Scalable and Generalizable Social Bot Detection through Data Selection

Kai-Cheng Yang,[1] **Onur Varol,**[2] **Pik-Mai Hui,**[1] **Filippo Menczer**[1,3]
[1]Center for Complex Networks and Systems Research, Indiana University, Bloomington, IN, USA
[2]Center for Complex Networks Research, Northeastern University, Boston, MA, USA
[3]Indiana University Network Science Institute, Bloomington, IN, USA

Abstract

Efficient and reliable social bot classification is crucial for detecting information manipulation on social media. Despite rapid development, state-of-the-art bot detection models still face generalization and scalability challenges, which greatly limit their applications. In this paper we propose a framework that uses minimal account metadata, enabling efficient analysis that scales up to handle the full stream of public tweets of Twitter in real time. To ensure model accuracy, we build a rich collection of labeled datasets for training and validation. We deploy a strict validation system so that model performance on unseen datasets is also optimized, in addition to traditional cross-validation. We find that strategically selecting a subset of training data yields better model accuracy and generalization than exhaustively training on all available data. Thanks to the simplicity of the proposed model, its logic can be interpreted to provide insights into social bot characteristics.

Introduction

Social bots are social media accounts controlled in part by software. Malicious bots serve as important instruments for orchestrated, large-scale opinion manipulation campaigns on social media (Ferrara et al. 2016a; Subrahmanian et al. 2016). Bots have been actively involved in online discussions of important events, including recent elections in the US and Europe (Bessi and Ferrara 2016; Deb et al. 2019; Stella, Ferrara, and De Domenico 2018; Ferrara 2017). Bots are also responsible for spreading low-credibility information (Shao et al. 2018) and extreme ideology (Berger and Morgan 2015; Ferrara et al. 2016b), as well as adding confusion to the online debate about vaccines (Broniatowski et al. 2018).

Scalable and reliable bot detection methods are needed to estimate the influence of social bots and develop effective countermeasures. The task is challenging due to the diverse and dynamic behaviors of social bots. For example, some bots act autonomously with minimal human intervention, others are manually controlled so that a single entity can create the appearance of multiple human accounts (Grimme et al. 2017). And while some bots are active continuously, others focus bursts of activity on different short-term targets. It is extremely difficult for humans with limited access to social media data to recognize bots, making people vulnerable to manipulation.

Many machine learning frameworks for bot detection on Twitter have been proposed in the past several years (see a recent survey (Alothali et al. 2018) and the Related Work section). However, two major challenges remain: scalability and generalization.

Scalability enables analysis of streaming data with limited computing resources. Yet most current methods attempt to maximize accuracy by inspecting rich contextual information about an account's actions and social connections. Twitter API rate limits and algorithmic complexity make it impossible to scale up to real-time detection of manipulation campaigns (Stella, Ferrara, and De Domenico 2018).

Generalization enables the detection of bots that are different from those in the training data, which is critical given the adversarial nature of the problem — new bots are easily designed to evade existing detection systems. Methods considering limited behaviors, like retweeting and temporal patterns, can only identify certain types of bots. Even more general-purpose systems are trained on labeled datasets that are sparse, noisy, and not representative enough. As a result, methods with excellent cross-validation accuracy suffer dramatic drops in performance when tested on cross-domain accounts (De Cristofaro et al. 2018).

In this paper, we propose a framework to address these two challenges. By focusing just on user profile information, which can be easily accessed in bulk, scalability is greatly improved. While the use of fewer features may entail a small compromise in individual accuracy, we gain the ability to analyze a large-volume stream of accounts in real time.

We build a rich collection of labeled data by compiling all labeled datasets available in the literature and three newly-produced ones. We systematically analyze the feature space of different datasets and the relationship between them; some of the datasets tend to overlap with each other while others have contradicting patterns.

The diversity of the compiled datasets allows us to build a more robust classifier. Surprisingly, by carefully selecting a subset of the training data instead of merging all of the datasets, we achieve better performance in terms of

cross-validation, cross-domain generalization, and consistency with a widely-used reference system. We also show that the resulting bot detector is more interpretable. The lessons learned from the proposed data selection approach can be generalized to other adversarial problems.

