Interactive Twitter Bot Detection through Network Graph Analysis

Jon Womack, Albert Chen, Madelyn Scandlen, Anirudh Prabakaran, Shannon Isaacs*

ABSTRACT

In this project, we present a scalable, interactive bot annotation pipeline for the detection of Twitter bots, combining network graph analysis with human input. Our approach addresses the limitations of automated bot detection methods in identifying sophisticated bots that imitate human behavior. We implement key modules, including graph embedding, hierarchical clustering, and an engaging user interface to facilitate the annotation process. We evaluate our pipeline using real Twitter data, specifically the Twibot-20 and Twibot-22 datasets, and conduct a user study involving five participants to assess the system's efficiency.

Our method's novelty lies in the integration of clustering, visualization, and user interface, expediting the efforts of social media moderators to rapidly identify bots with high confidence. In our user study, the majority of users (3/5), all of whom had no prior moderating experience, achieved accuracies (100, 88, 87) surpassing the accuracy (79) of the state-of-the-art BotRGCN classifier, while gaining nearly an order of magnitude in annotation speed (average annotation speed 7.5x faster) [7]. Our work contributes a valuable tool for quick and accurate bot detection.

1 INTRODUCTION

Since the early 2000s, the influence of social bots on public opinion via social media platforms has been a growing concern. [8] These social bots, which mimic human activity at high speeds, are challenging to identify and are often involved in malicious activities such as spam, phishing, propaganda, suppressing dissent, and infiltrating networks. These activities undermine the fundamental purpose of social media platforms, which is to prioritize authentic human interactions.

Developing a successful tool for social media bot moderation would provide a solution to this pressing issue. Our proposed approach, which focuses on labeling clusters of accounts rather than individual ones, aims to enhance the efficiency of moderators by several times. We will measure the success of our tool by evaluating the ratio of individual accounts viewed by moderators to the number of accurately labeled bot accounts.

2 PROBLEM DEFINITION

Instead of identifying bots individually, our method employs embeddings to represent bot behaviors. This approach allows for the clustering of bots with similar behaviors. By utilizing a cluster batch annotation process, bot moderators can achieve a significantly higher rate of bot identification. **Our primary objective is to design an account processing/clustering pipeline and a user interface that effectively presents bot behavior clusters to a moderator. By doing so, we aim to leverage human expertise and cluster visualizations to accurately and swiftly identify large groups of bots, thereby improving the overall moderation process on social media platforms.**

3 LITERATURE SURVEY

In conjunction with to the rise of content generating models like ChatGPT [14] and Dalle [15], recent detection methods have utilized characteristics that are less easily subverted by adversarial manipulation, like hand-crafted node features (centrality, authority, and eccentricity) and structural embeddings [1]. In fact, the top performing models on the most comprehensive twitter datasets (Twibot-20 and Twibot-22) are graph-based approaches [5].

Botometer provides a publicly available user interface with multiple bot scores (financial, scammer, etc.) for a given account from a supervised classifier [20]. Similarly, we will have a user interface that displays relevant account information such bot-language likelihood and a profile overview.

However, rather than automate bot detection, our approach enables human annotation. A method similar to our approach proposed by Papadopoulos et. al is Scaling Instance Annotation, which clusters instance segmentation masks and then propagates human assigned labels to the rest of the cluster [12]. Our tool extends this idea to a novel multi-modal domain and improves upon the random sampling within a cluster

*All authors contributed equally to this work.

by giving the moderator a choice in sample selection aided by a T-SNE embedding visualization [16].

There are many approaches to bot detection currently. One approach focuses on using the textual content of Tweets as input to deep-learning models to identify Twitter bots [4]. Wei and Nguyen improved on this deep by developing a long short-term memory model (LSTM) on sequential word-embeddings of a single Tweet to classify a user as bot [19]. Our natural language module for producing bot probabilities currently uses tweets alone for bot probabilities.

Abbaspour and Moghaddam suggest a novel friendship preference feature, which can scale with growing followers [10]. The method also uses screen names, descriptions, counts of followers and following, and profile images. Hayawi et. al adopt a similar approach utilizing LSTM units to process metadata features and generate bot probabilities [9]. The next iteration of our natural language processing module will include metadata.

