adds the larger conversation that an account interacts with (i.e. an ego-network conversation).

The data, feature space, and models associated with higher *tiers* are rich and provide higher accuracy, but computationally expensive, as indicated in Table 1. *Tier 3* models can take over 20 hours to process 250 accounts. Currently, researchers who use Tier 2 models on large datasets are forced to sample their data and assume that their sample is representative of overall bot distribution

and characteristics. By providing models at Tier 0 and Tier 1, our toolbox will allow these researchers to conduct bot detection on 100% of their data instead of sampling.

While numerous research efforts have attempted to exploit pieces and parts of this data spectrum, few have attempted to create a comprehensive approach that covers all tiers. The closest effort that we've seen is the Botometer effort discussed later in this paper. While offering an robust model through an accessible API, it is only offered at Tier 2, meaning high volume classification is computationally expensive. Additionally, if does not exploit the rich network features available at Tier 3.

This paper seeks to lay the groundwork for a toolbox approach to bot detection, discuss screen name focused data annotation, as well as build and evaluate a Tier 0 model. See [5] for discussion of our Tier 1 model and [4] for initial efforts to develop baseline Tier2 and Tier 3 models.

Our work therefore makes three primary contributions to the literature. First, we propose a novel random string detection model that is specifically designed to detect 15 character randomly generated strings. When applied to the *screen name* field of Twitter data, this technique is able to easily filter accounts that are likely bot accounts. Second, by applying this filtering technique to a large sample collected from the Twitter Streaming API, we have produced a large and diverse annotated data set for use in training more robust specialized and general purpose bot detection models. Finally, this paper lays the foundation for *bot-hunter*, a multi-model 'toolbox' approach to bot detection.

This paper begins with a brief description of the background of general bot detection, as well as past efforts perform random string detection. We will then describe the models and algorithms that we developed for random string classification, as well as methods that we used to evaluate them on the narrow tasks that they were created for. Finally, we describe how we've applied this algorithm to create a large and diverse annotated Twitter bot data set for use by the research community.

2 Related Work

2.1 Twitter bot detection

Although early work on classifying Twitter accounts dates back to as early as 2008 [16], the deliberate detection of automated accounts on the Twitter Platform began in earnest in 2010 when [8] conducted three-class classification (human, bot, cyborg) using an ensemble model. In 2011, a team from Texas A&M became the first to use *honey pots* to detect thousands of bots [17]. These *honey pots* used bots that generate nonsensical content, designed only to attract other bots. The Texas A&M bots attracted thousands of bots, and generated a labeled data set that has been used on many later research efforts. This *honey pot* method was repeated by others to create similar data sets in other parts of the world [25].

In 2014, Indiana University and the University of Southern California launched the *Bot or Not* online API service [9]. This used traditional classification models trained on the Texas A&M dataset to help users evaluate whether or not an account is a bot. *Bot or Not* leverages *network*, *user*, *friend*, *temporal*, *content*, and *sentiment* features with Random Forest classification.

In 2015 the Defense Advanced Research Projects Agency (DARPA) sponsored a Twitter bot detection competition that was titled "The Twitter Bot Challenge" [25]. This four week competition pitted four teams against each other as they sought to identify automated accounts that had infiltrated the informal Anti-Vaccine network on Twitter. Most teams in the competition tried to use previously collected data (mostly collected and tagged with *honey pots*) to train detection algorithms, and then leverage tweet semantics (sentiment, topic analysis, punctuation analysis, URL analysis), temporal features, profile features, and some network features to create a feature space for classification. All teams used various techniques to identify initial bots, and then used traditional classification models (SVM and others) to find the rest of the bots in the data set.

Most recently, the team from Indiana University re-branded *Bot-or-Not* to *Botometer*, increasing the set of features to 1,150 account related features [10]. Their team compared Random Forests, AdaBoost, Logistic Regression and Decision Tree classifiers and still found that Random Forests performed best. They also attempted to update their training data by manually annotating tweet accounts, and merging this with the original Texas A&M Dataset (collected in 2011).

