"Lockstep": An Ensemble Approach to Social Media Bot Behavior Analysis using Machine Learning

Proposal for an Ensemble Approach to Social Media Bot Behavior Analysis using Machine Learning

Hunter P. Williams

Author, Undergraduate Computer Science major at Georgia Southern University, hw07709@georgiasouthern.edu

Felix G. Hamza-Lup

Advisor, Georgia Southern University, fhamzalup@georgiasouthern.edu

Social media platforms dominate as key channels for interaction in the digital age, offering unparalleled convenience but also serving as powerful tools for influence and manipulation. This has led to the rise of machine-automated social profiles and increasingly complex interactions with human users. While a testament to human ingenuity, these interactions pose challenges to the authenticity of online experiences and raise ethical concerns. Current detection methods for social bots rely heavily on opaque, black-box machine learning models, sparking concerns about their interpretability. To address this, we introduce ***Lockstep****: an ensemble model leveraging engineered features and simpler, interpretable methods. Our novel approach outperforms state-of-the-art techniques in predicting social bot behavior, delivering superior accuracy and efficiency.*

Commented [FH1]: This is shorter and more powerful. Abstract needs to be very concise.

Keywords: Social Bots, Bot Detection, Machine Learning, Random Forest Classification, Ensemble Models, Graph Neural Networks

1. Introduction

The internet has become a vital platform for self-expression and communication, requiring minimal context to connect people over vast distances. Recognizing the influence of online social networks on community decision-making, businesses, organizations, and governments have sought to capitalize on this power. This demand has driven a surge in AI-generated social media accounts, or "bots," designed to mimic human interactions within online communities.

These bots now pervade every social media platform, performing tasks such as posting content, replying to conversations, interacting with users, and even building reputations within platform systems. Powered by advances in linguistics, statistics, and computer science, these sophisticated bots generate human-like conversations with remarkable ease. As the creation and deployment of such bots become increasingly accessible, they pose significant societal challenges and raise important ethical concerns. There is so much potential in having a *social bot* being able to participate in a social community. Where one *social bot* might post updates whenever somebody's favorite cricket team scores a goal, another bot engages in human-like conversations while advising their conversational partners on specialized topics that it might have been trained on.

Human users increasingly struggle to distinguish social bots from genuine interactions. Some platforms have even enabled or overlooked the widespread use of bots to inflate engagement metrics. These sophisticated, human-like bots remain largely unchecked, enabling harmful applications such as coordinated disinformation campaigns, election interference, and targeted harassment. A recent OpenAI report [1] highlights a staggering 900% year-over-year increase in AI-driven election disinformation and deepfake activity, underscoring the growing threat.

Even when not overtly malicious, bots contribute to the spread of misinformation, often lacking the nuance needed to present accurate context. Social media platforms, as key venues for news, interaction, and expression, bear a significant responsibility. Unchecked bot activity undermines the authenticity of online interactions, posing ethical challenges and influencing societal discourse in unintended ways.

To address these concerns, bot detection has emerged as a critical field, leveraging expert knowledge and statistical machine learning to identify behavioral anomalies. Recent advancements in neural networks have shown promise in uncovering complex patterns within behavior and relationship networks. This paper introduces a cost-efficient intermediary model, bridging traditional and deep-learning approaches to identify anomalous behavior before deeper, resource-intensive analysis.

Lockstep is an ensemble behavior analysis model for social bots. Success in the development of a social network will generally see a rapid growth in a user base and an even more explosive development of activity between users. The demand for scale in an environment like social media makes Lockstep an especially good contender as a starting point for moderation.

Lockstep was designed to analyze user behavior patterns across social media, focusing on profile details, submission metadata, and relationship dynamics within communities. Unlike traditional bot-detection approaches, Lockstep prioritizes interpretability, leveraging observable trends and explainable patterns to enhance generalization across diverse implementations. It combines a random forest classifier for profile analysis with a GraphSage-based shallow graph neural network (GNN) [9] to detect relationship trends within limited-depth networks. This streamlined, layered approach enhances usability, scalability, and performance, particularly when deployed against live data or larger, more diverse datasets. We hypothesize that this ensemble methodology, emphasizing simplicity and efficiency, offers a powerful and time-effective solution for large-scale, real-time social bot detection.

The paper is structured as follows: **Section 2** provides a concise literature review, contextualizing the field of social bot detection and highlighting current methodologies. **Section 3** introduces the Lockstep model, detailing the project environment and experimental setup. **Section 4** presents experimental results, comparing Lockstep's accuracy and effectiveness with prior models. **Section 5** concludes with a summary of findings, a discussion of limitations, and highlights of notable aspects of the experiment. Finally, **Section 6** lists references for the datasets, models, and literature cited throughout the study.

2. Background - Bots and Trends

2.1 Bots

If you've used Facebook or X in 2024, chances are you've encountered a bot—and if not, you soon will. A "bot," short for "software robot," is a program designed to automate tasks using algorithms. The concept dates back to the earliest days of computing, with chatbots first proposed in the 1950s by computing pioneer Alan Turing [11]. Notably, Joseph Weizenbaum's ELIZA [27], a 1960s chatbot simulating a psychologist, famously convinced many users unfamiliar with computers that they were interacting with a real person.

Modern social bots take this concept to the next level, combining advanced algorithms and intricate instructions to simulate human-like interactions online. While ELIZA could recall keywords, today's social bots can remember your name, interpret lengthy instructions, and engage in detailed conversations using your chat history for personalized context. Some social bots serve legitimate purposes, but many are used maliciously—to spread spam, execute phishing attacks, manipulate markets and communities, or inflate metrics to attract advertisers. [2] Lockstep focuses on addressing the growing deployment of these bots across modern social media platforms and the increasing need for effective tools to identify and mitigate their influence.

2.2 Trends

By 2017, 15% of the active userbase of Twitter was estimated to consist of social bot profiles. [28] In 2019, an IEEE Communications report deduced that about 11% of all Facebook accounts were created for automation. [29] Imperva, an organization devoted to internet-wide 'bad bot' trends and traffic analysis has consistently found that over the last three years (2021-23) almost half of all traffic on the internet was likely to fall into the category of a bot. [21]

Bot detection has become an area of increasing academic focus due to a highly correlated increase in observed bot activity and the amount freely available code designed to drive such activity. In 2019, there were over 40,000 active GitHub repositories dedicated to social bot development. [30] In 2024, a search for the terms 'twitter bot' code on the same platform returns over 26,000 results. Such rapid interest increase in these topics has marked the start of an "arms race" between bot developers who seek to refine their methodology to bypass detections and appear 'more human', and researchers who work to develop means of assisting moderation teams in detecting these attempts at mimicry with sophisticated and often costly models of behavior analysis. [16]

Why are these signs so dangerous? From spambots to chatbots, these programs that seek to mimic natural language have gotten much more believable. The introduction of models inspired by natural language processing has given social bots a sea of potential and capabilities outside text generation. They can be used in coordination to alter the public perception of a user's influence by 'inflating' metrics such as the number of likes and follows; They can run intimidation campaigns personalized with someone's publicly available information; they can operate as part of communities to influence public sentiment and opinion; They can steer conversations with a victim to a more intimate nature in attempt to reveal sensitive information, or use patterns identified in speech to attempt to masquerade as friends, or acquaintances. Just a few (2-4%) percentage points of bot interactions in a community can twist perceptions and alter decision making in a polarized. [31].

2.3 Bot Detection

A majority of bot detection techniques center around behavior. Bots, including social bots, do things that humans do not do – even when they are programmatically trying not to. Different types of bots exhibit different types of behavior, and much work has been done on distinguishing between the purposes of bots and identifying a complete taxonomy of their behaviors.

Some behavior is easier to spot and differentiate from human behavior: the ratios between certain public metrics of social profiles (like, follow, tweet count, etc.) have been found to be good starting indicators for social bot activity. [36] Account age may also be another easy way to add an additional filter: it can be assumed that very old accounts are less likely to be bot-controlled if not simply because the longer a bot is active the higher the likelihood of being caught. Another argument for this is that some accounts on social media that are older than a certain level of trend in using social bots, are less likely to be bots themselves.

