BotBuster: A Mixture of Experts Approach towards Social Bot Detection

Anonymous Submission

Abstract

Reliable social bot detection is crucial for maintaining the health of online conversations. Bots often manipulate online conversations as secondary spreaders and amplifiers for misinformation sent by low-credibility news site and state sponsored media. Despite rapid development, current bot detection models still face challenges in dealing with incomplete data and cross-platform applications. In this paper, we propose BotBuster, a stable social bot detection architecture built with the concept of a mixture of experts approach. BotBuster is trained individually using each input information type (i.e., user name, screen name etc), which then performs separate predictions corresponding to the available information, and finally combines the information to output a bot probability. Experiments on several Twitter datasets show that BotBuster outperforms popular bot-detection baselines ($\bar{F}1 = 73.54$ vs $\bar{F}1 = 45.12$). This is accompanied with F1=60.04 on a constructed Reddit dataset and F1=60.92 on an external evaluation set. Our analysis shows that only 21 posts is required for a stable bot classification.

Introduction

Social media bots are accounts that are partially or fully controlled by algorithms. Bots have been known to influence online narratives (Bovet and Makse 2019), promote activism (Ng and Carley 2021) and polarize the online conversation (Stella, Ferrara, and De Domenico 2018). Bot detection is thus a crucial step for influence campaigns analyses. Social bot detection have been extensively studied (Orabi et al. 2020) and many machine learning frameworks have sprung up to address this problem.

Despite this progress, two important issues remain: dealing with incomplete data and a multi-platform bot detector. Bot detectors rely on account features from content information to user metadata to network information to perform a prediction. However, many times during fast-moving events like elections or protests, data collection on accounts is often incomplete. It is near impossible to perform an extensive collection of all fields required by any given bot detection algorithm, especially when there are at least a million accounts. Other times, it is simply impossible to collect all required data: the account may have been suspended or turned protected, forcing analysts to rely on available data, or previously collected historical data. A popular method to fill incomplete data is to make missing values zero, but that may affect the prediction as zero or null are valid values that do occur in the data. Additionally, most bot detectors are currently tuned for the Twitter platform, leaving other social media platforms vulnerable.

In this work, we advance the problem of bot detection modules through an architecture that can handle incomplete account information. Given an account's information of user name, screen name, account description, user metadata, user posts, where some information in this set may be missing, our goal is to classify the account into one of two classes: bot, human. We proposed solution, **BotBuster**, aims to overcome the limitations of requiring the entire set of account information for effective bot prediction [1].

Contributions Our contributions include:

1. As a technical contribution, we introduce the concept of a mixture of experts approach to bot prediction. A successful bot detection approach overcomes the limitation of requiring data from all features and makes a decent prediction given the available data. The proposed BotBuster reaches state-of-the-art performance ($\bar{F}1$=72.23), outperforming baselines ($\bar{F}1$=49.61, 40.63).
2. We extend bot prediction from one platform to multiple platforms, incorporating bot prediction across Twitter and Reddit. F1-scores for Twitter ($\bar{F}1$=73.54) and Reddit (F1=60.04) for our final model, and an accuracy of F1=60.92 on an external validation set shows the robustness of the model.
3. Finally, we analyze the stability of our proposed bot-detection model, lending recommendations for at least 36 posts for a minimal amount of data for collection.

Related Work

Bot Detection Bot detection is a notorious problem. Common supervised Twitter bot-detection mechanisms can be divided into two groups: (1) feature-based, using account details such as tweet frequency (Yang et al. 2020), tweet content (Beskow and Carley 2018), temporal features

[1] Code and links to datasets are available on github-repo-removed-for-anonymity

(Chavoshi, Hamooni, and Mueen 2016) and network features (Yang et al. 2019); and graph-based, which uses an account's communication for bot classification (Cao et al. 2012; Magelinski, Beskow, and Carley 2020). Two popular libraries are Botometer (Varol et al. 2017) and BotHunter (Beskow and Carley 2018). Botometer uses a supervised ensemble classification based on user, tweet and network features extracted for each account. BotHunter classifies accounts using a multi-tiered random forest method, with each tier making use of more features as before, from content to user to network features. However, Botometer queries Twitter live, meaning it is unable to work with archived data, and BotHunter requires the full feature set for prediction.

Bot detection has been widely studied on Twitter; it is less so on Reddit. Past work includes exploiting temporal and network information to detect political bots (Ferraz Costa et al. 2015; Hurtado, Ray, and Marculescu 2019). While there have been research on cross-platform analysis of social bot trends (Mittos et al. 2020), to the best of our knowledge, there has not been a bot prediction model that successfully combines bot prediction for Twitter and Reddit platforms.

Mixture of Experts Mixture of experts models have been used in ensemble learning (Caruana et al. 2004): for translation (Peng et al. 2020), sequence learning (Shazeer et al. 2017) and text generation (Shen et al. 2019). A close cousin is multi-task learning, where there are shared parameters among the tasks (Ruder 2017). This approach has been used in stance detection (Li and Caragea 2021) and offensive language detection (Benton, Mitchell, and Hovy 2017; Djandji et al. 2020). To the best of our knowledge, no bot detection approach has harnessed this model for bot detection. We adapt these ideas from the natural language community for the social bot detection task.

BotBuster Architecture

Figure 1 illustrates our proposed BotBuster architecture.

Given the set of user account information: X ={user name, screen name, description, user metadata, posts}, we first construct a set of $n = 5$ experts $\{\mathbf{f}_i(E_i; \boldsymbol{\theta}_i)\}, i \in X$, one for each information type, returning a predicted probability by passing the hidden embedding layer of each expert $\boldsymbol{H}_i$ through a sigmoid function $\mathbf{S}(\cdot)$. That is:

$$\begin{aligned} \hat{\boldsymbol{p}}_i &= \mathbf{P}(\text{bot}|\boldsymbol{X}_i) \\ &= \mathbf{S}(\boldsymbol{H}_i) \\ &= \mathbf{f}_i(\mathbf{E}_i; \boldsymbol{\theta}_i) \\ &= \mathbf{f}_i(\mathbf{E}_i(\boldsymbol{X}_i; \psi)) \end{aligned} \tag{1}$$

where $\mathbf{f}_i(\cdot; \boldsymbol{\theta}_i)$ is a parameterized function with an expert-specific input representation $\boldsymbol{\theta}_i$, a vector embedding representing inputs for the expert, generated using function $\mathbf{E}$.