The continued use of the 2011 Texas A&M data highlights the difficulty that researchers have in creating and/or updating the labeled data that is used train algorithms to find these automated accounts. The use of aging training data for bot classification also ensures that emerging bots are likely to avoid detection. Additionally, since bots have a variety of purposes as well as a spectrum of actors that create/use them, the collection technique used for labeled data will bias the detection toward that family of bots. For example, the *honey pot* collection technique will bias toward bots that randomly follow accounts, but may not detect intimidation bots that conduct targeted following and messaging.

2.2   Classifying algorithmic character strings

Classifying strings as *random* or *not random* in order to filter or flag anomalous events has a limited background.

Several methods have been proposed for identifying or highlighting the randomness of character strings. Some have proposed leveraging Shannon's Entropy calculation [24] as a method for sorting strings by a measure of randomness. Some cyber security research teams have proposed a similar detection methods in order to detect domain names that are generated by Domain Generation Algorithms (DGA). These teams have separately used Kullback-Leibler Divergence [27], a dictionary approach [20] and Markov modeling [23].

The past research most closely connected to our effort was conducted by LinkedIn in 2013. At that time [11] presented the application of the Naive Bayes

model on Character N-grams features of LinkedIn account names in order to identify *spammy* accounts (first and last name as provided by the account owner). This effort was very effective, and replaced the legacy spam detection models that LinkedIn was using on their OSN. To date, our team has not found any team that has replicated a similar approach to Twitter screen names.

2.3   Project background

Our team has focused on detecting, characterizing, and modeling the behavior of bots, bot networks and their creators. In doing this we've studied several recorded bot events. Recently we focused on a known and publicized bot attack against the Atlantic Council Digital Forensic Labs (DFR Lab), and tangentially against the NATO Public Affairs Office. This attack primarily occurred between August 28 and August 30, 2017. We also focused on a recorded bot harassment event against journalists in Yemen [18]. In both events we observed numerous bot accounts that used 15 character randomly generated alpha-numeric strings for the screen name. Examples of this include **Wy3wU4HegLlvHgC**, **5JSQavWW3tvQwA7**, and **gG6RKc6QBqOLKyU** (these are not real Twitter accounts). Note that these randomly generated strings always sample from upper and lower case alpha-numeric characters. Observing this phenomenon motivated the construction of this algorithm and its application on Twitter at large in order to observe other bots and bot actors that are using these same type of bot screen names. More importantly, we hope this dataset can be used as a large and diverse annotated bot training data for larger and more comprehensive machine learning models.

3   Modeling

3.1   Feature engineering

In order to develop a random string detection model for this unique case, we constructed training data consisting of 200,000 non-random Twitter screen names (randomly sampled from Twitter and manually verified as non-random) and 200,000 randomly generated 15 digit strings. We then developed a combination of heuristic filtering and traditional machine learning models to label the string as *random* or *not random*. This development is described below.

For feature engineering, the primary feature that we extracted from the strings was character n-gram. For string s with length m, a character n-gram is the $(m - n + 1)$ sequential substrings of length n found in string s. In our case, we explored several settings for n, to include using multiple values in the same feature set (i.e. using both bigrams and trigrams).

We then transformed the resulting sparse character n-gram matrix using term frequency-inverse document frequency (TF-IDF). TF-IDF is defined in Equation 1 and 2 below, and is used to scale the characters by the information that they provide. In our case, frequent characters in a string provide information, but not

if they're frequent in all of the strings. To calculate the IDF for character c in strings s, we take the logarithm of the ratio of the total number of strings in corpus S by the number strings that contain c, as shown in Equation 1.

$$idf(c, S) = log\frac{N}{|\{c \in S : c \in s\}|} \tag{1}$$

We then calculate the TF-IDF for character c in string s found in corpus S as follows

$$tfidf(c, s, S) = tf(c, s)idf(c, S) \tag{2}$$

This therefore weights characters that have a high local frequency but a lower global frequency. At first it may seem that TF-IDF is unnecessary since each character n-gram is equally likely in random strings, given a strong pseudo-random number generator. n-grams are not equally likely for human generated strings, however. Given this fact we felt it appropriate to transform the data with TF-IDF.