Statistics like this are not an independently accurate method of filtering out a population of all bots: more modern bots will also employ camouflage and work in networks to fake public metrics that make them seem more 'human' to a detection system. As a consequence, many detection tools measure an ensemble of information.

Depending on the types of bots, the patterns in time for which these bots post content on a platform can differ in measurable ways from human equivalents. Previous research has identified several of these distinct time-based patterns: the diversity of times the bot is active in a week or within the span of a day, or the times of day at which a bot uses a URL or a hashtag, and the frequency of both occurrences. [26]

Additional work has been done on identifying patterns in the behavior of bots working together towards a goal through, through the analysis of social communities and directional interactions between entities. For every post, like, reply, quote, or retweet, these interactions between entities on a platform can be modeled on a directional graph. Analysis of these networks has given researchers more clues: bots operating synchronously will often be active at a time coinciding with the targeted communities or groups of bots might share some characteristics such as a group of profiles that are being followed or liked on all participating accounts. In general, bots have been found to interact with or attempt to be seen by profiles that have a high level of quantifiable influence in a community. [37]

Advances in machine learning, particularly in natural language processing (NLP) tasks, have allowed us to build powerful statistical models for generating meaningful text. Natural Language Processing (NLP) focuses on enabling computers to understand and process human language. Among recent advances, large language models (LLMs) and neural networks have captured public attention with convincingly human-like text and image generation capabilities controlled through a natural language prompt. LLMs are trained on vast datasets of human text and use layers of neural networks to try and predict the best next token in a sequence of tokens, given a map of semantic and positional information. Neural network models specialize in different tasks: Convolutional Neural Networks (CNNs) excel at image processing, Recurrent Neural Networks (RNNs) handle text and time-series data, and Graph Neural Networks (GNNs) analyze relationships in graph-structured data, such as social networks or object interactions.

2.3 Models

Traditional machine learning algorithms like Random Forests, Support Vector Machines (SVM), and Decision Trees are very effective in behavior analysis models. These algorithms are best used to recognize patterns within structured data such as user activity logs, profile metadata, or engagement metrics. Lockstep utilizes random two *forest classifiers*, which involves having many decision 'trees' operate as a 'forest' or in ensemble.

A **Decision Tree** is a model that makes predictions by splitting data into branches based on feature values. A **Random Forest Classifier** (RFC, or RF) model builds on this concept by utilizing and averaging the outcome or voting on the output of many trees. By doing this, complex interactions that aren't always linear can be captured even in higher-dimensional data. For example, a random forest might learn that bots often post with a much higher frequency than average human users during certain times of the day or that they tend to have unusual follower/following ratios when compared to the output of tweets.

Random forests can also provide estimates of feature importance, which can then be annotated to describe how a decision might be made based on a high or low weight given to a feature. The classification method that an RF model uses can vary resulting predictions. A usual random forest model will produce a final classification using a method of **Majority Voting,** simply choosing the winning classifier from the mode of all predictions made by the trees in the forest. **Averaging** takes the prediction probabilities from all of the trees and then averages them to decide on classification. Majority voting is often a choice when a small subset of outliers in a set of predicted probabilities are capable of causing the average in a set of mostly higher probabilities to output a classification closer to the smaller subset.

Commented [FH2]: This section should be shortened or removed … there is too much detail. We will discuss but when you send the paper you send it to a targeted conference so they know more on these terms….

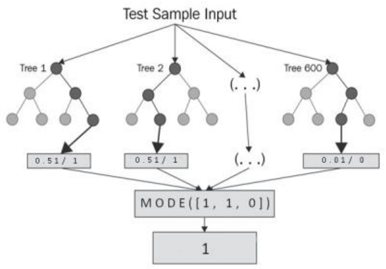


Figure 1: Random forest classification with majority vote. [23]

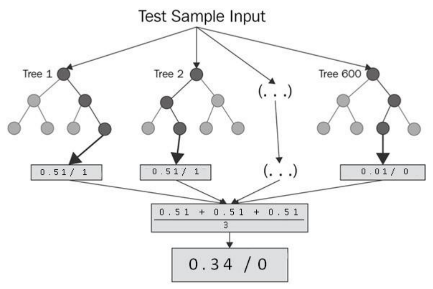


Figure 2: Random forest classification with averaging

3. Prominent Bot Detection Models

Several prominent bot detection models align with the goals of Lockstep but operate on a smaller scale. These models are analyzed for the methods and features they utilize to identify social bot profiles, highlighting components incorporated into the Lockstep framework during the experiment (see Table 1 for a summary).

Botometer [2014, 2016, 2018, 2020: (v1, v2, v3, v4)] (formerly known as BotOrNot) is a tool designed to assess the likelihood that a given Twitter account is automated (i.e., a bot) or operated by a human. It uses a combination of machine learning algorithms, social network analysis, and statistical modeling to determine the probability that an account exhibits bot-like behavior. Botometer v4 [2020] is an ensemble model utilizing random forests and embedded model features, much like the model that the *Lockstep* methodology proposes. [4] The original 2014 BotOrNot model operated on solely a random forest classifier with a high number of features. [32]

Botometer-X [2022] is built on the same framework as **Botometer-Lite**, which is in turn a more advanced and enhanced version of the original **Botometer**. It builds upon the same foundational principles of bot detection but introduces improvements in terms of model accuracy, flexibility, and broader applicability. Botometer-X was designed to tackle increasingly sophisticated social bots and adaptation to the changing dynamics of online behavior, particularly in the context of platforms like Twitter. [18]. Botometer-X is also an ensemble model like its predecessors – featuring the use of neural networks, including DNN and GNN to increase the accuracy of predictions in combination with other traditional algorithms. Botometer-X went into 'archival' mode as of 2023, partly due to the closing of access to the Twitter (now X) developer API. [10] [18]. All generations of Botometer have seen the use of ensemble models in different levels of complexity. Similarly, Lockstep aims to find an efficient balance between traditional algorithms and more complex neural network models.

BotBuster [2022] is a tool designed for detecting and mitigating the impact of social bots on social media platforms using Twitter as a basis for training. Like other tools such as Botometer, BotBuster's methodology analyzes patterns of behavior, content patterns, temporal patterns, and social network relationship features to identify the behavior profile of automated or fake accounts. Drawing similarities to previously described tools, BotBuster is also an ensemble-based model that divides analysis into several 'expert' models based on aspects of a user being analyzed. These 'expert' models include: 'Known Information', 'Username/Screenname/Description', 'User Metadata', 'Post Expert', 'Account-Level Post', and 'Post-Level Post'. The analysis from each expert has a level of independence from one-other, with some overlapping features. [5] The *Lockstep* framework seeks to operate on a more limited set of analyses with less overlap in features.

