Towards Automated Bot Detection in Political Social Networks

Berat Bicer
berat.bicer@bilkent.edu.tr
Bilkent University
Ankara, Turkey

Vahid Namakshenas
vahid@bilkent.edu.tr
Bilkent University
Ankara, Turkey

ABSTRACT

In this work, we study the effectiveness of some of the state of the art bot detection strategies from the literature that can be executed in parallel on recently proposed TwiBot-20 dataset and analyse how network science can contribute to improving the efficiency of bot detection algorithms. We report a contrastive performance overview of studied methods, and conclude with an analysis into obtained results regarding method efficiency and stability.

CCS CONCEPTS

• **Bot Detection**; • **Network Science**; • **Natural Language Processing**;

KEYWORDS

bot detection, network science, natural language processing

ACM Reference Format:
Berat Bicer and Vahid Namakshenas. 2022. Towards Automated Bot Detection in Political Social Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Social media platforms, since their inception, have grown significantly in terms of both their popularity and wide spread usage, and their ability to impact the socio-economic, political, and cultural discourse of countries around the globe. Such an influence undoubtedly attracts agents with nefarious intent, ranging from individuals to political activists and state actors, who seek to obtain a platform to further their political agendas, which necessitates a unified effort to combat such actors seeking to alter the public's perception through various means.

This project seeks to study how the political discourse on Twitter is influenced by or is otherwise subjected to bot activity and whether it's possible to identify them through computational methods. Specifically, we're interested in a small political network based on the broad bot-detection corpus called TwiBot-20. Our network is focuses on prominent figures in contemporary figures in U.S. politics and their connections. Based on this dataset, we propose multiple bot detection strategies that can be operated independently, raging from graph and node-level analysis based on centrality metrics and community detection, to node-based context encodings and natural language processing for bot detection. Our goal is to offer an overview and test the effectiveness of influential strategies from the literature on TwiBot-20 dataset, while studying how network science can contribute to bot detection in hopes of contributing to the fight against misinformation.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 LITERATURE REVIEW

The earlier proposals for Twitter bot detection concentrate on feature engineering with user information [5]. Lee et al. proposed a suspicious URL detection system for Twitter which focuses on the correlations of multiple redirect chains that share the same redirection servers [10]. Thomas et al. presented a real-time system that crawls URLs as they are submitted to web services and determines whether the URLs are direct to spam also they explored the distinctions between email and Twitter spam, including the overlap of spam features, the persistence of features over time, and the abuse of generic redirectors and public web hosting [17]. Gao et al. provided online spam filtering for social networks by using text shingling and URL comparison to incrementally reconstruct spam messages into campaigns, which are then identified by a trained classifier. Yang et al. designed new and robust features to detect Twitter spammers based on an in-depth analysis of the evasion tactics utilized by Twitter spammers [18]. Other features are also exploited, such as information that is related to the user homepage [9], social networks [15], and timeline of accounts [3].

With the emergence of deep learning, neural networks are increasingly used for Twitter bot detection. Stanton et al. proposed spamGAN, a generative adversarial network that relies on a limited set of labeled data as well as unlabeled data for opinion spam detection [16]. Wei et al. presented an RNN model with word embedding to distinguish Twitter bots from human accounts. Kudugunta et al. proposed a deep neural network based on contextual long short-term memory (LSTM) architecture that exploits both content and metadata to detect bots at the tweet level [8]. Alhosseini et al. proposed a model based on graph convolutional neural networks (GCNN) for spam bot detection regarding its ability to leverage both the features of a node and aggregate the features of a node's neighborhood [1].

3 METHODOLOGY

In this section, we start by specifying distinctions between bots and real accounts, bot in terms of the characteristics of their neighbourhoods and the communities they tend to form; both of which offers clues in regards to how bot detection can be handled. Later, we describe the methodologies used in the rest of the paper starting with graph and node-level analysis, followed by community detection strategies, mapping neighbourhood characteristics as mentioned

before, and then concluding with natural language processing applications.

3.1 Definitions

In this study, our goal is to separate bot accounts from accounts affiliated with real people or organizations. In this sense, a bot account is an account that is automated or has limited human intervention in its operations. A real account, on the other hand, is controlled by people directly. Such accounts' actions are directly undertaken by human operators whereas bot operations can be handled algorithmically.

In a political network, bots and real accounts have unique goals and principals that they operate on. For example, some of the probable goals of a bot account are reporting recent developments regarding public figures and elected officials, as well as their statements and political actions; spreading propaganda and ads in hopes of galvanizing a follower base for a certain political cause, etc. For this reason, automated accounts seek to expand their influence over other real accounts by following as many real accounts as possible, which in turn yields new followers for them. On the other hand, real accounts while they may share similar goals with bots, real accounts interact with real accounts more often than they do with bots; thus, their outgoing connections usually are other real-accounts while incoming connections are a mix of bots and real accounts. For this reason, real accounts generally have large betweenness and operate as hubs.