These features were merged with several other features. We started by merging the normalized count of upper case, lower case, and numeric characters. n-gram generation by default converts all text to lower case. We maintained this default behavior, but saw that the number of upper and lower case in letters in particular provided a strong signal. Since our training data contained some human generated strings that were not 15 characters in length, we normalized these counts.

Additionally, we included the Shannon string entropy in our feature set. Shannon string entropy, while not strong enough to use by itself in our case, still provides a strong signal that we felt would be useful. We will test this assumption below. Shannon entropy is defined in 3, where p_i is the normalized count for each character found in the string.

$$H(A) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{3}$$

The A full table of features is given in Table 2.

Table 2: Features for Random String Detection

Feature	Type	Description
Character Bi-gram	Numeric	Term frequency inverse document frequency of bi-gram
No. lower case	Numeric	Normalized count of lower case letters
No. upper case	Numeric	Normalized count of upper case letters
String entropy	Numeric	Shannon String entropy

We used the *scikit − learn* package [22] to explore and build the machine learning classification model for Random Strings. We evaluated Naive Bayes, Logistic Regression, and Support Vector Machines (SVM) with 10 fold cross-validation. The results are presented in Table 3. We conducted model comparisons between these models, and found SVM and Logistic Regression did are not statistically different ($t = 0.62912$, $df = 18$, $p.value < 0.5372$). Given these results, we used Logistic Regression for our production model, given that it is simpler and faster. Note that this result entails significantly more training data than we used in earlier research (see [6]), where SVM performed better.

Table 3: Model Performance in Classifying Randomly Generated Strings for Screen-names

Model	Accuracy	F1	Kappa	Precision	Recall	ROC AUC
Log. Regression	0.996	0.996	0.991	0.994	0.997	0.999
Naïve Bayes	0.969	0.97	0.939	0.947	0.995	0.996
SVM	0.996	0.996	0.993	0.995	0.998	1

Before predicting whether or not a string was random, we first applied several heuristic filters. These verified that 1) the string was 15 characters in length, and 2) contained at least one capital letter, lower case letter, and numeric digit. This final filter was applied given that 15 character strings have a 0.02% chance of not containing a capital or lower case letter and a 7% chance of not containing a numeric digit. This heuristic was applied given that precision was a higher priority than recall.

In Figure 1 we evaluate the best value of n (number of characters for n-gram) as well as whether or not using Shannon's Entropy as a column feature provides leverage in prediction. In this visualization we see that bigrams with Shannon's entropy provides the best leverage in predicting random strings.

In addition to exploring the feature based machine learning models discussed above, we also explored the use of Markov model of character sequencing, but found during initial exploration that this did not have sufficient power to classify the strings given the inherent random nature of human generated screen names. Additionally, we explored using Shannon entropy as the only measure for filtering these strings. Once again, while helpful, this method did not demonstrate sufficient power for our purposes.