BotHunter [2018] is an ensemble model for detection that utilizes a combination of machine learning techniques to spot suspicious behavior indicative of bots. It's distinct difference between other ensembles lies in information tiers: BotHunter chooses the type of model(s) to use based upon a tier level that is assigned by the amount of information about the user that is available for analysis. The attributes assessed during the original experiment included text semantics at tier 0, account metadata at tier 1, and temporal patterns such as unusually high posting frequencies and very constrained follower-following counts, at tier 2. With the most information available at tier 3, network patterns and relationships were also analyzed for repetitive content and bot network interactions. The Lockstep model does not discriminate against the use of models based on available information and instead assumes a level of information availability respective for methodology. Lockstep also utilizes more raw and engineered features in each would-be tier to make predictions. [6]

BotMoe [2023] analyzes many of the same features that other models have, including metadata, textual content, and network structure. BotMoe is a multi-stage model with a transformer performing predictions with processed data from previous layers. The layer of models deemed 'encoders' transform raw features into representational features based on the domain of each expert in the mid-layer ***mixture of experts*** **(MOE)** divide and conquer scheme. The metadata encoding model utilizes a limited number of features from a user profile including, followers, followings, status, active days, and screen name length to generate a representational score. The textual encoder analyzes tweet content using word-vectors, pretrained language models, long short-term memory RNN models, and attention mechanisms to generate representative scores. The graph encoder used seeks to identify sub-communities based on user attributes, and then measures interactions between communities via message passing in order to create its representative scores. The scores from these encoding layers are then gated into communities based on a 'learnability' score of features and the vote of a number of experts for the community that each feature's value is associated with. These processed representations are then weighed by an 'expert fusion' layer based on the community that the user has been assigned to, and a final representative score is generated and used for self-attention and prediction in the final classification layer. BoeMoe's extensive utilization of deep neural networks and complicated methodology are drawbacks affecting cost-effectiveness to implement and the efficiency of real-time inferences. Lockstep improves on this with a more diverse set of features weighed by much simpler techniques that are better suited for real-time implementation.

Commented [FH3]: Try to make this section more concise and better integrated ... meaning it should flow better from one model to the next based on features etc...

Table 1: Methodology Comparisons

NAME	METHODOLOGY	MODELS USED	DISTINCT FEATURES	TARGET PLATFORM	TWIBOT-22 SCORE (F1)
BOTOMETER [2020]	Ensemble	RF	Ensemble of specialized classifiers (ESC)	Generalized	0.4257
BOTOMETER-X [2023]	Ensemble	DNN, GNN, RF	Limited Data, Selective Training	Generalized	*
BOTBUSTER	Ensemble w/ MOE	LLM, Transformer, Multi-Layer Perceptron	Mixture of Experts	Reddit, Twitter	0.5418
BOTHUNTER	Ensemble w/ Information Tiers	Support Vector Machine, DT, RF, Naive Bayes, Logistic Regression	Selective models based on information availability.	Twitter	0.2346
BOTMOE	Ensemble w/ MOE	GNN, LLM, Transformer, Multi-Layer Perceptron	Deeper analysis with MOE	Twitter	0.5662

* Botometer-X has not had published results testing against the dataset that Lockstep is trained on.

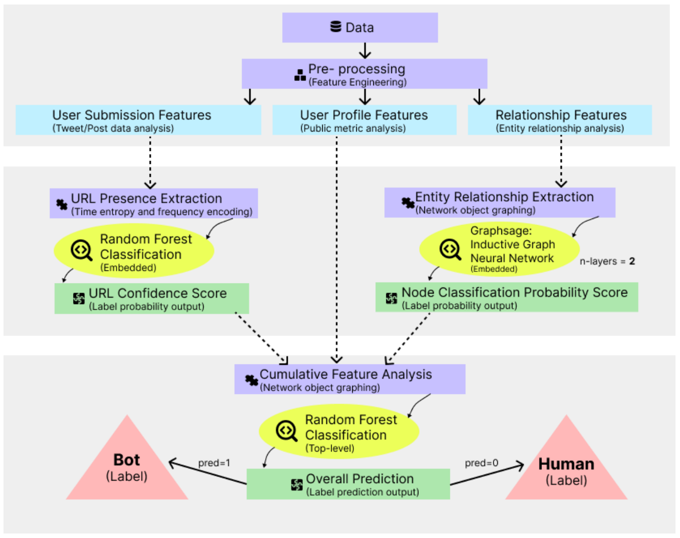
We can see recurring patterns from a brief analysis of popular options: ensemble models are generally accepted as a good architecture to evaluate bot behavior. There are also a particular set features analyzed by these models that overlap and appear in each methodology, and there is a lot of room for consideration in depth of analysis and range of models in reference to resulting scores.

4. Proposed Lockstep Model

4.1 Lockstep: The proposed ensemble model

Lockstep's strength lies in its simplicity and interpretability, focusing on shallow feature importance and deliberate overlap rather than deep pattern analysis. It employs a traditional Random Forest (RF) classifier with majority voting to make binary classifications based on 32 raw and engineered features. To enhance accuracy and discriminative ability, Lockstep integrates two additional models: a secondary RF classifier and a shallow Graph Neural Network

(GNN) based on the GraphSage architecture. These models refine predictions by improving AUC/ROC scoring and delivering higher precision/recall performance, even when individual features have low standalone predictive power. This ensemble approach ensures a balanced and robust evaluation.

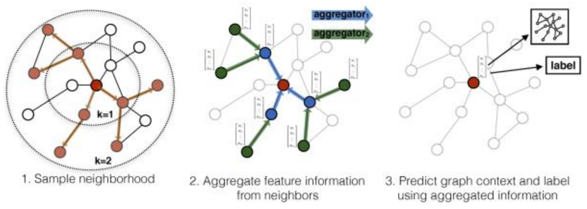


Data
Pre- processing
(Feature Engineering)
User Submission Features
(Tweet/Post data analysis)
User Profile Features
(Public metric analysis)
Relationship Features
(Entity relationship analysis)
URL Presence Extraction
(Time entropy and frequency encoding)
Random Forest Classification
(Embedded)
URL Confidence Score
(Label probability output)
Entity Relationship Extraction
(Network object graphing)
Graphsage: Inductive Graph Neural Network
(Embedded)
n-layers = 2
Node Classification Probability Score
(Label probability output)
Cumulative Feature Analysis
(Network object graphing)
Random Forest Classification
(Top-level)
Overall Prediction
(Label prediction output)
pred=1
pred=0
Bot
(Label)
Human
(Label)

Fig. 3 Proposed model ….

The decision to choose an RF classifier as the final predictive layer is based on its ability to identify thresholds within more complex interactions between features and true labels. Random forest classifiers have additionally been found to be resistant to over-fitting from set size and high dimensionality. [33] These attributes make RF classifiers well-suited for *Lockstep's* vision of behavior analysis, with special significance to the diversity of the training set which relates well to social platform communities observed in the real-world.

Lockstep selects GraphSage for node classification tasks based on similarities between the methods that GraphSage uses to sample a limited depth of neighbors in a target network, and how Lockstep selects a limited depth of entities and relationships to provide to the model for training. Lockstep does not expect a complete closed graph of the entire platform, and Graphsage better specializes in this type of constraint as part of a graph-based model.[9]



k=1
k=2
1. Sample neighborhood
aggregator1
aggregator2
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Figure 4: The GraphSage sampling approach.

4.2 Dataset and Sampling Methodology

For training, validation, and testing, Lockstep utilizes the resources provided by the Twibot-22 dataset.[8] Twibot-22 includes a robust set of public user profile fields, over 160,000,000 unidirectional relationships between network entities, and expert-annotated classifications of an account's authenticity for 1,000,000 users in total. Designed for social bot analytics and research, Twibot-22 surpasses prior benchmarks as one of the most comprehensive datasets currently available, improving upon its predecessor Twibot-20 in both scale and annotation quality. [Table 2] Collection techniques used to construct Twibot-22 are diverse with a low-to-moderate natural level of noise in terms of information availability and completeness. Techniques used to validate the classifications made in Twibot-22 include a range of model-based analysis and random-selection hand-annotation by human experts, detailed further in the original research paper. [8]

DATASET	**C-15**	**G-17**	**C-17**	**M-18**	**C-S-18**	**C-R-19**	**B-F-19**	**TWIBOT-20**	**TWIBOT-22**
USERS	5,301	2,484	14,368	50,538	13,276	693	518	229,500	1,000,000
BOTS	3,351	1,090	10,894	42,446	7,102	353	138	6,589	139,943
HUMANS	1,950	1,394	3,474	8,092	6,174	340	380	5,237	860,057
TWEETS	2,827,757	0	6,637,616	0	0	0	0	33,488,192	86,764,167
EDGES	7,086,134	0	6,637,616	0	0	0	0	33,716,171	170,185,937

Table 2: Dataset Comparisons

Twibot-22 has been extensively tested on a range of detection models. Performance scores for competing algorithms on the Twibot-22 dataset can be viewed in Table 4 and Figure 3. Scoring metrics for these models are further explained in section *VII*'s evaluation metrics. Many of these models report significantly higher F1 and AUC/ROC scores in their initial research, where datasets are carefully crafted and sampled to maximize the effectiveness of machine learning algorithms on anomalies. In contrast, Twibot-22 provides a dataset featuring a more diverse and unbalanced pool of users and features, at nearly a 9:1 ratio. This can be observed in Table 2. Twibot-22 does not discriminate on language features and textual factors, and subsequently includes data sourced from a wide array of languages, regions, and types of users. The range of information available per user is also limited based on relational depth away from original "starting seed" users. Information about these seed users and sampling methodology for TwiBot-22 is detailed in the original research paper. These challenges cumulatively make it difficult for statistical models to generalize well enough to maintain consistent performance when working at scale with Twibot-22. The decision to use Twibot-22 with Lockstep was driven by these difficulties: creating a more realistic information set that Lockstep's model is better capable of utilizing in more generalized applications.