To weight each of the n experts, we learn a gating network using the mean of $\boldsymbol{H}_i$. A prediction P(bot) is calculated from the ensemble of weights and expert probabilities.

During the prediction phase, not all elements in X may be present. BotBuster makes use of the available information to activate the corresponding experts. The experts are grouped by fields that are commonly retrieved together; should one of the fields the expert requires be absent, the expert is not activated. Finally, it outputs a score $\mathbf{P}(\text{bot}) \in [0, 1]$, where 1 is an indication of a bot and 0 an indication of a human.

Known Information Expert

The Known Information Expert (KIE) is the first gate which evaluates whether a user is a bot or not based on definite known information. If the signals from these definite known information are not present, the collected user data processes through the rest of the BotBuster architecture. Figure 1 depicts this element in the bottom left corner as a brown box.

During prediction, if it this expert is activated, $\mathbf{P}(\text{bot}) = \mathbf{f}(\boldsymbol{X}_{i=\text{KIE}})$ and all other experts are not activated. The expert are activated under the following two conditions: (1)$\mathbf{P}_{bot}$=0 if the "is_verified" field is True for Twitter agents, and (2) $\mathbf{P}_{bot}$=1 if the agent has the word "bot" in the user name or screen name (e.g. "xxUpdateBot"). Although the final probability of the account has been determined through these definite signals, posts and user information are still include in the training dataset to teach the other experts to differentiate between bots and human accounts.

User name, Screen name, Description Experts

The user name, screen name and description experts are modelled similarly and trained on their respective data. The experts $\mathbf{f}_i, i \in$ {user name, screen name, description} are parameterized with embeddings from a fine-tuned BERT language model.

The embedding function $\mathbf{E}_i$ is parameterized with sequence embeddings from the default tokenizer of the BERT language model (Devlin et al. 2019). The function $\mathbf{f}_i$ extracts BERT embeddings from $\boldsymbol{E}_i$ using a pre-trained BERT model, producing a 768-dimensional vector. The 768-d vector is then fed into a Multi-Layer Perceptron (MLP) network for fine tuning, which outputs a 64-d hidden layer representation $\boldsymbol{H}_i$. $\boldsymbol{H}_i$ is fed into a sigmoid function $\mathbf{S}(\boldsymbol{H}_i)$ to obtain $\hat{\boldsymbol{p}}_i$ and is used for the gating network.

User Metadata Expert

$\mathbf{f}_{user\_metadata}$ is parameterised using a Multi-Layer Perceptron with a sigmoid output. User metadata (e.g. number of followers, number of followings) is a set of numeric values that are normalized across the dataset and constructed into a vector with $\mathbf{E}_{user\_metadata}$. $\mathbf{E}_{user\_metadata}$ is fed into a 4-layer MLP, which feeds into a dense layer, returning a 64-d layer representation $\boldsymbol{H}_{user\_metadata}$. We obtain $\hat{\boldsymbol{p}}_{user\_metadata} = \mathbf{S}(\boldsymbol{H}_{user\_metadata})$ and use $\boldsymbol{H}_{user\_metadata}$ for the gating network.

Posts Expert

We test two types of posts experts: account-level post expert and post-level post expert.

Post Pre-Processing We first filter for posts that are origin posts, i.e. posts that are not quotes or retweets. This step gives us an insight into the post information and linguistic style originating from the agent. We then remove any

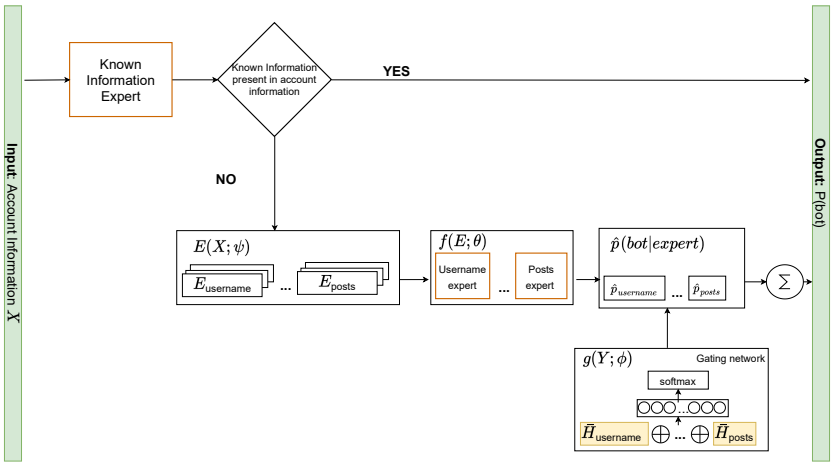

Figure 1: Diagram of the BotBuster Architecture.

hashtags, @-mentions, URLs and stop words. We derive linguistic values (i.e., sentiment) from the post text as features. These are described in further detail in *Feature Engineering*.

Input Representation A post consists of two main portions: post texts and posts metadata (e.g., retweet count, like count, and derived linguistic values). For post texts, we tokenize sentences using the default tokenizer of the BERT language model to obtain sequence embeddings using $\mathbf{E}_{post\_text}$ (Devlin et al. 2019).

For post metadata, we first construct a normalized float vector $\boldsymbol{V}_{extracted\_metadata}$ of extracted post metadata. We also construct a normalized float vector $\boldsymbol{V}_{derived\_metadata}$ of derived post linguistic values. We then obtain $\mathbf{E}_{post\_metadata} = \boldsymbol{V}_{derived\_metadata} \oplus \boldsymbol{V}_{extracted\_metadata}$

Account-Level Post Expert In this variation, we construct two sub-post experts: $\mathbf{f}_{post,text}(\cdot)$ and $\mathbf{f}_{post,metadata}(\cdot)$, representing post texts and post metadata (i.e., retweet count, like count etc).

For $\mathbf{f}_{post,text}(\cdot)$, we feed $\mathbf{E}_{post\_text}$ in a BERT model and fine-tuned it with a dense layer to obtain $\boldsymbol{H}_{post\_text}$. For $\mathbf{f}_{post,metadata}(\cdot)$, we feed $E_{post\_metadata}$ into a 4-layer MLP followed by a dense layer to obtain $\boldsymbol{H}_{post\_metadata}$.