The most successful deep-learning approach for bot detection uses word-embeddings and user metadata as input to a neural network that can predict the likelihood a user is a bot [11] [3]. Deep Profile-based Bot (DeeProBot) detection framework employs GLoVe word embeddings and user metadata to learn representations of users [9] [13], which inspires our work to generate a bot-score for users which can then be used to create similar user clusters.

Using the graphical structure of accounts has proven to be a useful feature to further advance bot-detection. BotRGCN demonstrates that a relational graph convolutional network with encoded user data can detect bots [7]. A Graph Attention Network incorporates an attention mechanism to Graph Neural Networks, which allows it to better discern the significance of neighboring users when aggregating information [17]. Our embeddings used for hierarchical clustering are based on graphical structure and are entirely unsupervised.

These methods are all end-to-end bot detection frameworks, however to the best of our knowledge, there is no publicly available framework for bot detection with a human-in-the-loop. We hope that our method enables efficient labelling, while surpassing the accuracy of existing methods.

4 METHOD

Our method of bot detection combines several state-of-the-art techniques with an innovative human-in-the-loop process. A human moderator annotates a few accounts from a cluster before propogating those labels to the rest of the accounts in the cluster. Clusters are formed via hierarchical agglomerative clustering on network topology. Prior to being presented to moderators, clusters are selected based on "cluster scores", which are calculated from tweets input to a bidirectional-LSTM and ordered based on size.

4.1 Language Modeling

To provide a baseline likelihood that an account is a bot, we used a natural language model on the tweets of each user to generate a probability of bot-likelihood. We utilized Wei and Nguyen's bidirectional long short-term memory (bi-LSTM) neural network that used GloVe word embeddings on user's tweets as input. Each user's tweets are combined to a single sequence, where the sequence is then tokenized and the tokens are converted into 25-dimensional GloVe Twitter embeddings. The model consists of 3 stacked bi-LSTM layers with an attention mechanism with ReLu activation and a softmax activation layer that outputs the probability a user is a bot (label=1) and the probability a user is a human (label=0).

4.2 Graph Embeddings

Nodes that reside in different parts of a graph can have similar structural roles within their local network topology. Our team used GraphWave to produce embeddings for the Twibot-20 dataset [2]. Graphwave represents each node's local network neighborhood via a low-dimensional embedding by leveraging wavelets to produce embeddings in an unsupervised way. However, graphwave requires hyperparameters, so we tuned the number of characteristic function samples (ϕ) and the order of Chebyshev polynomials $T_n(x)$ and $U_n(x)$ used to generate our embeddings. For each hyperparameter, the following values were used in a grid search $\phi, n = 4, 8, 16, 32, 64$.

4.3 Hierarchical Clustering

We employed the Fastcluster Python library to execute hierarchical clustering on the high-dimensional

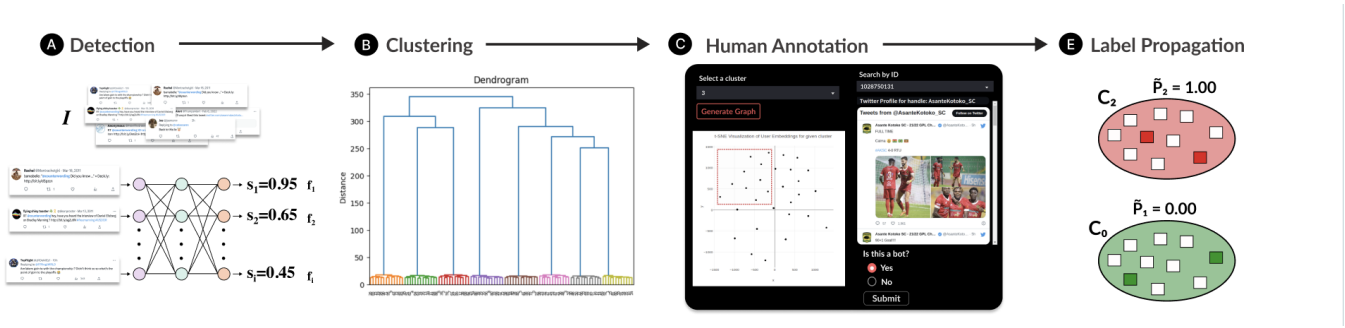

Figure 1: Pipeline for detecting bots at scale

embedding data. The library utilizes Ward's method, a technique known for its efficiency and capacity to generate balanced, compact clusters. This method aims to minimize the intra-cluster variance by merging clusters that result in the smallest increase in total variance [18]. Due to its favorable performance on high-dimensional datasets, Ward's method was deemed appropriate for our analysis.

