Using Random String Classification to Filter and Annotate Automated Accounts

David M. Beskow(✉) and Kathleen M. Carley

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

Abstract. Automated social media *bots* have existed almost as long as the social media platforms 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 using random string detection applied to user names to filter twitter streams for potential bot accounts and thereby generating annotated data.

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 [9], many others range from nuisance (i.e. a spam bot) to propaganda [14], suppression of dissent [20], and network infiltration/manipulation [1,7]. They have recently gained widespread notoriety due to their use in several major international events, including the British Referendum known as "Brexit" [10], the American 2016 Presidential Elections [2], the aftermath of the 2017 Charlottesville protests [8], and most recently with less publicized reporting regarding the conflict in Yemen [13].

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.

R. Thomson et al. (Eds.): SBP-BRiMS 2018, LNCS 10899, pp. 367–376, 2018.
https://doi.org/10.1007/978-3-319-93372-6_40

Our work makes two 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.

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 [11], the deliberate detection of automated accounts on the Twitter Platform began in earnest in 2010 when [3] 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 [12]. 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 [19].

In 2014, Indiana University and the University of Southern California launched the *Bot or Not* online API service [4]. 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" [19]. 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 have re-branded *Bot-or-Not* to *Botometer*, increasing the set of features to 1,150 account related features [5]. 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 [18] 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 [21], a dictionary approach [15] and Markov modeling [17].

The past research most closely connected to our effort was conducted by LinkedIn in 2013. At that time [6] 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 [13]. 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 4,000 non-random Twitter screen names (randomly sampled from Twitter and manually verified as non-random) and 4,000 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 diagrams 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 Eqs. 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 Eq. 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

$$tfidf(c, s, S) = tf(c, s)\dot{i}df(c, S) \tag{2}$$

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 1.

Table 1. Features for random string detection

Feature	Type	Description
2–3 character N-gram	Numeric	Term frequency inverse document frequency of n-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 [16] 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 2. We conducted model comparisons between these models, and found that the SVM models provided *Highly Significant Improvement* in all cases ($p.value < 0.001$). Given these results, we used SVM for our production model.

Table 2. Model performance in classifying randomly generated strings for screen-names

Model	% Correct	Kappa
Naive Bayes	93.3%	0.8659
Logistic regression	94.25%	0.948
SVM	97.4%	0.975

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 Fig. 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 digrams with Shannon's entropy provides the best leverage in predicting random strings.

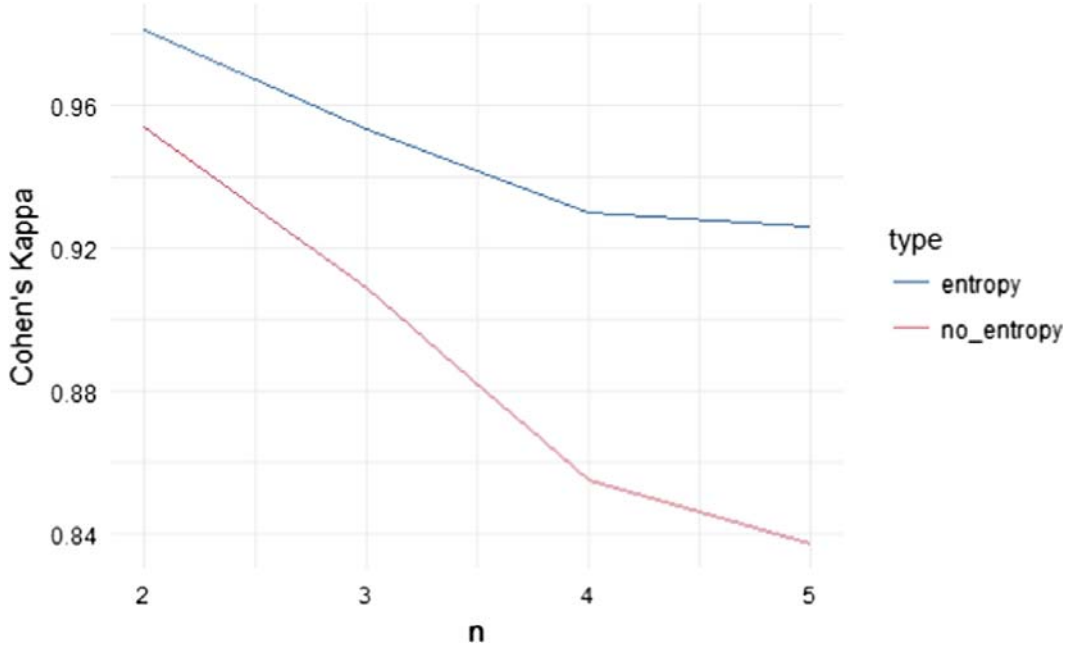

Fig. 1. Evaluating n (number of characters in n-gram) and use of Shannon's entropy as a feature

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 14 December 2017 to 10 January 2018. During this time the stream collected approximately 33 million tweets. This collection was done without any text 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 33 million tweets, filtering out all accounts that were labeled as having 15 digit randomly generated screen name. This produced a collection of 487,000 tweets from 235,000 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 5.6%.