For this experiment, sets of tests were conducted within trials based on the scale of **sampling** from Twibot-22's dataset. For each trial, the number of samples from the dataset was changed. In each test, constraints on samples evaluated based on availability of information were modified. In the case of both trial and test, the selection of sampled users from the dataset for each label type (bot = 1, human = 0) was randomized before selection for additional constraints. There are more human users than bot users in the Twibot-22 dataset, at a ratio of around 9:1. To help prevent scenarios in which a high score is achieved by guessing 1 label 100% of the time, sample sets are striated through minority oversampling to a 2(+-1):1 ratio between human and bot labels, respectively.

Table 3 shows the variables changed in each test and the sample sized used when testing.

TRIAL	**TEST**	**SET SIZE**	**MIN. POSTS**
1	1	20,000	10
	2	20,000	25
	3	20,000	50
2	1	100,000	10
	2	100,000	25

3	100,000	50

Table 3: Trial/Test scopes

It is necessary to have a minimum number of sampled posts to accurately score behavior in the overall model due to the features directly dependent on the existence of and information in a post's metadata. The best scores were achieved when there were 50 or more posts required for analysis, and the lowest when 10 posts were required.

Of the relationship edges included in the Twibot-22 dataset, representing interactions between users, tweets, lists, and hashtags – we select a limited number of nodes to analyze per user based on the distance of a relationship from a targeted user. These interactions are each represented on a single line storing a unidirectional relationship, include following/followers, posts, pins, likes, mentions, retweets, quotes, replies, list ownership, list membership, list follows, tweets contained in lists, and hashtags contained in tweets. For example:

Line 1:	**u00001 follows u00002**
Line 2:	u00012 like t00000222
Line 3:	u00002 post t00000222

To avoid sampling the entire relationship network for resource implications and to increase time-efficiency, a predetermined number of recursive scans are made over the relationship edge file to collect a limited 'depth' of neighbor node and relationships from each target user in a training set. For all tests, this N-depth has been limited to 3. Figure 5 illustrates the process of building the network around a single target user in the trial set.

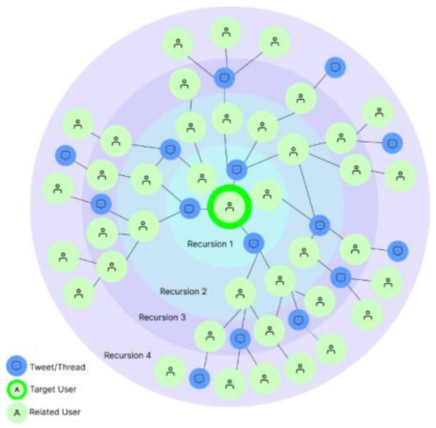


Figure 5: Entity aggregation for graphing

4.3 Data Pre-processing

Preprocessing steps are performed on the raw features that Twibot-22 dataset provides to reduce the time taken between iterations of tests. During deployment, these steps could be performed in-place in memory. To assist the model in making predictions, some raw features from the dataset are engineered into multiple features. Other features that could create noise or are outside of the scope of this project are removed before being used for evaluation.

4.3.1 User Profile Information

The information provided about each user by Twibot-22 includes raw features for: user identification strings (user ID), username, date of creation, profile description, the voluntarily set name listed on a profile, the display URL of a profile, the location set on the profile, the public metrics of a profile (such as number of likes, followers, people following), metadata entities (relating to the presence of data that includes embedded information, such as a URL in a post that is assigned a twitter URL shortener link or for a text with a hyperlink to a URL, or a hashtag that is assigned an integer ID for URL patterns), the pinned tweet on the profile, the profile image and its location on the internet, protected account status, and verification status.

From the text elements of a user profile, we extract the length of usernames and profile descriptions before dropping the feature for analysis. Data analysis on the entire has seen a trend of the profile descriptions of users labelled as bots being *shorter* than their human counterparts. This trend can be viewed in Figure 8. From a profile's public metrics, we add features for the ratio of user followers to users following for a profile, and the ratio of user tweets to user posts. These simple features have shown significant correlation in the final model, seen in Figure 8.

4.3.2 Submission Information

From a user's tweet submissions, we engineer the following features: total retweets of all posts, total likes on all posts, total replies to a post, an average of all three of these metrics, the ratio of posts that have zero likes or retweets to those that have one or more, calculated entropy scores for the day of a week and hour of a day that the user posts, the average time between posts, and the chi distance or similarity between a Benford's law normal distribution and the distribution of likes on posts and retweets of posts from a user.

Bots that have been purposed for advertisement of commercial sites or for engagement tend to post URLs from the same domains often. It's been observed that bots tend to either have a rigid schedule or pattern related to when posts are made, or no schedule at all and a history of posting around the clock. For each URL that is mentioned in a user's sampled post history, e.g. in a tweet, the URL confidence sub model uses the frequency of its occurrence and the calculated diversity (entropy) of when a URL appears as a feature influencing the output probability score.

Commented [FH4]: This should be normal text .... Also you have x and phy ... you need to explain this more clear.

Commented [HW5R4]: I simply removed it. It was mostly fluff anyway, designed to look pretty.

5. Evaluation Metrics

Commented [FH6]: Don't you want to move Evaluation Metrics before Evaluation ?

Commented [HW7R6]: Sure. It works.

To measure how well Lockstep identifies bots in the dataset, we used two main metrics: the **F1 score** and the **AUC/ROC** score. These metrics give insight into the model's accuracy, precision, and strength of classification.

F1 Score

The **F1 score** balances precision (how many detected bots are actual bots) and recall (how many bots are correctly detected). It is particularly useful in situations where there might be an imbalance between bot and human accounts. Precision and recall are represented by the equation:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

F1 scoring then combines these metrics:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In practical terms, a high F1 score means that the model is accurately identifying bots without mistakenly classifying too many human accounts as bots.

Aera Under the Receiver Operating Characteristic Curve (AUC/ROC)

The **AUC/ROC score** helps evaluate the model's ability to distinguish between bot and human accounts over different classification thresholds. The ROC curve compares the true positive rate (sensitivity) with the false positive rate. The AUC, or area under this curve, summarizes this ability: an AUC close to 1 indicates strong discrimination between bots and humans, while an AUC near 0.5 suggests random guessing. We calculate the true positive rate and false positive rate as follows:

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}}$$

Using both F1 and AUC/ROC provides a more comprehensive evaluation. The F1 score shows the balance between precision and recall, while the AUC/ROC score illustrates the model's performance across various detection thresholds, ensuring that Lockstep is effective and adaptable for different use cases.

What does a high score mean?

In the context of the bot detection model, high F1 and AUC/ROC scores serve as key indicators of strong performance. An F1 score approaching 1.0 demonstrates a well-balanced trade-off between precision and recall, indicating the model's ability to accurately identify bots while minimizing both false positives (misclassified human accounts) and false negatives (undetected bots). Similarly, an **AUC/ROC score** close to 1.0 indicates that the model effectively distinguishes between bot and human accounts across various classification thresholds.