We combine the output of the sub-post experts in a uniform manner:

$$\hat{\boldsymbol{p}}_{post} = 0.5 \times \mathbf{f}_{text} + 0.5 \times \mathbf{f}_{numeric} \tag{2}$$

The outputs of the two sub-post experts are concatenated for use in the gating network: $\boldsymbol{H}_{post} = \boldsymbol{H}_{post\_text} \oplus \boldsymbol{H}_{post\_metadata}$, and we obtain the post prediction $\hat{\boldsymbol{p}}_{posts} = \mathbf{S}(\boldsymbol{H}_{post})$.

Post-Level Post Expert The post-level post expert is a joint model in a Siamese network structure. We first feed $\mathbf{E}_{post\_text}$ on a BERT model and fine-tuned it with a dense layer to obtain $\boldsymbol{EB}_{post}$. We next feed $\mathbf{E}_{post\_metadata}$ into a 4-layer MLP to obtain $EB_{post\_metadata}$. We concatenate the outputs: $\boldsymbol{EB}_{post} = \boldsymbol{EB}_{post\_text} \oplus \boldsymbol{EB}_{post\_metadata}$. We then trained a dense layer on top, outputting a 64-d $\boldsymbol{H}_{posts}$ to predict $\hat{\boldsymbol{p}}_{post} = \mathbf{S}(\boldsymbol{H}_{post})$.

For each type of post expert, we experimented with and without the inclusion of derived post linguistic values as numeric features. We do not construct a mixture that combines both types of post experts to prevent duplicate information. The account-level experts aggregates information from multiple posts as input to train the model, while the post-level expert uses information from each post singularly. Combining both post experts will thus use the information from each post twice, once individually and once in an aggregate fashion. We thus reflect on the comparison between both types of post experts in the results.

Gating Network for Expert Importance

We adapt the approach from Peng et al. (2020) to weight experts for a combined ensemble model. BotBuster learns an input-dependent parameterised gating function $g(;\phi)$ and outputs $\mathbf{P}(\text{bot})$. More formally, for a given account information X and the hidden layer embeddings H, BotBuster is represented in Equation 3.

$$\mathbf{P}(\text{bot}) = \sum_{i=1}^{n} \mathbf{g}_i(\boldsymbol{Y};\boldsymbol{\phi}) \cdot \mathbf{f}_i(\boldsymbol{X}_i;\theta_i), \quad \text{where } \boldsymbol{Y} = \bar{\boldsymbol{H}}_1 \oplus \bar{\boldsymbol{H}}_2 .. \oplus \bar{\boldsymbol{H}}_n \tag{3}$$

$\mathbf{g}(\cdot;\boldsymbol{\theta})$ is parameterized with a MLP followed by a softmax layer. It first takes in the 64-d hidden layer embeddings $\boldsymbol{H}_i$ from $\mathbf{f}_i(\cdot)$ and averages it along the row (i.e., the sequence direction), producing one $\bar{\boldsymbol{H}}_i$ per data point. The resultant $\bar{\boldsymbol{H}}_{i..n}$ are concatenated together into $\boldsymbol{Y}$ which is fed into a two-layer MLP, interspersed with *tanh* activation layers. The *softmax* layer normalizes the n layer weights to output a $n-$dimensional vector that sums to 1, indicating the relative strengths of the experts. It can be seen as a learned probability distribution over the experts $f_{i..n}$, assigning more responsibility to the experts that are more important in bot detection, allowing them to contribute more. As elements in set $\boldsymbol{X}$ may be empty, we train the gating network to produce $|\boldsymbol{X}|! = 120$ combinations.

Datasets

To train and test our model, we collect 10 Twitter public datasets from 2017 to 2020. These datasets are human annotated with bot/human labels and are publicly available on a bot repository [2]. The datasets range from accounts collected from the US 2018 midterm elections to stock market bots to bots that were manually discovered on Twitter, covering a wide range of bot activities. We hydrate these accounts using the Twitter V2 REST API in July 2021, collecting user information and the latest 40 tweets per user. The bot repository provides a user data file of some user information (i.e., user name, screen name...) of the accounts at the time of dataset creation. For some accounts, we are unable to retrieve information as they have been suspended by Twitter or became a protected account. To fill in the data gaps, where possible, we use the bot repository's user data file to enrich the account information.

We also construct an additional dataset for Reddit. Unlike Twitter accounts, user information for Reddit accounts only consists of user names, providing us without information for screen name, posts and user metadata. We select a list of 500 bots that has the highest "bad bot" ranking in BotRank[3]. BotRank is a bot that relies on the Reddit community's exposure of bots and provides a good/bad bot ranking based on community approval (Trujillo et al. 2021). For Reddit humans, we collected a list of users from 5 subreddits that generally require conscious writing and thought and manually verified the users are likely to be humans. We then use the Pushshift Reddit API to collect user information and the latest 20 posts of each account's timeline. Table 2 provides a dataset overview and indicates the fields with incomplete data. An ontology map unifies the field names to account for different naming conventions across social media platforms.

Experiments

Baseline Comparisons

For each expert, we perform baseline comparisons of our neural network based formulation against common classifiers. For the text-based classifiers like user name, screen name, description and posts experts, we used a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer as a preprocessing step before fitting the data into the classifier. For the numeric data such as user metadata expert, we formulate the numeric data into a vector and normalize each numeric category before fitting the data into the classifier. For post experts that make use of the post text and derived values, we concatenate the TF-IDF vector with a vector of post derived values.

We perform baseline comparisons to two commonly used commercial bot-detection algorithms: Botometer (Varol et al. 2017) and BotHunter (Beskow and Carley 2018). Both algorithms provide a bot prediction score $\boldsymbol{P}(\text{bot}) \in [0, 1]$. These algorithms have been continually updated since their development in 2017-18 and have been widely used in social bot detection studies.

[2]http://botometer.org/botrepository

[3]http://botrank.pastimes.eu

For Botometer, we queried the online API to retrieve the universal score returned as $\boldsymbol{P}(\text{bot})$. We set a threshold of 0.5 for bot classification (i.e., $\boldsymbol{P}(\text{bot}) \geq 0.5 = \text{bot}$, as suggested by its authors. (Varol et al. 2017) For BotHunter, we obtained its library from its authors to perform bot detection. We set a threshold of 0.7 for bot classification (i.e., $\boldsymbol{P}(\text{bot}) \geq 0.7 = \text{bot}$ (Ng, Robertson, and Carley 2021). We report the results of these baseline predictions for each dataset where it exists (i.e. the baseline algorithms return results). We also report the average of these baseline predictions across all datasets.