3.2 Model Deployment

Our primary use for the algorithm was to filter accounts with 15 character random strings from a Twitter data stream. To do this we ran a random sample from the Twitter Streaming API from 23 December 2017 to 20 June 2018. During this time the stream collected approximately 433 million tweets. This collection was
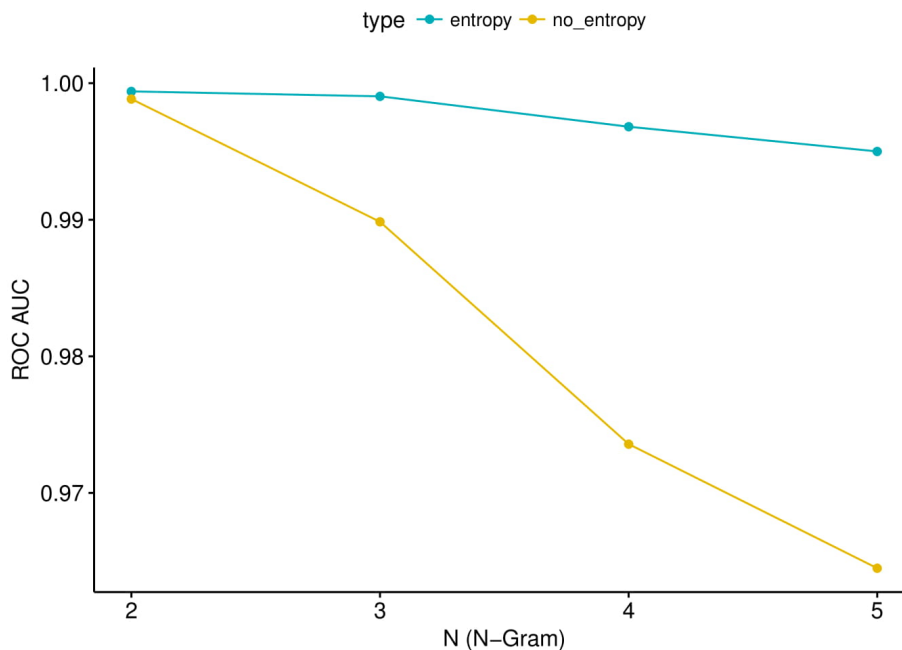
Fig. 1: Evaluating n (number of characters in n-gram) and use of Shannon's entropy as a feature

done without any semantic or geographic filters, and stored the raw JSON files that are returned by the Twitter API.

Having performed the collection, we next applied our algorithm to all 433 million tweets, filtering out all accounts that were labeled as having 15 digit randomly generated screen name. This produced a collection of 7.8 million tweets from 1.7 million unique accounts.

4 Model Evaluation

Given the desired use case of annotating diverse bot accounts, we conducted two evaluations on our results. First, we wanted to estimate the false positive rate on our random string detection, since false positives have a high likelihood of not being an autonomous account. To accomplish this we randomly selected 1,000 of the screen names that were labeled as random, and manually identified those that contained clear words or acronyms. Given this method, we estimate that our false positive rate is approximately 1%.

Additionally, we wanted to estimate the percentage of random character screen name accounts that are automated, or appear automated. In other words, how many of our true positive random string accounts are truly bots. To estimate

this, we randomly sampled 100 accounts, verified that the user name appeared random, and inspected the account in the Twitter web client. Of the 100 that we manually inspected, five were suspended, eight provided no results (most likely the account was closed by the user), and all others exhibited autonomous behavior. After thoroughly evaluating these 100 randomly sampled accounts we were were satisfied that this methodology provides annotated bot data that is at least as accurate as *honey pot* data, and likely has a wider range of bot types.

4.1 Data Characterization

One of our first tasks in exploring the data is to understand how these accounts differ from the average Twitter account, and whether those differences were uniform across the language of the bot creator.

99% of the 7.8 million tweets in this dataset are associated with seven languages. It's interesting to note that none of the Continental European Languages (French, Spanish, German, Portuguese, Italian, etc) are in this list. Somewhat surprisingly, the proportion associated with Japanese and Arabic accounts is very high, second only to English. A full breakdown of the languages and a short general description of our observations are provided in Table 4. Only 840 tweets contained coordinate locations, and these locations are strongly correlated to the languages mentioned below (United States, Japan, the broader Middle East, Russia, and Thailand).