In some cases, a tightly-connected group of bots form what can be described as botfarms. These groups generally seek to boost the popularity of a certain post or attempt to spread an idea to deceive other real accounts into believing the idea is genuine and is widespread. Most of the time, hierarchical clustering and timestamp-based analysis on site-wide interactions are strong tools for detecting these botfarms. This is so because there are two main structure botfarms form: One resembles the small world phenomena and creates densely packed groups of bots. The other creates chains of connected bots forming a hierarchical network through which information passes down.

3.2 Graph and Node-Level Analysis

In this section, we share some probable strategies for bot detection based on graph-level and node-level metrics discussed during the semester. These candidates include density and node degree, average geodesic distance for a node, eigenvector centrality, betweenness, and lastly PageRank values.

As the density of a network refers to the number of dyadic linkages between its nodes, a low density manifests a sparse graph. Comparing the average density and density of each node can give us some ideas about determining whether the account associated with a node is a bot. For example, if a node has a high density compared to the average, it can be deduced that this node clamors for attention. In this case, we encounter three different modes. In the first mode, if the node has a high number of outgoing edges (followings) and only a mere number of incoming edges (followers), that node might be considered a bot. In the second mode, it is possible to have a high number of followers and a low number of followings. This situation can be a characteristic of a critical node: A node that is a touchstone and can be a basis for spreading information (such as certain politicians). In the third mode, we have a node that has a high number of followers and followings. In this case, we have to investigate reciprocity. If the connections are mostly symmetric, it can be concluded that the node belongs to a human; otherwise, the node can be considered a bot account. We assume that if an account belongs to humans, the followers and followings should be mostly the same. However, this deduction is not robust and needs to be scrutinized by other measurements. The density formula for our directed graph is as follows:

$$D = \frac{L}{N(N-1)}$$

where L is the number of edges and N is the number of nodes in the graph.

Another important metric is the characteristic path length, or more specifically, average geodesic distance for each node. Seemingly our network does not meet the "Six degrees of separation" theorem since the characteristic path length is roughly long. While it may be possible that, in a network containing many bot accounts, characteristic path length may be quite large, we do not have baselines for reliable comparisons. Thus, we instead opt-out for average geodesic distance instead; We observe a correlation between the average geodesic distance for a given node and the likelihood that the associated account is labeled as a bot based on the property that real accounts tend to be hubs or operate in a similar fashion. For computing the geodesic distance, the formula may be utilized where $d(i, j)$ is the geodesic distance from node i to node j:

$$d_i = \frac{\sum_{j=1}^{n} d(i, j)}{(n-1)}$$

Considering the definition in the previous section, a bot farm consists of many bots connected and can influence real accounts by over-advertising. One exciting metric which corresponds to this issue is eigenvector centrality. As per definition, an influential node that is highly connected to another strongly connected nodes is considered to have a high eigenvector centrality. In our case study, we can detect the important bot nodes that are connected to bot farms. Although not all dense clusters are bot farms, we can locate candidates for further investigations.

Betweenness centrality identifies the nodes that can function as brokers or gatekeepers. Hence, from a semantics perspective, the nodes connecting two groups to each other have to be meaningful. They have to share some common aspects with both groups. Hence, in our case, the bot nodes should be the nodes with a low betweenness centrality value. A bot account cannot be a broker between two groups of real accounts. However, in this case, we assume that both clusters contain real accounts. The formula for the betweenness centrality is:

$$b(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

$b(v)$ is the betweenness of node v, $\sigma_{st}(v)$ is the number of shortest paths between s and t that pass through v, and σ_{st} is the number of shortest paths between s and t.

According to PageRank's definition, the Nodes are important that have a high number of incoming edges. Therefore, one possible

methodology will be calculating the PageRank metric to describe a node's importance. Obviously, a node having a high PageRank value is a reference node. Such a node has the least possibility of being a bot account.

3.3 Community Detection

Cluster Analysis or clustering, also referred to as community detection, is a process that can be used to separate a set of objects into distinct groups called clusters such that members of each cluster display similar properties amongst themselves. For this reason, The members of each cluster show strong similarities to one another compared to the the degree of similarity between members of distinct clusters. The purpose of clustering is, therefore, to assign labels to objects that indicate the membership of each object to their respective cluster. One key distinction between clustering and classification is that during clustering, we assume datasamples lack their primary labels, thus, cluster assignments are performed without this knowledge:; contrary to classification. As a result, clustering is considered an unsupervised learning strategy whereas classification is referred to as a supervised learning stretegy.