The Fastcluster library produces a dendrogram matrix as output, which visualizes the hierarchical clustering process. To facilitate a streamlined workflow, we have developed utility functions that enable the selection and investigation of sub-clusters based on their respective cluster scores. This approach provides a comprehensive understanding of the underlying patterns and relationships within the clusters, contributing to an intuitive and useful tool to annotate bot clusters.

Solution to Cluster Calculation Scaling. Agglomerative hierarchical clustering algorithms have a time complexity of $O(n^3)$ [18], which can be computationally expensive for large datasets. The Twibot-22 dataset has over a million labels, which was difficult for hierarchical clustering. We partition the dataset into several segments. This allows us to run clustering on more smaller and tractable subsets of data. The benefit of clustering accounts with similar behavior remains while also enabling the scaling out of cluster calculations.

4.4 User Interface

Streamlit Application. Streamlit was chosen as the package to create the user interface due to its native caching capabilities and ease of use. Before starting the application, clusters containing users and embeddings must be loaded into the SQLite database using provided scripts. All entries initially have a label of 0 (unassigned).

The annotator can use a reset flag to modify the database before launching the Streamlit application. Once launched, the annotator selects a cluster, performs a t-SNE reduction, and investigates Twitter user IDs. The right-hand section displays user profiles for examination, allowing annotators to label profiles as bots or not.

The process can be repeated for several users within the same cluster before submitting. A majority vote is used to determine and propagate labels to all nodes in the given cluster, which is then stored back into the SQLite database. This method significantly accelerates the manual labeling process.

T-SNE Visualisation. T-SNE allows the visualization of high-dimensional data in a lower-dimensional space, which makes it easier to identify and interpret potential clusters or patterns in the data. Using t-SNE to present a cluster to a user allows the user to identify any possible sub clusters that may represent different behavior. They can use the t-SNE visualization to select accounts that may represent several sub-clusters. This provides coverage in identifying if there is any heterogeneity in the cluster in terms of bot/not-bot classes. In most cases, a few account observations of accounts in a variety of positions in the t-SNE visualization is enough to annotate the whole cluster.

5 EXPERIMENTS

This section contains the set-up and results of the language and graph models and the clustering algorithm, as well as the specifics of the user-study.

5.1 Experimental Settings

The datasets Twibot-20 and Twibot-22 were obtained by Feng et al. [5][6] .

- *Language Modeling* The tweet data was tokenized and cleaned using GloVe embeddings [13]. The bi-LSTM was developed using PyTorch 1.13.1 and training was performed on a machine with 8GB RAM and a 1.6 GHz Dual-Core Intel Core i5 CPU.
- *Graph Embeddings* The user graph was developed using NetworkX 1.11. Graph embeddings were carried out on a machine with 8GB RAM.
- *Clustering* Clustering was carried out on a single ec-highmem-8 machine with 8vCPUs and 64GB of RAM.

5.2 Language Modeling

We used the Twibot-20 dataset as the training set for the model, withholding 20% of users for the validation set. The Twibot-20 dataset had 1.5 million tweets for 10k users with 5092 users having the maximum 200 tweets. After training for 10 epochs, the results for the training set of 8223 users and the validation set of 1173 users are shown Table 1 below.

	accuracy	precision	recall	F1
training	0.738	0.714	0.892	0.793
validation	0.705	0.670	0.895	0.767
overall	0.734	0.708	0.893	0.790

Table 1: Accuracy, precision, recall, and F1 of the Bi-LSTM on Twibot-20 training and validation sets.

Using a Kolmogorov-Smirnov goodness-of-fit test, we found the training and validation probability score distributions were not significantly different (p>0.01), we chose to use the training probabilities with the validation probabilities as the bot-score for testing our interface on the Twibot-20 dataset in the user study, as the size of the validation set was not large enough. Cross-validation would have been a better method for generating probabilities on the training set, however due to the constraints of time, the original training-validation split from Feng et al.'s data was used [6].