Moderate scores can still provide meaningful insights and offer value in analysis. The embedded models used in the architecture do not perform as well as the entire model. These models do, however, provide significant weight for an overall a better scoring response when used as confidence scores. One might not need perfect classification, especially if our goal is to prioritize high-probability bot accounts for further review rather than outright removal, which is the intention of Lockstep. A statistical model should never be 100% of the decision-making power in such an action. A moderate F1 score (e.g., between 0.6 and 0.8) can still effectively identify many bots, particularly in high-volume environments like social media platforms, where complete accuracy is challenging to achieve.

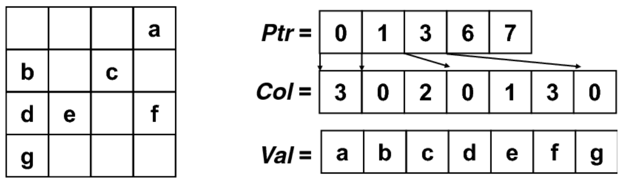
6. Evaluation

Lockstep evaluates based on 58 total metrics. As depicted in [Figure 1], the top-level random forest model utilizes the probability of two embedded models included in these features during evaluation: a random forest for predictions based a vector containing URL appearance time entropy and frequencies for all domains that are present in all submissions from users within a set, and a shallow graph neural network utilizing the *GraphSage* methodology as seen in the paper 'Inductive Representation Learning on Large Graphs' to evaluate relationships between neighboring relationships as indicators.

5.1 URL Confidence with Random Forest Classification

From the URL entropy/frequency features described by section III.D, a compressed sparse row (CSR) matrix indexed with the features *z* in the shape (x, y) is created, wherein x represents the total count of users, *y* represents the length of the complete list of domains collected as *y*, and *z* as the list of features we collect for each domain. The three features we collect for each domain include the total frequency of appearances, the entropy of hour of the day a URL is posted, and the entropy of the day of a week a URL is posted. This sparse matrix is pruned based on a minimum frequency that a URL must appear to be weighted, then split into a test/train set and the properties paired with the true labels of associated users are used as features in a random forest classification model for training and probability output for weighting.

Figure 6: CSR matrix format [34].



The random forest model is tuned with the parameters: *number of estimators = 1, minimum samples leaf = 2, maximum features = sqrt, max tree depth = 30,* a *balanced* class weight, and *entropy-based* criterion.

Formatted: Font: 10 pt

After training and fitting, the resulting fit is used to generate a *URL Confidence* score for all users being classified. The *URL Confidence* score feature is generated by averaging the predictive probability output from. This *probability* is used in the final layer of prediction as a weight for Lockstep's model.

$$URL\ Confidence = \ Majority\ Vote\ Probability = \ \hat{y}(x) = \frac{\underset{c}{arg\ max} \sum_{t=1}^{T} 1(\hat{y}_t(x) = c)}{number\ of\ estimators}$$

Formatted ...

Formatted: Font: 10 pt

'GraphSAGE' Node Classification

GraphSAGE uses inductive learning, neighbor sampling, and feature aggregation to provide a means of node classification for incomplete subsets of very large graphs made. Unlike transductive approaches, which require retraining when new nodes are added, GraphSAGE generates embeddings for unseen nodes by aggregating the features of their local neighborhoods.

The graph G = (V, E) is represented by V is the set of entities in our relationship network, and E as the unidirectional edges generated from the relationships between entities. In *k* layers:

$$[\ \mathrm{h}_{\mathrm{N}}^{k}(v) \ = \ \mathrm{AGGREGATE}^{(\mathrm{k})} \left(\left\{ \mathrm{h}_{\mathrm{u}}^{(\mathrm{k}-1)} : \mathrm{u} \ \in \ \mathcal{N}(\mathrm{v}) \ \right\} \right) \ ]$$

Formatted ...

For every node *v* ∈ V, a sample of neighboring nodes *N(v)* and their edges are taken. For each sampled node, features are aggregated from neighbor nodes along with embeddings from a previous layer *k*, represented by $h_{u}^{(k-1)}$. Node embeddings for *v* are generated by combining the node's own embedding value with aggregated neighbor embeddings to compute a new representation:

Formatted ...

Formatted ...

$$\mathrm{h}_{\mathrm{v}}^{k} \ = \sigma \left(W^{k} * CONCAT \left(h_{v}^{k-1}, h_{N}^{k}(v) \right) \right)$$

Formatted ...

Where W^k is a set of weights, and σ is the rectified linear unit (ReLU) activation function.

Formatted ...

From the total number of entities extracted in all recursions over the entire dataset of relationships, 50% are used for training, 20% for validation, and 30% for testing. The *GraphSage* model is run with the hyperparameters, *dropout*: 0.3, *sample_size:* [15,10], *hidden_size:*[128], *num_layers:* 2, and *aggr(gation method): mean*.

Predictive probabilities for each classification are saved after pre-processing and the positive bot-label classification is used as a weight in the top-level classification model.

Formatted: Justified

Top-level Random Forest Classification

With the included positive prediction probability outputs from both sub-models, these weights and the remainder of pre-processed and raw features are scaled through min-max normalization methods. Outliers are not dropped.

The top-level classification model is a random forest classification model with the tuned hyperparameters: *max tree depth: 40, no max features, no maximum leaf nodes, no minimum impurity decreases, minimum samples per leaf: 2, minimum samples in a split: 15, number of estimators = 500, no class weight* and evaluation based on Gini co-efficient criterion. There is no further logic performed on the prediction output by the top-level model, and a classification of either ‘0’ for human or ‘1’ for bot is made.

Commented [FH8]: We need to discuss on this section ... needs a lot of corrections ... we will set a time to work on this together.

Commented [HW9R8]: My vision for this section was a ‘how do the models in this architecture work’. It would be boring to say that we take all the previous features and feed it into this model that sits there and guesses at coefficients until the number gets big. The coolest thing this project really does is using that stupid URL confidence thingy. We are going to try and avoid the word unique outside of these models haven’t been smashed together in this way yet.

Do I just... leave it out?

7. Hardware and Platform

All experiments are run on a locally hosted headless Ubuntu 22.04.05 LTS virtual machine on a Jupyter notebook hosted Python 3.12.4 kernel. Our computational resources included one Tesla P40 graphics accelerator, 6/12 x Intel(R) Xeon(R) CPU E5-1650 v3 cores @ 3.50GHz, and 40gb of available DDR4-ECC memory.

Our random forest models were provided within the open source *scikit-learn* Python library @ scikit-learn.org. The graph neural network was an implementation of the GraphSage methodology as seen in the paper at [9], made possible by the *CogDL0.6* Python toolkit for deep learning models on graphs. 12] We used *PyTorch* as an engine for training and inference on the implemented mode [r2].

8. Results

Each of the six tests is run five times, and the average score across these runs is used as the final score for each variant. The overall average score across all experiments serves as the basis for comparison with similar models. Details on sampling and constraints for trial and test combinations are provided in Table 3 within the methodology section. For benchmarking, we include a top-level decision tree model alongside the primary random forest classifier. The scores for each test variant combination are summarized in Table 4. The results are further compared against a publicly available benchmark dataset, Table 5, maintained by the creators of the TwiBot-22 dataset. This benchmark and related information can be accessed via the dataset’s GitHub repository [15].

The results for each test in each experiment are below. An F1 and AUC/ROC score is provided for each top-level model used, and an F1 score is provided for the embedded models. The emboldened scores are for the primary chosen top-level model for this project. The italicized and emboldened scores are those used for showcasing.

Table 4: Results for each experiment variant.