Experimental Setup

In order for the BotBuster to learn from the diverse dataset acquired, we used a merged training strategy by training a single model on all the training data. Our model is implemented with Tensorflow. For training and hyperparameter tuning of each expert, we select accounts that have a value for that expert. We partition the entire dataset into 80:10:10 train:validation:test ratio. The dataset presents a class imbalance – a huge proportion of bots compared to humans, so we use the stratified sampling partition to ensure there is an equal proportion of each type of account in each set. After individual expert training, we perform joint training for the gated network, which represents the weights of each expert. In the joint training setup, we partition the dataset similarly into 80:10:10 ratio using the stratified sampling technique.

For all experiments, we use ADAM as the learning optimization with a learning rate of 0.001. We use 20 for the number of epochs and 32 for batch size. For the user name, screen name, description, user metadata and account-level post experts, we use the binary cross-entropy loss. For the post-level post expert, we use the binary cross-entropy loss from logits, in order to back propagate the gradients to the individual posts.

Evaluation

In the evaluation phase, we use the fine-tuned BotBuster model to perform predictions. With BotBuster, we classify accounts as a bot when $\boldsymbol{P}(\text{bot}|\text{information}) \geq 0.50$ and a human when $\boldsymbol{P}(\text{bot}|\text{information}) < 0.50$, stemming from the use of a binary classification model.

Individual dataset analysis In this phase, we train and test the BotBuster model on each dataset one by one. Each dataset is partitioned into a 80:10:10 train:validation:test stratified split and trained singularly using the same architecture. We compare the predicted classification against the original bot/human annotations and report the results as **BotBuster-singular**.

Individual dataset analysis with combined data We also perform individual dataset analysis with a combined data setup for training. In the combined data setup, we aggregate the data instances from all datasets. During each dataset run, we partition 10% of the targeted dataset for testing. We partition the remaining combined data into 90% for training and 10% for validation, accounting for the distribution of bots/humans in the combined dataset. We use the remaining combined data for training. We compare the predicted

classification against the original bot/human annotations and report the results as **BotBuster-Full**.

We opted for this strategy as it mimics the baseline bot detection algorithms. These algorithms are continually updated by training on diverse datasets, including those selected in this work. As we cannot retrain these state-of-the-art models on separate train-test data subsets, although we would love to, we adopt a setup where we train on data from all datasets and test on data from each dataset separately.

Evaluation against baseline returns The two baseline algorithms do not return results for all queried accounts. Botometer requires the account to currently still be active as it retrieves its data online; while BotHunter requires the presence of all the data it uses as features, or otherwise set to null fields. We thus collect $\boldsymbol{P}(\text{bot})$ for accounts the algorithms returned without modifying any data fields. Using the set of accounts in which the baselines returned $\boldsymbol{P}(\text{bot})$, we construct two sets of **BotBuster-Subset** based on the respective returns. We then compare our approach against the results returned by the baselines.

Evaluation against external dataset We perform an external validation by predicting bot probability using the BotBuster model on an unknown dataset. We run the BotBuster-4 prediction model on the TwiBot-20 test data set, consisting of 1183 users (Feng et al. 2021). This dataset was collected and annotated in 2020, whereas the other datasets were collected up to 2019. This evaluation helps us ascertain the robustness of the architecture towards newer bot behavior and an unknown set of annotations. For BotBuster, we used the tweets and user data provided in the dataset. We also ran the dataset through the baseline algorithms. For the Botometer baseline algorithm, we did not use the historical data provided; the API does an online pull of account data at the point of query.

Evaluation Metrics The micro-F1 score is adopted to evaluate the performance of our models. The micro-F1 score is preferred to give the same importance to each sample, accounting for the class imbalance situation. We ran each experiment three times and report the mean F1-score.

Feature Engineering

We perform feature engineering to exploit the account's features for the BotBuster architecture. Supplementary Table 1 lists the features used in the BotBuster model.

For BotBuster-2 and BotBuster-3 additionally used derived post linguistic values as features, which are linguistic characterization of the post texts. Past studies observed bots used simpler language than humans (Uyheng, Ng, and Carley 2021) and differ in the use of pronouns. We implement the Flesch-Kincaid reading difficulty score to characterize the text. This reading score is standardized by the U.S. Military and is used to gauge the ease of readability of a text. It is calculated as: $0.39 \times$ average words per sentence + $11.8 \times$ average syllabus per word - 15.59. We use the Pysentimento library (Pérez, Giudici, and Luque 2021) to perform sentiment analysis. The library fine-tunes a BERT-embedding based network on a Twitter dataset annotated with positive, negative and neutral sentiments. The Linguistic Inquiry and Word Count (LIWC) library (Tausczik and Pennebaker 2010) is a lexicon that summarizes the emotional, cognitive, and structural components in a given text sample. We use the 2015 version and focus on the time, affect, social, drives and pronouns components. We extract affective social identities through EPA scores, represented via three values: Evaluation (good vs. bad), Potency (strong vs. weak) and Activity (active vs. passive). These scores are extracted through a dictionary developed from surveys from 2012-2014 (Smith-Lovin et al. 2016).

Models

We perform experiments on four variations of the BotBuster architecture. In all variations, the structure of the user name, screen name, description and user metadata experts remain the same; we mainly vary the post expert. We vary the usage of an account-level post expert and a post-level post expert, and experimented on the inclusion of derived post linguistic values. In summary, the four model variations are:

1. **BotBuster-1**: account information experts + account-level post expert
2. **BotBuster-2**: account information experts + account-level post expert with derived post linguistic values
3. **BotBuster-3**: account information experts + post-level post expert
4. **BotBuster-4**: account information experts + post-level post expert with derived post linguistic values

Results

Training of Each Expert

We first trained each expert separately on the accounts that have values present and present the F1 accuracy scores in Table 1. We identify that the BotBuster formulation performs better than traditional classifiers. We observe that the inclusion of derived posts values increases accuracy and that the post-level post expert performs better than the account level expert. Individual experts perform as well as the ensemble results, showing that each expert can make a fairly accurate prediction of $P(\text{bot})$.

Individual Dataset Analysis

Percentage Analyzed In comparison to the BotHunter and Botometer baseline algorithms, BotBuster is able to analyze 100% of the accounts (Table 2), and additionally analyzes Reddit accounts. BotBuster analyzes $\boldsymbol{P}(bot)$ based on the available information, resulting in a weighted sum of probabilities for prediction. In contrast, BotHunter requires the presence of the entire feature set it is trained upon, while Botometer fetches information from Twitter in real-time, hence suspended accounts will return no results despite having previously data collected.