The goal of the project was to use the Twibot-20 tweets as the training input for the model and then generate probabilities on the unseen Twibot-22 tweets. However due to time and computational constraints, we were unable to scale up to 1 million users and 86 million tweets in the Twibot-22 dataset. We instead generated probabilities for each user for the end-model with 70% accuracy, reported by Wei and Nguyen's model [19].

5.3 Graph Embeddings

The hyperparameters (number of characteristic function samples, order of Chebyshev polynomials) were tuned via a grid search using the Twibot-20 dataset. The grid search provided the following hyperparameters $\phi = 16, n = 64$, which were later used to produce embeddings on the Twibot-22 dataset. 693762 of the million twitter accounts in the Twibot22 dataset had corresponding network structure information. Embeddings were produced for each of these accounts.

5.4 Hierarchical Clustering

Hierarchical agglomerative clustering with ward linkage was run over 7 partitions of 100k embeddings each. Combined, the 693762 embeddings were 1.82GB. Each cluster was scored by taking the average of the bot likelihood of each account in a cluster. Clusters with a bot likelihood of greater than .75 were selected to be presented to moderators in our user study.

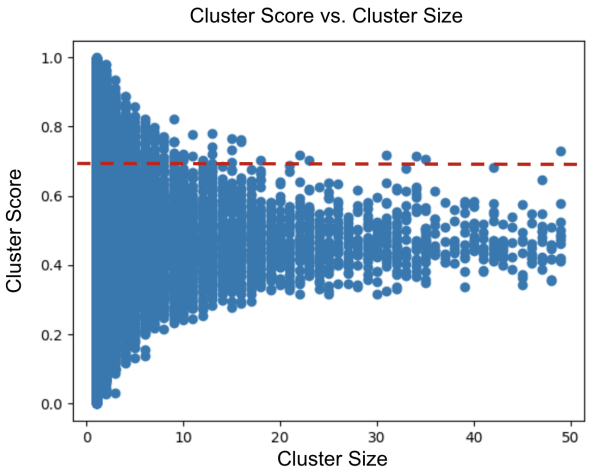

Figure 2: A scatter plot displaying the size and score of each cluster from hierarchical clustering. Accounts in clusters with cluster scores of greater than .75 are handed to moderators for label propagation.

5.5 User Study

Our five person user study was conducted to evaluate the effectiveness of our method and interface using the Twibot-22 dataset. Users were given 5 minutes to annotate as many clusters as possible. They were instructed to sample 5 accounts per cluster and used a TSNE visualization aid for selecting representative samples across the cluster. Results are shown in Table 2 from each of the 5 users and the mean accuracy.

	Labelled	Propogated	Clusters	Accuracy
User 1	10	78	2	0.6026
User 2	10	34	2	1.0
User 3	15	111	3	0.8829
User 4	20	141	4	0.6312
User 5	10	78	2	0.8718
Overall	65	442	13	0.7897

Table 2: Per-user metrics from the user study. Labelled is the number of accounts manually labelledl by the user, propogated is the total number of accounts labelled after propogation, clusters is the number of clusters, and accuracy is accuracy of all predicted/propogated labels.

6 CONCLUSION AND DISCUSSION

The user study confirmed the efficacy of our bot labelling system. Figure 3 highlights two key metrics for bot annotation - speed and accuracy. The majority of our users were able to achieve accuracies 10% higher than the most performant bot classifier, BotRGCN. Other than User 2, there is a clear trade-off between speed accuracy, which is expected in most systems with a human-in-the-loop. User 2's subpar performance is due to an error that resulted in an incorrect ordering of clusters.

Another important consideration for deploying a system like this for actual moderators is cost. The most expensive portion of our pipeline is clustering as it requires high RAM machines. Our costs were under $5 to cluster the Twibot-22 dataset, however clustering on hundreds of millions of accounts would require a very expensive machine or a different approach (discussed more in future work).