The major observations from Table 4 are that the random string accounts are younger, less popular, and less active than the average Twitter account. We see that the median age for the random bots is 224 days, compared to 1,248 days for your average active Twitter account. The median number of followers/friends ratio for the random string bots is 6/39 versus 277/294 for the average Twitter account. We also see that the median random string bot account only produced 54 tweets over its lifetime, versus 8,216 for the average account (this comparison is affected by age difference).

While some languages (Arabic, Japanese, Korean, and Thai) appear to be slightly more popular and active, in general these random string accounts appear to have a high number of accounts that are dormant, or at least in a state of low activity. Some of these may be waiting to be activated for a given event or task, while others may be used for intimidation attacks (as some of these were with the Yemen journalist discussed above). Intimidation accounts (accounts that follow a user in mass) do not need to be active or popular. Their intent is to push another account out of the Twitter conversation through intimidation.

Given the fact that our data set contains primarily bot accounts, we observed a number of account suspensions during the course of our study. Between mid December 2017 and August 22 2018, 247,022 accounts (~15%) were suspended by Twitter, while 46,985 accounts (~2.7%) were removed by the user. As the media and politicians put pressure on Social Media companies, the natural response is to increase their policing of this automated behavior on their platforms.

Table 4: Summary Statistics by Language

Language		Arabic	English	Japanese	Korean	Russian	Thai	Chinese	other	Normal*
# of Accounts		246K	626K	593K	103K	61K	47K	21K	18K	1599K
Age	min	61	61	61	61	62	61	62	61	6
	25%	181	105	214	193	162	167	186	192	487
	50%	264	165	361	260	292	246	288	297	1,248
	75%	413	213	570	427	365	383	423	626	2,235
	max	3,046	17,763	3,731	3,020	3,075	3,306	3,431	3,662	4,421
	mean	326	210	449	342	322	310	357	550	1,412
	std	216	253	315	229	228	219	247	609	1,008
Followers Count	min	0	0	0	0	0	0	0	0	0
	25%	3	0	2	0	0	1	0	2	78
	50%	15	2	19	2	1	5	1	15	277
	75%	63	17	108	10	5	24	6	85	818
	max	828K	1087K	1322K	23,681	54K	177K	50K	944K	40,550K
	mean	171	61	136	32	93	163	97	295	3376
	std	2,716	2,054	2,366	268	1,013	2,044	921	7,581	94990
Friends Count	min	0	0	0	0	0	0	0	0	0
	25%	26	5	10	1	6	31	7	29	118
	50%	79	26	49	21	31	88	32	91	294
	75%	226	73	168	74	53	258	70	242	695
	max	640K	349K	75K	25K	17K	18K	12K	88K	2,441K
	mean	297	101	178	92	130	257	98	298	1044
	std	1,543	682	482	326	594	498	322	1,034	8227
Tweet count	min	1	1	1	1	1	1	1	1	1
	25%	15	6	25	26	16	24	15	19	1813
	50%	71	20	117	134	69	99	59	109	8216
	75%	319	83	515	601	234	370	286	627	27318
	max	532K	806K	994K	228K	114K	570K	106K	304K	16,176K
	mean	819	301	930	934	456	684	517	1,727	26,652
	std	3,753	3,180	4,301	3,652	2,226	4,195	2,036	7,981	66,180

* *Normal* Twitter Accounts were sampled from the Twitter Streaming API

5 Conclusion

Research in this area is limited by a rich enough data set that supports identification of the wide range of types of bots, and that is sufficient to support studies of bot-evolution. While the data used herein begins to address this issue, it is by no means comprehensive and needs further expansion. We are working on such expansion. However, restrictions on data sharing make it difficult to share this data. Consequently, we are also working on data format that can be shared.

Bots are part of the conversation in social media. But not all bots are the same. They vary in what they do, how they do it, and intent. While some bots act independently others work in concert and still others are part of a cyborg - a human-bot partnership. Research is needed to characterize types of bots and their evolution. Research is also needed to identify the mapping between types

of bots in use and types of information maneuver or social-group creation that, that type of bot supports or thwarts.