Test Combinations			**URL Confi. Model**	**GNN Model**	**Top-level Random Forest**		**Top-Level Decision Tree**	
Trial:	# Users	# Min. Posts	F1	F1	**F1**	**AUC/ROC**	F1	AUC/ROC
1	20,000	10	0.619784294	0.6974	**0.774965507**	**0.845513682**	0.67611455	0.671617157
	20,000	25	0.690221242	0.6974	**0.770548094**	**0.846511651**	0.690221242	0.684948723
	20,000	50	0.644292527	0.6974	***0.811495249***	***0.873978613***	0.743424468	0.715753782
2	100,000	10	0.605928959	0.7216	**0.76483415**	**0.842718087**	0.670623509	0.666765887
	100,000	25	0.604568964	0.7216	**0.77098033**	**0.846982435**	0.669508999	0.661805674
	100,000	50	0.649511997	0.7216	**0.788226885**	**0.848692144**	0.706577432	0.673531711
				Average:	**0.780175036**	**0.850732768**		

Baseline Comparison

The results for each test are averaged and this score used as a baseline in evaluating the model. In all results listed here, the competitor models are run with on same dataset that this experiment uses, Twibot-22. References for these can be found in a separate area in the references section. [15]

TABLE 5: COMPETITOR EXPERIMENTS ON THE SAME DATA. REFERENCES IN VI-A

NAME	Accuracy	F1	Type
NAME	Accuracy	F1	Type
NAMEBOT [R1]	0.7061	0.0050	Logistic Regression
EVOLVEBOT [R2]	0.7109	0.1409	random forest
ROBERTA [R3]	0.7207	0.2053	RoBERTa
BGSRD [R4]	0.7188	0.2114	BERT GAT
BOT HUNTER [R5]	0.7279	0.2346	random forest
HAYAWI ET AL. [R6]	0.7650	0.2474	lstm
EFTHIMION ET AL. [R7]	0.7408	0.2758	efthimion
KOUVELA ET AL. [R8]	0.7644	0.3003	random forest
LEE ET AL. [R9]	0.7628	0.3041	random forest
MOGHADDAM ET AL. [R10]	0.7378	0.3207	random forest
SGBOT [R11]	0.7508	0.3659	random forest
KNAUTH ET AL. [R12]	0.7125	0.3709	random forest
ALHOSSEINI ET AL. [R13]	0.4772	0.3810	gcn
LOBO [R14]	0.7570	0.3857	random forest
HGT [R15]	0.7491	0.3960	GNN
BOTOMETER [R16]	0.4987	0.4257	Ensemble
RGT [R17]	0.7647	0.4294	GNN
MILLER ET AL. [R18]	0.3037	0.4529	k means
SIMPLEHGN [R19]	0.7672	0.4544	Graph Neural Networks
KUDUGUNTA ET AL. [R20]	0.6587	0.5167	SMOTENN, random forest
ABREU ET AL. [R21]	0.7066	0.5344	RANDOM FOREST
WEI ET AL. [R22]	0.7020	0.5360	Ensemble
BOTBUSTER [R28]	0.7406	0.5418	LLM, Transformer, MLP
KIPF ET AL. [R23]	0.7839	0.5496	Graph Neural Network
VELICKOVIC ET AL. [R24]	0.7948	0.5586	
RODRIGUEZ-RUIZ [R25]	0.4936	0.5657	SVM
BOTMOE [R27]	0.7925	0.5662	Transformer, MOE
BOTRGCN [R26]	0.7966	0.5750	BotRGCN
LOCKSTEP	**0.7916**	**0.7802**	**Ensemble**

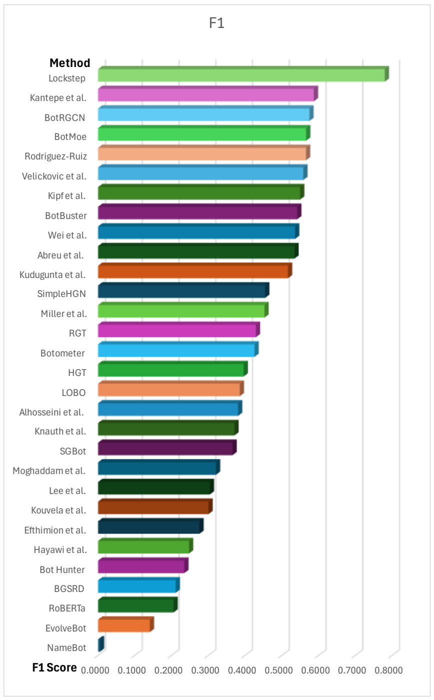


Figure 7: F1 Scores comparison bar chart

As a foreword before closing this section, there are some considerations to make for the sibling methodologies here.

- These methodologies may have been trained on a subset of the database, or the whole dataset. Lockstep has not been tested on its performance while utilizing the entire dataset as a training sample, and instead randomly sampled 20,000 or 100,000 users for 30 different tests.
- No verification was performed on whether every sample had been seen once.
- Models may or may not striate their sampling to increase the balance of classification examples they are trained on, and if they do, they may do so to different extents. As stated in the methodology section, Lockstep balances sample sets that are used to train with a 2(+-1):1 ratio between human labels and bot labels.

Lockstep shows improvement.

The scores and comparison indicate that our model is performing well versus its peers on the TwiBot-22 dataset. While there are clear indications that there is still room for improvement and refinement of this methodology, it shows early success on cutting the complexity of behavior analysis models in favor of using ensembles of better interpretable models. An F1 score of 0.80 classifies this model as high performance in terms of classification complexity. With more fine-tuning and research, we strongly believe that a .90 or higher is in the future for this methodology limited to the model.

9. Discussions

We predicted that some features might have more weight in the final decision of the top-level forest, and our original goal set out to use interpretable models like the RF classifier model- so during training we analyzed the individual branch weights that our top-level RF classifier was fit to. Common indicators of social bot activity are well understood to include unusual follower-to-following ratios, high posting frequency, repetitive patterns in posting, and URL posting patterns for similar sources [3] [25]. Much of our findings corroborate this, however we found interesting traits that correlate better than random guesses when analyzed for patterns:

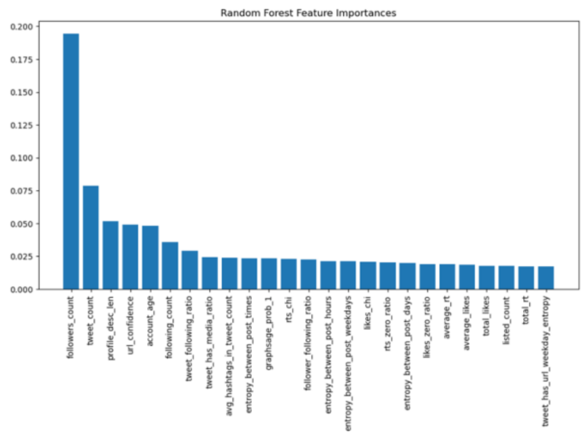


Figure 8: Feature significance representations for the top-level random classifier fit.

There are some noteworthy things here in [Figure 8]:

- Followers count has an outsized amount of influence on the classification with a bot level. Tweet count also seems to have an outsized amount of influence on classification, although not nearly to the same extent. This corroborates there being a strong relationship between bot behavior and unusual public metric ratios.
- The URL confidence model we use is an especially significant weight in predicting a bot label. This might be an area of interest in future research.
- The graphsage_prob_1 score is significant and may be an are of interest in the future.
- Profile description length seems to have identifiable patterns between labels that allow them to be weighted heavily in this model. After analysis, it seems that on average, the description in accounts detected as bots is actually *shorter* than that of human labelled accounts.