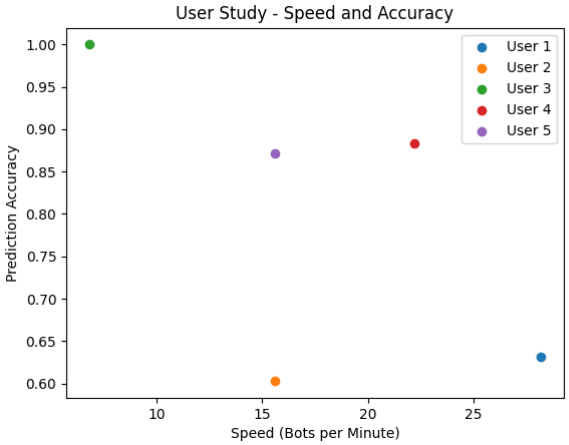

Figure 3: A scatter plot comparing bot detection speeds and accuracies for each user in the user study.

6.1 Limitations

There are a number of limitations in this study that leave room for future work. The size of the clusters and speed of bot detection scales with the embeddings input to the clustering algorithm. Two potential approaches, other than using machines with massive RAM, are hierarchical merging schemes to combine clusters after clustering on the partitioned embeddings , and sampling embeddings so only a subset of the embeddings are used for clustering, but all the embeddings are assigned cluster.

The order in which clusters are presented is an important design feature. Instead of using cluster size for ordering, a combination of cluster score and size could be user to order clusters.

Our user study could also be improved. Our users did not have the opportunity to use the system over a period of time. It is typical for users to get better at using systems over time. Establishing a learning curve before evaluating users may provide even better results.

6.2 Team Effort Statement

All team members have worked diligently and contributed a similar amount of effort.

REFERENCES

[1] Ashkan Dehghan, Kinga Siuta, Agata Skorupka, Akshat Dubey, Andrei Betlen, David Miller, Wei Xu, Bogumil Kaminski, and

Pawel Pralat. 2022. Detecting Bots in Social-Networks Using Node and Structural Embeddings. (2022).

[2] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1320–1329.

[3] David Dukić, Dominik Kea, and Dominik Stipic. 2020. Are You Human? Detecting Bots on Twitter Using BERT. *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)* (2020), 631–636.

[4] Michael Färber, Agon Qurdina, and Lule Ahmedi. 2019. Identifying Twitter Bots Using a Convolutional Neural Network. In *Conference and Labs of the Evaluation Forum*.

[5] Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, et al. 2022. TwiBot-22: Towards graph-based Twitter bot detection. *arXiv preprint arXiv:2206.04564* (2022).

[6] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. Twibot-20: A comprehensive twitter bot detection benchmark. (2021), 4485–4494.

[7] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. 2022. BotRGCN: Twitter Bot Detection with Relational Graph Convolutional Networks. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Virtual Event, Netherlands) *(ASONAM '21)*. Association for Computing Machinery, New York, NY, USA, 236–239.

[8] Emilio Ferrara. 2018. Measuring Social Spam and the Effect of Bots on Information Diffusion in Social Media. *Computational Social Sciences* 10 (2018), 229–255.

[9] Kadhim Hayawi, Sujith Samuel Mathew, Neethu Venugopal, Mohammad Mehedy Masud, and Pin-Han Ho. 2022. DeeProBot: a hybrid deep neural network model for social bot detection based on user profile data. *Social Network Analysis and Mining* 12 (2022).

[10] Samaneh Hosseini Moghaddam and Maghsoud Abbaspour. 2022. Friendship Preference: Scalable and Robust Category of Features for Social Bot Detection. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–1. https://doi.org/10.1109/TDSC.2022.3159007

[11] Sneha Kudugunta and Emilio Ferrara. 2018. Deep Neural Networks for Bot Detection. *ArXiv* abs/1802.04289 (2018).

[12] Dim P Papadopoulos, Ethan Weber, and Antonio Torralba. 2021. Scaling up instance annotation via label propagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 15364–15373.

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *Proceedings of the 2018 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 1–15.

[15] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. DALL·E 2: Exploring Cross-modal Embeddings for Image Generation. arXiv:2110.13147 [cs.CV]

[16] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[18] Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.

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

[20] Kai-Cheng Yang, Emilio Ferrara, and Filippo Menczer. 2022. Botometer 101: Social bot practicum for computational social scientists. *Journal of Computational Social Science* (2022), 1–18.