Expert Importance Appendix Tables 6 summarize the expert weights derived by the gating network for each BotBuster variation where all account information is present

Expert	Decision tree	Support Vector Classifier	Random Forest	BotBuster-formulation
User name	54.96	54.96	54.96	62.01
Screen name	59.70	60.79	59.56	62.01
Description	63.19	60.79	60.79	69.33
User metadata	44.52	55.63	55.63	62.38
Posts (account-level)	61.48	59.34	59.34	64.22
Posts (post-level)	59.22	63.59	63.32	64.79
Posts (post-level) + derived values	67.44	71.25	72.15	74.50
Posts (account-level) + derived values	68.74	68.65	61.81	72.98

Table 1: Macro F1 accuracy scores of individual experts.

Dataset	BotHunter	Botometer	BotBuster
astroturf	27.65	34.02	100
botometer-feedback-2019	61.44	71.07	100
botwiki-2019	90.34	92.90	100
cresci-rtbust-2019	74.97	78.78	100
cresci-stock-2018	40.57	47.03	100
gilani-2017	72.39	0	100
midterm-2018	11.26	1.31	100
political-bots-2019	0	20.60	100
varol-2017	83.17	72.28	100
verified-2019	88.60	98.15	100
reddit	0	0	100
Average	55.04± 31.32	51.61± 34.49	100

Table 2: Percentage of dataset analyzed by each algorithm.

with an additional breakdown for combinations for the best performing variation (BotBuster-4). A higher value reflects a higher importance for the expert. The gating network places a disproportionate emphasis on post information where the post expert is calculated by account level (BotBuster-1/2). For the best scoring variation (BotBuster-4), the weights are evenly distributed across the experts, suggesting an almost equal importance on each expert for final prediction. Experiments with BotBuster-3/4 where experts are equally weighted (i.e., $\boldsymbol{g}_i = \frac{1}{|\boldsymbol{X}|}$) fares with $F1 = 60.72 \pm 32.25$ compared to the weighted variation of $F1 = 72.23 \pm 18.84$. Hence, differentiation of experts is needed.

All variations place an almost equal weights on user name and screen name, which are usually very similar to each other. User names and screen names are single words. The average user name length is 3.02 ± 5.15 characters, and the average screen name length is 3.28 ± 6.23 characters. We compared the difference between the two strings using the Levenshtein distance. The average Levenshtein distance between both values across our dataset is 2.36 ± 4.89, suggesting users typically use similar strings for both account meta-features. The account's description is more lengthy and thus contains more information. It is usually a sentence with 3.62 ± 6.33 words, resulting in the gating network placing more emphasis on it.

Individual Dataset Results We present the results of BotBuster-Singular and BotBuster-Full in Table 3. These results are compared against the baseline algorithms. In general, singular runs of each dataset (BotBuster-Singular) performed better than the baselines, but does not perform as well as the merged training model (BotBuster-Full). Thus, augmenting the training data by merging all the datasets for model training and hyperparamter tuning enhances the model performance.

Out of the three model variations trained, the BotBuster-4 variation performed the best (F1=72.23±18.84) across all the test datasets. It outperforms the baselines of BotHunter (F1=49.61±30.10) and Botometer (F1=40.63±26.33). From the independent dataset analysis, we observe that the standard deviation of F1 scores of BotBuster is smaller than the baseline algorithms. This aids to the generalizability of the BotBuster classifier across vastly different datasets.

Observing that BotBuster-3/4 variations performed better than BotBuster-1/2 variations lends to the fact that a post-level post expert differentiates post information between bots and humans better than an account-level post expert. While the use of derived post linguistic values does not significantly improve model accuracy for account-level post expert (BotBuster-1 vs BotBuster-2), it does so for the post-level post expert (BotBuster-3 vs BotBuster-4). BotBuster performs most poorly are datasets from 2017, suggesting the mutable behavior of bots over time (Luceri et al. 2019).

Our input-agnostic BotBuster architecture requires only specification of matching field and does not rely on any specific data format and opens up possibilities for multi-platform bot-detectors. As such, BotBuster is able to perform predictions on Reddit data, despite it not having the same fields as Twitter data. Observing that BotBuster-4 performs well on both the Reddit and Twitter datasets supports our architecture and opens up discussion for the similar behaviour of bots on both platforms.

Evaluation against external dataset The results of the external evaluation against the TwiBot-20 dataset are presented in Table 4. BotBuster outperforms the baseline models (F1=60.92 vs $\bar{F}1$=34.94), illustrating the robustness of the model in characterizing bots.

Stability of BotBuster Scores

One key characteristic of a good bot detection algorithm is the stability of its resultant bot-probability scores. A stable