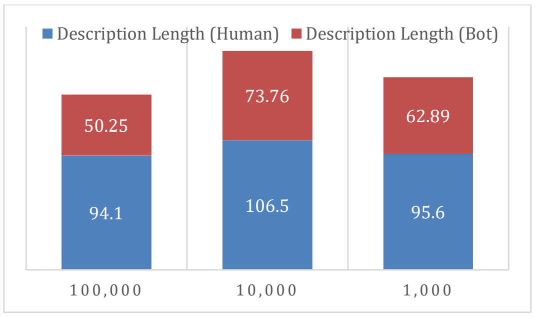


Figure 9: Description length in Twibot-22 for each sample size

- Account age is a significant factor in classification. This is understandable given that many accounts used to perform ‘botting’ are generally new accounts that are removed after detection, go unused once the initial ‘campaign’ is over, or are otherwise deleted after use. Even if not for this, one might surmise that some accounts are simply old enough to have outsized themselves over some point in the historical social bot trendlines.

Limitations

Availability of Data: The highest F1 and AUC/ROC scores are won by the tests that have a higher number of samples available in their tweet history available. While the model is still relatively well performing even with a reduced number of samples, having no samples at all is not expected to be a situation in which this model can accurately predict behavior.

Interpretation of Predictions: A positive prediction that Lockstep makes does not definitively mean that a user is a bot, but that this model finds high confidence in a prediction based on feature similarities between the test and the training set.

Dataset Age: The data that Lockstep is trained on was collected in 2022 and 2023. While this is still a recent collection in comparison to other datasets, it means that the trends displayed by users during these dates and the abnormalities that these models detected on this experiment may more strongly correlate to this time period. Introducing live data to this with a large enough discrepancy might generate noise in trying to get the most accurate prediction.

Commented [FH10]: We’ll see where this fits later

10. Conclusion

Lockstep demonstrates significant improvements in performance metrics compared to sibling models on the same dataset, consistently outperforming them across various testing scenarios. Its architecture also shows better scalability, maintaining performance with minimal loss even as testing complexity increases—a distinct advantage over other approaches. This research aims to advance tools for moderating harmful social bot behavior, where the stakes for misinformation, manipulation, and abuse are high. By achieving superior results without reliance on costly computational resources, Lockstep validates the principle that effective first-line defenses against social bots can be both efficient and accessible. While future models leveraging high computational power and complex features may achieve incremental gains, the ability to deliver robust, cost-effective performance at scale remains essential for fostering secure and authentic online communities. Lockstep’s success marks an important step toward that goal.

References

[1] Hayden Field. 2024. OpenAI says bad actors are using its platform to disrupt elections, but with little “viral engagement.” (October 2024). Retrieved November 18, 2024 from https://www.cnbc.com/2024/10/09/openai-says-more-cyber-actors-using-its-platform-to-disrupt-elections.html

[2] Cloudflare. What is a social media bot? | social media bot definition. Retrieved November 18, 2024 from https://www.cloudflare.com/learning/bots/what-is-a-social-media-bot/

[3] Microsoft 365. 2024. How to spot bots on social media. (June 2024). Retrieved November 18, 2024 from https://www.microsoft.com/en-us/microsoft-365-life-hacks/privacy-and-safety/how-to-spot-bots-on-social-media

[4] M. Sayyadiharikandeh, O. Varol, K.-C. Yang, A. Flammini, and F. Menczer, "Detection of Novel Social Bots by Ensembles of Specialized Classifiers," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, in CIKM '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 2725–2732. doi: 10.1145/3340531.3412698.

[5] Lynnette Hui Xian Ng and Kathleen M. Carley. 2022. BotBuster: Multi-platform bot detection using a mixture of experts. (July 2022). Retrieved November 18, 2024 from https://arxiv.org/abs/2207.13658

[6] David M. Beskow and Kathleen M. Carley. 2018. Bot-hunter: A Tiered Approach to Detecting & Characterizing Automated Activity on Twitter. (2018). Retrieved November 18, 2024 from http://www.casos.cs.cmu.edu/publications/papers/LB_5.pdf

[7] Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. 2019. RTbust: Exploiting temporal patterns for botnet detection on Twitter. (February 2019). Retrieved November 18, 2024 from https://arxiv.org/abs/1902.04506

[8] Shangbin Feng et al. 2023. TwiBot-22: Towards graph-based Twitter bot detection. (February 2023). Retrieved November 18, 2024 from https://arxiv.org/abs/2206.04564

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs. (September 2018). Retrieved November 18, 2024 from https://arxiv.org/abs/1706.02216

[10] Botometer x. Retrieved November 19, 2024a from https://botometer.osome.iu.edu/

I1] A.M. Turing. 1950. I.-Computing machinery and intelligence. (October 1950) ,433–460. https://doi.org/10.1093/mind/LIX.236.433

[12] Yukuo Cen et al. 2023. COGDL: A comprehensive library for graph deep learning. (April 2023). Retrieved November 18, 2024 from https://arxiv.org/abs/2103.00959

[13] Emilio Ferrara et al. 2016. The rise of Social Bots. (June 2016). *Commun. ACM*, vol. 59, no. 7, pp. 96–104. https://doi.org/10.1145/2818717

[14] PyTorch. Retrieved November 18, 2024 from https://pytorch.org/

[15] Luoundergradxjtu/TwiBot-22. Retrieved November 18, 2024b from https://github.com/LuoUndergradXJTU/TwiBot-22

[16] Kai-Cheng Yang et al. 2023. Social Bots: Detection and challenges. (December 2023). Retrieved November 19, 2024 from https://arxiv.org/abs/2312.17423

[17] Yang Zhang. 2019. Language in our time: An empirical analysis of hashtags. (May 2019). Retrieved November 19, 2024 from https://doi.org/10.48550/arXiv.1905.04590

[18] Yang, K.-C., Varol, O., Hui, P.-M., & Menczer, F. (2020). Scalable and Generalizable Social Bot Detection through Data Selection. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), 1096-1103. https://doi.org/10.1609/aaai.v34i01.5460

[19] Joseph, Sethunya et al. 2016. Natural Language Processing: A Review, 6(03) , 207-210. https://www.researchgate.net/publication/309210149_Natural_Language_Processing_A_Review

[20] Aran Komatsuzaki. 2020. Current limitations of language models: What you need is retrieval. (September 2020). Retrieved November 19, 2024 from https://doi.org/10.48550/arXiv.2009.06857

[21] Bad Bot Report. (September 2024). Retrieved November 19, 2024 from https://www.imperva.com/resources/resource-library/reports/2024-bad-bot-report/

[22] Wayne Xin Zhao et al. 2024. A survey of large language models. (October 2024). Retrieved November 19, 2024 from https://arxiv.org/abs/2303.18223

[23] Sanjib Ghosh. Comparing Regular Random Forest Model with Weighted Random Forest Model for Classification Problem (2024). Retrieved November 19, 2024 from https://www.researchgate.net/publication/344038267_Improved_Weighted_Random_Forest_for_Classification_Problems

[24] Lamija Lemes and Amila Akagic. 2022. Prediction of Real Estate Market Prices with Regression Algorithms. Retrieved November 19, 2024 from https://www.researchgate.net/publication/364542925_Prediction_of_Real_Estate_Market_Prices_with_Regression_Algorithms

[25] Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023. SEQXGPT: Sentence-level AI-generated text detection. (December 2023). Retrieved November 19, 2024 from https://arxiv.org/abs/2310.08903

[26] Richard J. Oentaryo, Arinto Murdopo, Philips K. Prasetyo, and Ee-Peng Lim. 1970. On profiling bots in social media. (January 1970). Retrieved November 19, 2024 from https://link.springer.com/chapter/10.1007/978-3-319-47880-7_6

[27] "ELIZA—a computer program for the study of natural language communication between man and machine | Communications of the ACM." Accessed: Nov. 20, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/365153.365168

[28] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online Human-Bot Interactions: Detection, Estimation, and Characterization," Mar. 27, 2017, *arXiv*: arXiv:1703.03107. doi: 10.48550/arXiv.1703.03107.

[29] M. Zago *et al.*, "Screening Out Social Bots Interference: Are There Any Silver Bullets?," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 98–104, Aug. 2019, doi: 10.1109/MCOM.2019.1800520.