While there are many different clustering algorithms, these may be categorized broadly as hierarchical clustering and expectation-maximization based algorithms. In hierachical clustering such as DBSCAN[4], nodes and clusters are iteratively merged to form larger clusters. Expectation-maximization based approaches such as k-nearest neighbours (k-nn) algorithm [12], on the other hand, attempt to solve the problem of partitioning the samples into m distinct cluster such that within-cluster variances are minimized, which is in fact a NP-Hard problem. Thus, such algorithm offer heuristic solutions that converge to local maximas rather than an optimal solution.

For the purpose of this study, we believe attempting to construct few high-level clusters may not be informative for bot detection since we expect the inter-cluster variance will be quite high. In other words, using k-nn to construct two clusters, one for bots and one for real accounts, likely will fail to construct pure clusters. Thus, we instead propose employing hierarchical clustering to detect many unique communities that share similar properties within, which can also help locating bot farms discussed earlier. These farms are mainly described as a chain or cluster of connected nodes attached to a few hub-like anchor points. Since hierarchical clustering starts at the leaf nodes, in a few iterations the entire structure can be grouped together and studied later for verification.

3.4 Neighbourhood Embedding

As mentioned in Section 3.1, characteristics of the connections of a certain node is a rich source of information when inferring bots based on two factors: Bots generally have high out-degrees than in-degrees and they tend to cluster around real accounts. Real accounts on the other hand have a mixture of bots and real accounts amongst their neighbourhoods, yet we expect the ratio of bots to real accounts in outgoing connections to be significantly smaller than the same ratio in incoming edges. For these reasons, we propose encoding the characteristics of a node's connections based on the labels of its neighbours by concatenating the aforementioned labels in an arbitrary order to construct a binary string of varying length. This string can then be used in multiple ways: for example, by converting it to base-10 and normalizing the same quantity across the graph, we can apply thresholding for detecting bots based on previous criteria. We cound also directly use the string by padding it to the max length, the number of connections of the node with the highest number of neighbours, and directly use the strings as binary features for classification through traditional classifiers such as support vector machines (SVM) [2], random forests, etc. Since we know the ground truths for each node in the dataset, the predictions employing these features can be validated through comparison with the ground truths. Implementation-wise, connections of a node can be used directly to visit its neighbours, thus spatially-aware hashing methods such as kd-trees are not necessary.

3.5 Natural Language Processing

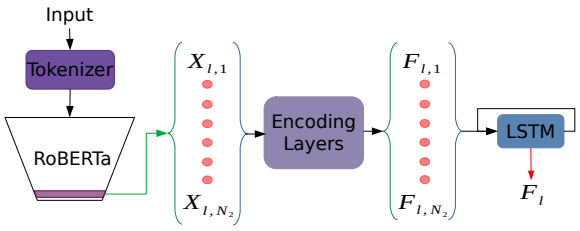

Figure 1: Visualization of the preliminary design of the text model used for tweet-based bot detection.

Natural language processing (NLP) is a fundamental part of our analysis. Our method seeks to convert a sequence of tweets into a 1-dimensional tensor representation for inferring whethet an account is a bot based on the following pipeline. Note that this preliminary design is subject to change based on performance measures.

Firstly, just like many NLP applications, we first plan to filter out the dataset due to difficulties expressed in Section 4, namely existence of nonunicode characters, emojis, Twitter tags, HTML elements; mentions of other people and links to other tweets, accounts, and external sites, etc. As described previously, regular expression matching can be utilized for this purpose, however, we predict a significant amount of manual labour will be required to eliminate some of the aforementioned problems.

Second, having filtered out most of the inherent problems of the dataset, we employ a tokenizer for detecting language elements based on semantics and grammatical rules, and to convert the detected tokens into vectors for further processing. At this stage, we believe a transformer bsaed tokenizer such as RoBERTa [11] is a good candidate, however, recurrent units and transformers are known to be incompatible in many language tasks, For this reason, we may opt-out for using traditional vectorization algorithms such as *word2vec* [13, 14] if necessary.

After tokenization, we trained a recurrent encoder that reads the tokens in sequence and outputs a single context encoding for the entire tweet, encapsulating its information content for bot detection. Our preliminary design for the said network can be found in Figure 1. According to this, we first apply tokenization using

a RoBERTa-based transformer, which outputs token embeddings. We then process each token by a sequence of encoding layers and finally pass each token through the recurrent unit to obtain the final context embedding. Here, LSTMs [7] are viable recurrent units thanks to their ability to deal with long temporal chains. After each statement for a specific subject is processed this way, we plan on averaging the context embeddings to output a single subject-level representation that can be used for classification. Note that thanks to the nature of this design, entire structure is end-to-end trainable.

Lastly, using validation data set aside from the exiting training set, we plan to engage in hyperparameter optimization and design validation to obtain the highest detection score possible. This will followed by a test step where, by comparing node-level ground truths, classification score obtained by the model will be reported.