Dataset	BotHunter	Botometer	BotBuster-Singular	BotBuster-Full	BotBuster-Subset (BotHunter returns)	BotBuster-Subset (Botometer returns)
BotBuster-1: username, screenname, descriptions, user metadata experts + account-level post expert						
astroturf	15.60	33.50	**50.00**	**65.90**	15.72	33.60
botometer-feedback-2019	74.10	53.68	49.52	29.10	25.19	25.20
botwiki-2019	53.13	92.89	45.31	79.03	79.00	79.02
cresci-rtbust-2019	62.90	60.10	61.84	51.78	51.78	51.78
cresci-stock-2018	37.36	38.12	**61.22**	**71.49**	56.17	54.53
gilani-2017	63.91	-	53.62	47.88	52.55	-
midterm-2018	15.30	11.90	**46.64**	**84.80**	8.00	16.06
political-bots-2019	-	20.60	**45.45**	**98.38**	-	95.03
varol-2017	73.80	65.30	44.74	32.18	32.89	33.40
verified-2019	100	30.20	46.80	87.10	97.29	80.34
reddit	-	-	43.51	53.21	-	-
Average	55.12	45.14	49.87	**63.71**	46.51	52.11
BotBuster-2: username, screenname, descriptions, user metadata experts + account-level post expert with derived post values						
astroturf	15.60	33.50	**43.13**	**68.24**	16.50	33.54
botometer-feedback-2019	74.10	53.68	22.05	29.11	29.23	25.20
botwiki-2019	53.13	92.89	44.88	79.12	78.93	79.36
cresci-rtbust-2019	62.90	60.10	51.32	51.78	51.67	51.00
cresci-stock-2018	37.36	38.12	**65.89**	**71.50**	56.17	54.53
gilani-2017	63.91	-	50.04	47.88	52.55	-
midterm-2018	15.30	11.90	**66.14**	**84.80**	8.02	16.00
political-bots-2019	-	20.60	**97.45**	**98.38**	-	99.23
varol-2017	73.80	65.30	42.02	32.20	32.90	33.40
verified-2019	100	30.20	47.36	**87.05**	98.60	88.40
reddit	-	-	48.32	57.02	-	-
Average	55.12	45.14	52.60	**64.28**	47.17	53.41
BotBuster-3: username, screenname, descriptions, user metadata experts + post-level post expert						
astroturf	15.60	33.50	**50.34**	**80.96**	18.90	29.95
botometer-feedback-2019	74.10	53.68	28.30	29.67	25.46	29.23
botwiki-2019	53.13	92.89	38.57	80.96	78.93	79.51
cresci-rtbust-2019	62.90	60.10	42.32	51.57	52.10	51.66
cresci-stock-2018	37.36	38.12	**70.33**	**71.49**	56.17	54.53
gilani-2017	63.91	-	55.09	48.00	-	52.56
midterm-2018	15.30	11.90	**83.89**	**84.87**	8.10	6.26
political-bots-2019	-	20.60	**52.36**	**49.59**	-	97.95
varol-2017	73.80	65.30	47.80	32.18	32.89	33.40
verified-2019	100	30.20	47.37	**87.00**	97.29	88.44
reddit	-	-	52.87	57.03	-	-
Average	55.12	45.14	51.74	**59.39**	46.23	52.35
BotBuster-4: username, screenname, descriptions, user metadata experts + post-level post expert with derived post values						
astroturf	15.60	33.50	**58.00**	**76.90**	17.21	34.28
botometer-feedback-2019	74.10	53.68	41.82	54.20	68.62	59.90
botwiki-2019	53.13	92.89	70.00	80.90	78.90	79.51
cresci-rtbust-2019	62.90	60.10	**63.14**	**67.20**	72.23	70.20
cresci-stock-2018	37.36	38.12	**62.19**	**79.35**	75.44	71.13
gilani-2017	63.91	-	57.04	48.80	53.23	-
midterm-2018	15.30	11.90	**83.93**	**94.13**	90.20	73.48
political-bots-2019	-	20.60	**100**	**98.38**	-	97.95
varol-2017	73.80	65.30	41.83	45.50	48.78	51.39
verified-2019	100	30.20	**100**	89.15	99.72	90.62
reddit	-	-	69.77	60.04	-	-
Average	-	-	67.97	72.23	-	69.83
Avg (BotHunter returns)	55.12	-	64.21	63.61	67.14	-
Avg (Botometer returns)	-	45.14	70.68	76.19	-	69.83

Table 3: Macro-F1 scores for three variations of BotBuster and the baseline comparisons. BotBuster results better than all baselines are **bolded** and results better than only one baseline are underlined. For BotBuster-4, we also report the average BotBuster scores compared to the datasets the baselines return results in addition to the overall average score

Model	Users analyzed (%)	Macro-F1
BotHunter	99.15	49.02
Botometer	91.38	20.86
BotBuster-4	100	60.92
TwiBot-20	100	85.46

Table 4: Macro-F1 scores of external evaluation dataset

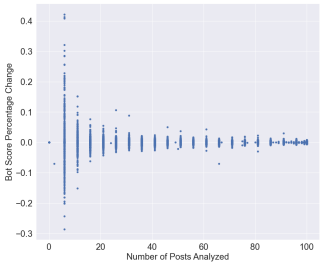

Figure 2: Difference in BotBuster-4 scores across increasing number of posts of 1000 randomly selected accounts. The difference drops drastically after 21 posts.

bot score is one that changes minimally across an investigation time frame, thus providing reliable characterization of bots for downstream tasks like detection of influence campaigns (Ng, Robertson, and Carley 2021).

We empirically study the amount of data required for a stable BotBuster score. We randomly select 500 still-alive bots and 500 non-bots from the dataset and collected their latest 100 posts using the Twitter V2 REST API. We then run the BotBuster-3 algorithm beginning with one post then by incremental steps of 5 posts, up to 100 posts. We analyze the percentage change in BotBuster score across the number of posts (Figure 2). The difference in scores initially changes drastically from an initial change of -0.286 ± 0.0871, then drops to a change of 3.70E-5$\pm$1.15E-2 at 21 posts, and tends to 0 after 36 posts (-7.80E-6 $\pm$ 6.86E-3). Thus, BotBuster scores do stabilize, lending confidence in the algorithm. The observations provide evidence for usage of BotBuster estimation: at least 21 posts should be collected for a stable bot score and 36 for a score that changes minimally.

Discussion

In this work, we built a social bot detection model, BotBuster, leveraging on the mixture of experts architecture. This architecture overcomes the limitation of requiring every single feature field before being able to make a bot prediction of the account, and extend the bot prediction from a bot detector that deals with only one platform to one that can deal with two platforms.

We performed the same feature engineering on 10 Twitter datasets and one Reddit dataset, demonstrating that the features are generalizable across datasets and platforms. These features are general to the social media space, because every user in all social media platforms have a username, posts and other account metadata statistics. In the event any of the feature is missing because the platform does not have the feature tagged to the user, the expert is not used in the evaluation to whether the user is a bot or not. Thus, this method is scalable to general social bot detection and only requires a mapping of feature names.

Our results identify that a merged training strategy performs better than individual dataset training strategies. Training the model on an augmented dataset aids in model generalizability where the model learns features across a variety of data types. BotBuster demonstrated its robustness in achieving a 2-4 times better accuracy than the baseline algorithms on an external dataset. Although BotBuster was trained on older datasets, it is sufficiently robust enough for bot detection on the newer TwiBot-20 dataset. The accuracy achieved by BotBuster is lower than the model built by the TwiBot-20 authors, which may be due to the inclusion of network information in bot prediction.

In that light, the BotBuster model should be continually updated with new datasets to keep up with the changing nature of bots. Its construct relies on human annotations of social accounts as a supervised learning approach. Additionally, the bot/human labels are manually annotated, meaning there could be false positives in either of the classes where humans are unable to distinguish bot and human account. Future work can exploit features like network structure information or temporal activity patterns which have shown to yield robust results in bot detection as additional experts.