6   Future Work

Our future effort begins with the exploration of this dataset so that we can cluster these accounts by type and function. We then intend to develop and train several specialized as well as a general purpose bot detection algorithms for use in detecting and classifying bots. Once complete, our effort will shift to the detection and characterization of bot networks and the actors behind them.

ACKNOWLEDGMENT

References

1. Twitter rate limiting. https://developer.twitter.com/en/docs/basics/rate-limiting. Accessed: 2018-05-02.
2. A. Ananthalakshmi. Ahead of malaysian polls, bots flood twitter with pro-government..., Apr 2018.
3. Matthew Benigni and Kathleen M Carley. From tweets to intelligence: Understanding the islamic jihad supporting community on twitter. In *Social, Cultural, and Behavioral Modeling: 9th International Conference, SBP-BRiMS 2016, Washington, DC, USA, June 28-July 1, 2016, Proceedings 9*, pages 346–355. Springer, 2016.
4. David Beskow and Kathleen M Carley. Bot conversations are different: Leveraging network metrics for bot detection in twitter. In *Advances in Social Networks Analysis and Mining (ASONAM), 2018 International Conference on*, pages 176–183. IEEE, 2018.
5. David Beskow and Kathleen M Carley. Introducing bothunter: A tiered approach to detection and characterizing automated activity on twitter. In Halil Bisgin, Ayaz Hyder, Chris Dancy, and Robert Thomson, editors, *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 2018.
6. David Beskow and Kathleen M Carley. Using random string classification to filter and annotate automated accounts. In Halil Bisgin, Ayaz Hyder, Chris Dancy, and Robert Thomson, editors, *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 2018.

7. Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. 2016.

8. Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *Proceedings of the 26th annual computer security applications conference*, pages 21–30. ACM, 2010.

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

10. Emilio Ferrara. Measuring social spam and the effect of bots on information diffusion in social media. *arXiv preprint arXiv:1708.08134*, 2017.

11. David Mandell Freeman. Using naive bayes to detect spammy names in social networks. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 3–12. ACM, 2013.

12. Carlos Freitas, Fabricio Benevenuto, Saptarshi Ghosh, and Adriano Veloso. Reverse engineering socialbot infiltration strategies in twitter. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 25–32. ACM, 2015.

13. April Glaser. Russian bots are trying to sow discord on twitter after charlottesville. 2017.

14. Timothy Graham and Robert Ackland. Do socialbots dream of popping the filter bubble? *Socialbots and Their Friends: Digital Media and the Automation of Sociality*, page 187, 2016.

15. Philip N Howard and Bence Kollanyi. Bots,# strongerin, and# brexit: Computational propaganda during the uk-eu referendum. *Browser Download This Paper*, 2016.

16. Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.

17. Kyumin Lee, Brian David Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. In *ICWSM*, 2011.

18. Al Bawaba The Loop. Thousands of twitter bots are attempting to silence reporting on yemen. 2017.

19. Cristian Lumezanu, Nick Feamster, and Hans Klein. # bias: Measuring the tweeting behavior of propagandists. In *Sixth International AAAI Conference on Weblogs and Social Media*, 2012.

20. Mahdi Namazifar. Detecting randomly generated strings, December 2015. [Online; posted 25 December 2015].

21. LM Neudert, B Kollanyi, and PN Howard. Junk news and bots during the german federal presidency election: What were german voters sharing over twitter?, 2017.

22. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

23. Jayaram Raghuram, David J Miller, and George Kesidis. Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling. *Journal of advanced research*, 5(4):423–433, 2014.

24. Claude E Shannon. The bell system technical journal. *A mathematical theory of communication*, 27:379–423, 1948.

25. VS Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, 2016.
26. John-Paul Verkamp and Minaxi Gupta. Five incidents, one theme: Twitter spam as a weapon to drown voices of protest. In *FOCI*, 2013.
27. Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 48–61. ACM, 2010.