4 DATASET

TwiBot [5, 6] is a bot detection dataset collected from Twitter users. In [5], which is referred to as TwiBot-20, there are over 2000 unique accounts that include prominent figures from contemporary U.S. politics and their followers engaged in political discourse. Each node has a certain number of neighbours that form follows/followed by relationships in form of outgoing and incoming edges. There are also 200 tweets associated with each node for text-based classification, as well as ground truth annotations to indicate whether a Twitter account is a bot. These labels form the basis of our validation strategy: predictions from the methodologies mentioned previously will be compared to these annotations for computing classification accuracy, as well as class-wise classification accuracy for bots and real accounts separately.

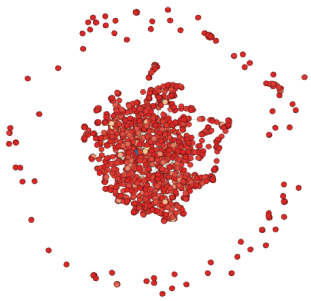

Figure 2: Layout of the vertices in TwiBot-20 under politics category visualized after removal of disconnected nodes.

While TwiBot-20 is useful for political bot detection, it has some shortcomings that we need to address. Firstly, node degrees in the dataset aren't uniform. There are disconnected regions that may, in truth, be connected to the rest of the graph which is caused by limited number of connections the dataset includes for each node. This requires a manual filtering of isolates by eliminating the nodes having zero total degree before the graph-based analysis can be made. Moreover, some accounts that do not participate in political discourse are also removed from the dataset, which yields approximately 1,800 unique nodes.

Second, the dataset contains two broad, disconnected clusters of nodes: a disk-like region outside the center cluster containing disconnected communities and the center cluster itself which houses the bulk of the nodes and their connections (see Figure 2). This implies that any detected cluster larger than a certain size is unlikely to be informative for the context of bot detection. In other words, since high-level clusters are expected to be strongly intertwined based on visualizations obtained, bot detection based on these clusters would likely result in poor detection performance. We plan to address this issue by employing hierarchical clustering for detecting local communities instead, which can be informative for small communities of bots operating interdependently. Another issue to consider that relates to this problem is the existence of the outer ring and its disconnectivity to the center cluster. Since there are almost no connections between these massive clusters, and the fact that most prominent people reside in the center cluster instead of the outer ring, later on in our analysis we may exclude the outer ring completely since the center cluster appears more interesting in terms of bot detection.

Thirdly, while there seem to be sufficient tweet data to justify an extensive natural language processing-based (NLP) bot detection approach, many problems exist regarding the tweets provided that necessitate a preprocessing step before tokenization, which is a crucial step in any NLP pipeline. For example, there are many non-unicode characters such as emojis and Twitter tags exist that needs to be eliminated through regular expression matching. Next, many informal language elements, slangs and abbreviations need to be filtered out since these may not exist within the tokenizer's dictionary and would likely cause errors otherwise. Also, links to other tweets and websites, as well as HTML tags can be found in many tweets. Lastly, some tweets tend to be too short in its content, which makes it harder to understand the concept of the tweet independent from the other tags, people that are linked, and those the subject replies to, etc. As mentioned previously, while most of these problems can be filtered out by regular expression matching, we may need to manually fix in particular abbreviations and informal language elements before the tweets become usable.

REFERENCES

[1] Seyed Ali Alhosseini, Raad Bin Tareaf, Pejman Najafi, and Christoph Meinel. 2019. Detect me if you can: spam bot detection using inductive representation learning. In *Companion Proceedings of The 2019 World Wide Web Conference*, 148–153.

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

[3] 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, 58–64.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* number 34. Vol. 96, 226–231.

[5] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. Twibot-20: a comprehensive twitter bot detection benchmark. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 4485–4494.

[6] Shangbin Feng et al. 2022. Twibot-22: towards graph-based twitter bot detection. *arXiv preprint arXiv:2206.04564*.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9, 8, 1735–1780.

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

[9] Kyumin Lee, Brian Eoff, and James Caverlee. 2011. Seven months with the devils: a long-term study of content polluters on twitter. In *Proceedings of*

the international AAAI conference on web and social media number 1. Vol. 5, 185–192.

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

[11] Yinhan Liu et al. 2019. Roberta: a robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[12] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, 281–297.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

[15] Amanda Minnich, Nikan Chavoshi, Danai Koutra, and Abdullah Mueen. 2017. Botwalk: efficient adaptive exploration of twitter bot networks. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*, 467–474.

[16] Gray Stanton and Athirai A Irissappane. 2019. Gans for semi-supervised opinion spam detection. *arXiv preprint arXiv:1903.08289*.

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

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

Received December 3, 2022; revised XXXXX; accepted XXXXX