Ethical Considerations

Social bot detection through automated means bring about a key ethical consideration: accuracy, transparency and robustness of a social bot detection algorithm collectively forms a "devil's triangle" (Thieltges, Schmidt, and Hegelich 2016). Accuracy is paramount as misclassification can lead to the deplatforming of legitimate social media users. On the other hand, to enable wider usage, the algorithm should be transparent. However, an increase in algorithm transparency provides bot-operators information to adapt their bot account characteristics to evade detection, increasing the variation of bot characteristics. The alteration of bot behavior based on the knowledge of the bot detection algorithm creates a drop in the robustness and accuracy of the algorithm in the detection of new and evolved bot accounts. Bot detection is a cat-and-mouse game; transparency must be balanced with robustness (Fazzolari et al. 2020).

Conclusion

Given the prevalence of bots on social media platforms and the possibility of incomplete data collection and improved platform API data formats, the need for an improved bot detection method to be format-agnostic and handle incomplete data is apparent. We proposed BotBuster, a novel mixture of experts approach allowing us to perform bot detection despite incomplete data collection. Our model can achieve an extremely high accuracy of $\bar{F}1 = 72.23$ and provides stable bot probability scores after 36 posts. The performance of BotBuster on both Twitter and Reddit datasets gives hope to generalizability of bot detectors and suggests that bots operate similarly across these two platforms.

Acknowledgments

The research for this paper was supported in part by the Knight Foundation, the Office of Naval Research (Bot Hunter, Grant N000141812108, Group Polarization in Social Media N000141812106), Additional support was provided by the Center for Computational Analysis of Social and Organizational Systems (CASOS) and the Center for Informed Democracy and Social Cybersecurity (IDeaS) at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Knight Foundation, the Office of Naval Research, or the U.S. Government.

References

Benton, A.; Mitchell, M.; and Hovy, D. 2017. Multitask Learning for Mental Health Conditions with Limited Social Media Data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 152–162.

Beskow, D. M.; and Carley, K. M. 2018. Bot-hunter: a tiered approach to detecting & characterizing automated activity on twitter. In *Conference paper. SBP-BRiMS: International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, volume 3, 3.

Bovet, A.; and Makse, H. A. 2019. Influence of fake news in Twitter during the 2016 US presidential election. *Nature Communications*, 10(1): 7.

Cao, Q.; Sirivianos, M.; Yang, X.; and Pregueiro, T. 2012. Aiding the detection of fake accounts in large scale social online services. In *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 197–210.

Caruana, R.; Niculescu-Mizil, A.; Crew, G.; and Ksikes, A. 2004. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, 18.

Chavoshi, N.; Hamooni, H.; and Mueen, A. 2016. Debot: Twitter bot detection via warped correlation. In *IEEE International Conference on Data Mining (ICDM)*, 817–822.

Cresci, S.; Lillo, F.; Regoli, D.; Tardelli, S.; and Tesconi, M. 2018. FAKE: Evidence of spam and bot activity in stock microblogs on Twitter. In *Twelfth international AAAI conference on web and social media*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Diesner, J.; Ferrari, E.; and Xu, G. 2017. *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*.

Djandji, M.; Baly, F.; Antoun, W.; and Hajj, H. 2020. Multi-task learning using AraBert for offensive language detection. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, 97–101.

Fazzolari, M.; Pratelli, M.; Martinelli, F.; and Petrocchi, M. 2020. Emotions and Interests of Evolving Twitter Bots. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 1–8. IEEE.

Feng, S.; Wan, H.; Wang, N.; Li, J.; and Luo, M. 2021. TwiBot-20: A Comprehensive Twitter Bot Detection Benchmark. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 4485–4494.

Ferraz Costa, A.; Yamaguchi, Y.; Juci Machado Traina, A.; Traina, C.; and Faloutsos, C. 2015. RSC: Mining and Modeling Temporal Activity in Social Media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 269–278. New York, NY, USA: Association for Computing Machinery. ISBN 9781450336642.

Hurtado, S.; Ray, P.; and Marculescu, R. 2019. Bot Detection in Reddit Political Discussion. In *Proceedings of the Fourth International Workshop on Social Sensing*, SocialSense'19, 30–35. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367066.

Kniffin, J. D. 1979. The New Readability Requirements For Military Technical Manuals. *Technical Communication*, 26(3): 16–19.

Li, Y.; and Caragea, C. 2021. A Multi-Task Learning Framework for Multi-Target Stance Detection. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2320–2326. Online: Association for Computational Linguistics.

Luceri, L.; Deb, A.; Giordano, S.; and Ferrara, E. 2019. Evolution of bot and human behavior during elections. *First Monday*, 24(9).

Magelinski, T.; Beskow, D. M.; and Carley, K. M. 2020. GraphHist: Graph Classification from Latent Feature Histograms with Application to Bot Detection. In *AAAI*, 5134–5141.

Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; and Tesconi, M. 2019. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM Conference on Web Science*, 183–192.

Mittos, A.; Zannettou, S.; Blackburn, J.; and Cristofaro, E. D. 2020. Analyzing Genetic Testing Discourse on the Web Through the Lens of Twitter, Reddit, and 4chan. *ACM Transactions on the Web (TWEB)*, 14(4): 1–38.

Ng, L. H. X.; and Carley, K. M. 2021. Bot-Based Emotion Behavior Differences in Images During Kashmir Black Day Event. In Thomson, R.; Hussain, M. N.; Dancy, C.; and Pyke, A., eds., *Social, Cultural, and Behavioral Modeling*, 184–194. Cham: Springer International Publishing. ISBN 978-3-030-80387-2.

Ng, L. H. X.; Robertson, D. C.; and Carley, K. M. 2021. Stabilizing a Supervised Bot Detection Algorithm: How Much Data is Needed for Consistent Predictions? Special Issue on Information and Opinion Diffusion in Online Social Networks and Media.

Orabi, M.; Mouheb, D.; Al Aghbari, Z.; and Kamel, I. 2020. Detection of Bots in Social Media: A Systematic Review. *Information Processing & Management*, 57(4): 102250.

Peng, H.; Schwartz, R.; Li, D.; and Smith, N. A. 2020. A Mixture of h-1 Heads is Better than h Heads. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6566–6577.