Additionally, we wanted to estimate the percentage of random character screen name accounts that are autonomous, or appear autonomous. In other words, how many of our true positive random string accounts are truly autonomous. 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 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 was to check on those features that are highly indicative of an autonomous account to see if our data exhibited these tell-tale characteristics. In general, autonomous accounts produce tweets at a much higher volume and rate than human actors. The mean number of tweets for our accounts was 7,918 (median = 1,125). In general bots have a low number of followers (most people don't follow bots), but they tend to follow many accounts, trying to build influence. Following this pattern, the median number of followers is in our data set is 55, but the median number of accounts they follow is 130.

94% of the roughly 500,000 tweets in this dataset are associated with seven languages. Somewhat surprisingly, the volume associated with Japanese and Arabic accounts is greater than those associated with English speaking accounts.

A full breakdown of the languages and a short general description of our observations are provided in Table 3. Only 674 tweets contained coordinate locations, and these locations are strongly correlated to the languages mentioned below.

Table 3. Characterization by language

Language	% Total	General description
Japanese	184, 385	High concentration of anime media sharing
Arabic	111, 523	High percentage of young accounts, some automated Koran passage sharing
English	94, 804	Contains a high number of non English hash tags
Korean	41, 870	Varied
Thai	14, 195	High concentration of adult content
Russian	13, 461	Varied

The mean age of the accounts is 274 days, with 50% of the accounts created in the last 150 days and 75% of the accounts younger than 1 year old. The relative young age of these accounts is highly indicative of their automated behavior. The oldest accounts are associated with English account settings, and date back to 2008.

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 mid March 2018, 23,532 accounts (~10%) were suspended by Twitter, while 2,201 accounts (~1%) 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 autonomous behavior on their platforms.

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. This work was supported in part by the Office of Naval Research (ONR) Multidisciplinary University Research Initiative Award N000140811186 and Award N000141812108, the Army Research Laboratory Award W911NF1610049, Defense Threat Reductions Agency Award HDTRA11010102, and the Center for Computational Analysis of Social and Organization Systems (CASOS). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR, ARL, DTRA, or the U.S. government.

References

1. Benigni, M., Carley, K.M.: From tweets to intelligence: understanding the Islamic jihad supporting community on Twitter. In: Xu, K., Reitter, D., Lee, D., Osgood, N. (eds.) SBP-BRiMS 2016. LNCS, pp. 346–355. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39931-7_33
2. Bessi, A., Ferrara, E.: Social Bots Distort the 2016 US Presidential Election Online Discussion (2016)
3. Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S.: Who is tweeting on Twitter: human, bot, or cyborg? In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 21–30. ACM (2010)
4. Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F.: Botornot: a system to evaluate social bots. In: Proceedings of the 25th International Conference Companion on World Wide Web, pp. 273–274. International World Wide Web Conferences Steering Committee (2016)
5. Ferrara, E.: Measuring social spam and the effect of bots on information diffusion in social media. arXiv preprint arXiv:1708.08134 (2017)
6. Freeman, D.M.: Using Naive Bayes to detect spammy names in social networks. In: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, pp. 3–12. ACM (2013)
7. Freitas, C., Benevenuto, F., Ghosh, S., Veloso, A.: 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, pp. 25–32. ACM (2015)
8. Glaser, A.: Russian bots are trying to sow discord on Twitter after charlottesville (2017)
9. Graham, T., Ackland, R.: Do socialbots dream of popping the filter bubble? In: Socialbots and Their Friends: Digital Media and the Automation of Sociality, p. 187 (2016)
10. Howard, P.N., Kollanyi, B.: Bots,# strongerin, and# brexit: computational propaganda during the uk-eu referendum. Browser Download This Paper (2016)

11. Krishnamurthy, B., Gill, P., Arlitt, M.: A few chirps about Twitter. In: Proceedings of the First Workshop on Online Social Networks, pp. 19–24. ACM (2008)
12. Lee, K., Eoff, B.D., Caverlee, J.: Seven months with the devils: a long-term study of content polluters on Twitter. In: ICWSM (2011)
13. Al Bawaba The Loop. Thousands of Twitter bots are attempting to silence reporting on Yemen (2017)
14. Lumezanu, C., Feamster, N., Klein, H.: # bias: measuring the tweeting behavior of propagandists. In: Sixth International AAAI Conference on Weblogs and Social Media (2012)
15. Namazifar, M.: Detecting randomly generated strings, December 2015. Accessed 25 Dec 2015
16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
17. Raghuram, J., Miller, D.J., Kesidis, G.: Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling. J. Adv. Res. **5**(4), 423–433 (2014)
18. Shannon, C.E.: The bell system technical journal. In: A Mathematical Theory of Communication, vol. 27, pp. 379–423 (1948)
19. Subrahmanian, V.S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., Zhu, L., Ferrara, E., Flammini, A., Menczer, F.: The DARPA Twitter bot challenge. Computer **49**(6), 38–46 (2016)
20. Verkamp, J.-P., Gupta, M.: Five incidents, one theme: Twitter spam as a weapon to drown voices of protest. In: FOCI (2013)
21. Yadav, S., Reddy, A.K.K., Reddy, A.L., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 48–61. ACM (2010)