[30] D. Assenmacher, L. Clever, L. Frischlich, T. Quandt, H. Trautmann, and C. Grimme, "Demystifying Social Bots: On the Intelligence of Automated Social Media Actors," *Social Media + Society*, vol. 6, no. 3, p. 2056305120939264, Jul. 2020, doi: 10.1177/2056305120939264.

[31] B. Ross, L. Pilz, B. Cabrera, F. Brachten, G. Neubaum, and S. Stieglitz, "Are social bots a real threat? An agent-based model of the spiral of silence to analyse the impact of manipulative actors in social networks," *European Journal of Information Systems*, vol. 28, no. 4, pp. 394–412, Jul. 2019, doi: 10.1080/0960085X.2018.1560920.

[32] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "BotOrNot: A System to Evaluate Social Bots," in *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*, Montr&#233;al, Qu&#233;bec, Canada: ACM Press, 2016, pp. 273–274. doi: 10.1145/2872518.2889302.

[33] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random Forests and Decision Trees," vol. 9, no. 5, 2012.

[34] "A sparse matrix and its CSR format," ResearchGate. Accessed: Nov. 26, 2024. [Online]. Available: https://www.researchgate.net/figure/A-sparse-matrix-and-its-CSR-format_fig1_330071609

[35] "Chi-Square Distance," in *The Concise Encyclopedia of Statistics*, New York, NY: Springer, 2008, pp. 68–70. doi: 10.1007/978-0-387-32833-1_53.

A. Competitor Evaluation Method References

Commented [FH11]: Assuming you will clean this up ....

Commented [HW12R11]: In what manner? I'm kind of scared of it. I see that one is missing and that those are multiple spaces and not tabs. I suppose a final pass over it is inevitable though. Can I say I didn't do it? Cause that's mostly true.

[R1] David M. Beskow and Kathleen M. Carley. 2018. Its all in a name: Detecting and labeling bots by their name - computational and mathematical organization theory. 24–35. https://doi.org/10.1007/s10588-018-09290-1

[R2] Chao Yang, Robert Harkreader, Guofei Gu. 2013. Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. IEEE 8,8 (Aug. 2013), 1280–1293. https://doi.org/10.1109/TIFS.2013.2267732

[R3] Yinhan Liu et al. 2019a. Roberta: A robustly optimized Bert pretraining approach. (July 2019). Retrieved November 19, 2024 from https://arxiv.org/abs/1907.11692

[R4] Qinglang Guo, Haiyong Xie, Yangyang Li, Wen Ma, and Chao Zhang. 2021. Social Bots detection via fusing bert and graph convolutional networks. (December 2021). Retrieved November 19, 2024 from https://www.mdpi.com/2073-8994/14/1/30

[R5] Kenny, R., Fischhoff, B., Davis, A., & Canfield, C. (2024). Improving Social Bot Detection Through Aid and Training. Human Factors, 66(10), 2323-2344. https://doi.org/10.1177/00187208231210145

[R6] Hayawi, K., Mathew, S., Venugopal, N. et al. DeeProBot: a hybrid deep neural network model for social bot detection based on user profile data. Soc. Netw. Anal. Min. 12, 43 (2022). https://doi.org/10.1007/s13278-022-00869-w

[R7] Efthimion, Phillip George; Payne, Scott; and Proferes, Nicholas (2018) "Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bots," SMU Data Science Review 1, 2. https://scholar.smu.edu/datasciencereview/vol1/iss2/5

[R8] Maria Kouvela, Ilias Dimitriadis, Athena Vakali, Aristotle University of Thessaloniki Maria KouvelaInformatics Department, Aristotle University of Thessaloniki Ilias DimitriadisInformatics Department, and Aristotle University of Thessaloniki Athena VakaliInformatics Department. 2020. Bot-detective: Proceedings of the 12th International Conference on management of Digital Ecosystems. (November 2020). Retrieved November 19, 2024 from https://dl.acm.org/doi/10.1145/3415958.3433075

[R9] Kyumin Lee, Brian Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on Twitter. Retrieved November 19, 2024 from https://ojs.aaai.org/index.php/ICWSM/article/view/14106

[R10] Samaneh H. Moghaddam, Maghsoud Abbaspour, Friendship Preference: Scalable and Robust Category of Features for Social Bot Detection. *IEEE Transactions on Dependable and Secure Computing*, 20, 2(March. 2023) , 1516–1528. https://doi.org/10.1109/TDSC.2022.3159007.

[R11] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. 2019. Scalable and generalizable social bot detection through data selection. (November 2019). Retrieved November 19, 2024 from https://arxiv.org/abs/1911.09179

[R12] Jurgen Knauth. 2019. Language-Agnostic Twitter-Bot Detection. Proceedings of the International Conference on Recent Advances in Natural Language Processing (Sep. 2019), 550–558. https://dl.acm.org/doi/abs/10.1145/3308560.3316504

[R14] Juan Echeverrï£¡a et al. 2018 . LOBO: Evaluation of Generalization Deficiencies in Twitter Bot Classifiers. ACSAC (2018), 137–146. https://doi.org/10.1145/3274694.3274738

[R15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph Transformer. (March 2020). Retrieved November 19, 2024 from https://arxiv.org/abs/2003.01332

[R16] Mohsen Sayyadiharikandeh, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, Filippo Menczer. 2020. Detection of Novel Social Bots by Ensembles of Specialized Classifiers. ACM (Oct. 2020), 2725–2732. https://doi.org/10.1145/3340531.3412698.[R17] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo. 2021. Heterogeneity-aware Twitter bot detection with relational graph Transformers. (December 2021). Retrieved November 19, 2024 from https://arxiv.org/abs/2109.02927

[R18] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, Alex Hai Wang. 2024. Twitter spammer detection using data stream clustering. Information Sciences: an International Journal 260 (March 2014). 64 - 73 https://dl.acm.org/doi/10.1016/j.ins.2013.11.016

[R19] Qingsong Lv et al. 2021. Are we really making much progress? revisiting, benchmarking, and refining heterogeneous graph neural networks. (December 2021). Retrieved November 19, 2024 from https://arxiv.org/abs/2112.14936

[R20] Sneha Kudugunta and Emilio Ferrara. 2018. Deep Neural Networks for BOT detection. (February 2018). Retrieved November 19, 2024 from https://arxiv.org/abs/1802.04289

[R21] Jefferson Abreu; Célia Ralha; João Gondim. 2020. Twitter Bot Detection with Reduced Feature Set. *IEEE (*Nov. 2020), 1–6. https://doi.org/10.1109/ISI49825.2020.9280525.

[R22] Feng Wei and Uyen Trang Nguyen. 2020. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. (February 2020). Retrieved November 19, 2024 from https://arxiv.org/abs/2002.01336

[R23] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph Convolutional Networks. (February 2017). Retrieved November 19, 2024 from https://arxiv.org/abs/1609.02907

[R24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. (February 2018). Retrieved November 19, 2024 from https://arxiv.org/abs/1710.10903

[R25] Jorge R. Ruiz, Javier I. Mata-Sánchez , Raúl Monroy , Octavio Loyola-González, Armando López-Cuevas. 2020. A one-class classification approach for bot detection on Twitter. *Computers & Security* 91 (April 2020). https://doi.org/10.1016/j.cose.2020.101715

[R26] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. 2021. BOTRGCN: Twitter bot detection with relational graph convolutional networks. (September 2021). Retrieved November 19, 2024 from https://arxiv.org/abs/2106.13092

[R27] Y. Liu, Z. Tan, H. Wang, S. Feng, Q. Zheng, and M. Luo, "BotMoE: Twitter Bot Detection with Community-Aware Mixtures of Modal-Specific Experts," Apr. 25, 2023, *arXiv*: arXiv:2304.06280. doi: 10.48550/arXiv.2304.06280.

[R28] Lynnette Hui Xian Ng and Kathleen M Carley. 2022. BotBuster: Multi-platform Bot Detection Using A Mixture of Experts. arXiv preprint arXiv:2207.13658 (2022)