Pérez, J. M.; Giudici, J. C.; and Luque, F. 2021. pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks. arXiv:2106.09462.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sayyadiharikandeh, M.; Varol, O.; Yang, K.-C.; Flammini, A.; and Menczer, F. 2020. Detection of novel social bots by ensembles of specialized classifiers. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2725–2732.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shen, T.; Ott, M.; Auli, M.; and Ranzato, M. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, 5719–5728. PMLR.

Smith-Lovin, L.; Robinson, D. T.; Cannon, B. C.; Clark, J. K.; Freeland, R.; Morgan, J. H.; and Rogers, K. B. 2016. Mean affective ratings of 929 identities, 814 behaviors, and 660 modifiers by university of georgia and duke university undergraduates and by community members in durham, nc, in 2012-2014. *University of Georgia: Distributed at UGA Affect Control Theory Website: http://research. franklin. uga. edu/act*.

Stella, M.; Ferrara, E.; and De Domenico, M. 2018. Bots increase exposure to negative and inflammatory content in online social systems. *Proceedings of the National Academy of Sciences*, 115(49): 12435–12440.

Tausczik, Y. R.; and Pennebaker, J. W. 2010. The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. *Journal of Language and Social Psychology*, 29(1): 24–54.

Thieltges, A.; Schmidt, F.; and Hegelich, S. 2016. The devil's triangle: Ethical considerations on developing bot detection methods. In *2016 AAAI Spring Symposium Series*.

Trujillo, M.; Rosenblatt, S.; de Anda Jáuregui, G.; Moog, E.; Samson, B. P. V.; Hébert-Dufresne, L.; and Roth, A. M. 2021. When the Echo Chamber Shatters: Examining the Use of Community-Specific Language Post-Subreddit Ban. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, 164–178. Online: Association for Computational Linguistics.

Tyagi, A. 2021. *Challenges in Climate Change Communication on Social Media*. Ph.D. thesis, Carnegie Mellon University.

Uyheng, J.; Ng, L. H. X.; and Carley, K. M. 2021. Active, aggressive, but to little avail: characterizing bot activity during the 2020 Singaporean elections. *Computational and Mathematical Organization Theory*.

Varol, O.; Ferrara, E.; Davis, C.; Menczer, F.; and Flammini, A. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the international AAAI conference on web and social media*, volume 11.

Yang, K.-C.; Varol, O.; Davis, C. A.; Ferrara, E.; Flammini, A.; and Menczer, F. 2019. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies*, 1(1): 48–61.

Yang, K.-C.; Varol, O.; Hui, P.-M.; and Menczer, F. 2020. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1096–1103.

Appendix: Supplementary Tables

Dataset Information

Table 7 describes the datasets composition used in this paper. Twitter datasets are obtained from online bot repository http://botometer.org/botrepository and hydrated in July 2021. We constructed the Reddit dataset with information from http://botrank.pastimes.eu and the following subreddits: r/showerthoughts, r/nosleep, r/AmItheAsshole, r/changemyview. We annotate the cells where there are accounts where those information are incomplete.

Feature Engineering Information

Table 5 lists the features used in our BotBuster framework. For the LIWC features used in the derived post features, the exact feature names used are:

1. Time: focuspast, focuspresent, focusfuture
2. Affect: anx, anger, sad, family, friend
3. Social: female, male, social, affiliation
4. Drives: achieve, power, reward, risk
5. Pronouns: first, second, third

Feature	Type	Reference library
Extracted User Features		
username	string	-
screenname	string	-
description	string	-
number of followers	integer	-
number of following	integer	-
total number of posts	integer	-
listed count	integer	-
is verified	boolean	-
is protected	boolean	-
Extracted Post Features		
post text	string	-
number of retweets	integer	-
number of likes	integer	-
number of quotes	integer	-
number of replies	integer	-
Derived Post Features		
sentiment	float	(Pérez, Giudici, and Luque 2021)
reading score	float	(Kniffin 1979)
EPA scores	integer	(Smith-Lovin et al. 2016; Tyagi 2021)
LIWC features: time, affect, social, drives, pronouns values	integer	(Tausczik and Pennebaker 2010)

Table 5: List of features used in our BotBuster framework. BotBuster-1 and BotBuster-3 does not use the derived features while BotBuster-2 and BotBuster-4 used all the features.

Expert Weights

Table 6 show expert weights in BotBuster variations and for the best performing BotBuster-4 variation with missing information.

Model	User name	Screen name	Description	User Metadata	Post (account-level)	Post (post-level)
BotBuster variation for accounts with all information						
BotBuster-1	13.88	10.43	26.85	9.29	40.46	-
BotBuster-2	15.36	15.41	23.64	14.96	30.63	-
BotBuster-3	27.46	20.57	20.71	15.58	-	15.67
BotBuster-4	20.99	20.89	24.13	19.26	-	15.71
BotBuster-4 variation for accounts with missing information						
BotBuster 4.1	-	31.46	35.17	12.61	-	18.76
BotBuster 4.2	24.90	-	31.51	14.43	-	29.17
BotBuster 4.3	36.09	21.02	25.80	-	-	17.08
BotBuster 4.4	18.45	20.85	-	27.25	-	33.45
BotBuster 4.5	39.26	21.13	20.34	-	-	

Table 6: Sampling of expert weights (%) for each BotBuster variation for accounts with all information and the variations for BotBuster-4. A higher value reflects higher importance is placed on the expert.

Dataset	Bots	Humans	Reference	User name	Screen name	Desc	Posts	User Meta data
astroturf	585	0	(Sayyadiharikandeh et al. 2020)					
botometer-feedback-2019	143	386	(Yang et al. 2019)					
botwiki-2019	704	0	(Yang et al. 2020)					
cresci-rtbust-2019	391	368	(Mazza et al. 2019)					
cresci-stock-2018	18508	7479	(Cresci et al. 2018)					
gilani-2017	1130	1522	(Diesner, Ferrari, and Xu 2017)					
midterm-2018	42446	8092	(Yang et al. 2020)					
political-bots-2019	62	0	(Yang et al. 2019)					
varol-2017	826	1747	(Varol et al. 2017)					
verified-2019	0	2000	(Yang et al. 2020)					
reddit	500	167	-					
Total	65295	21761	-					
TwiBot-20 (external evaluation)	640	543	(Feng et al. 2021)					

Table 7: Dataset composition of labelled bot and human accounts. The datasets are hydrated in July 2021 and the greyed-out cells indicate where fields with incomplete data.