Related Work

Most bot detection methods are based on supervised machine learning and involve manually labeled data (Ferrara et al. 2016a; Alothali et al. 2018). Popular approaches leverage user, temporal, content, and social network features with random forest classifiers (Davis et al. 2016; Varol et al. 2017; Gilani, Kochmar, and Crowcroft 2017; Yang et al. 2019). Faster classification can be obtained using fewer features and logistic regression (Ferrara 2017; Stella, Ferrara, and De Domenico 2018). Some methods include information from tweet content in addition to the metadata to detect bots at the tweet level (Kudugunta and Ferrara 2018). A simpler approach is to test the randomness of the screen name (Beskow and Carley 2019). The method presented here also adopts the supervised learning framework with random forest classifiers, attempting to achieve both the scalability of the faster methods and the accuracy of the feature-rich methods. We aim for greater generalization than all the models in the literature.

Another strain of bot detection methods goes beyond individual accounts to consider collective behaviors. Unsupervised learning methods are used to find improbable similarities among accounts. No human labeled datasets are needed. Examples of this approach leverage similarity in post timelines (Chavoshi, Hamooni, and Mueen 2016), action sequences (Cresci et al. 2016; 2018a), content (Chen and Subramanian 2018), and friends/followers (Jiang et al. 2016). Coordinated retweeting behavior has also been used (Mazza et al. 2019). Methods aimed at detecting coordination are very slow as they need to consider many pairs of accounts. The present approach is much faster, but considers individual accounts and so cannot detect coordination.

Feature Engineering

All Twitter bot detection methods need to query data before performing any evaluation, so they are bounded by API limits. Take Botometer, a popular bot detection tool, as an example. The classifier uses over 1,000 features from each account (Varol et al. 2017; Yang et al. 2019). To extract these features, the classifier requires the account's most recent 200 tweets and recent mentions from other users. The API call has a limit of 43,200 accounts per API key in each day. Compared to the rate limit, the CPU and Internet i/o time is negligible. Some other methods require the full timeline of accounts (Cresci et al. 2016) or the social network (Minnich et al. 2017), taking even longer.

We can give up most of this contextual information in exchange for speed, and rely on just user metadata (Ferrara 2017; Stella, Ferrara, and De Domenico 2018). This metadata is contained in the so-called user object from the Twitter API. The rate limit for users lookup is 8.6M accounts per API key in each day. This is over 200 times the rate limit that bounds Botometer. Moreover, each tweet collected from Twitter has an embedded user object. This brings two extra advantages. First, once tweets are collected, no extra queries are needed for bot detection. Second, while users lookups always report the most recent user profile, the user object embedded in each tweet reflects the user profile at the moment when the tweet is collected. This makes bot detection on archived historical data possible.

Table 1 lists the features extracted from the user object. The rate features build upon the user age, which requires the probe time to be available. When querying the users lookup API, the probe time is when the query happens. If the user object is extracted from a tweet, the probe time is the tweet creation time (created_at field). The user_age is defined as the hour difference between the probe time and the creation time of the user (created_at field in the user object). User ages are associated with the data collection time, an artifact irrelevant to bot behaviors. In fact, tests show that including age in the model deteriorates accuracy. However, age is used to calculate the rate features. Every count feature has a corresponding rate feature to capture how fast the account is tweeting, gaining followers, and so on. In the calculation of the ratio between followers and friends, the denominator is $\max(\text{friends\_count}, 1)$ to avoid division-by-zero errors.

The screen_name_likelihood feature is inspired by the observation that bots sometimes have a random string as screen name (Beskow and Carley 2019). Twitter only allows letters (upper and lower case), digits, and underscores in the screen_name field, with a 15-character limit. We collected over 2M unique screen names and constructed the likelihood of all 3,969 possible bigrams. The likelihood of a screen name is defined by the geometric-mean likelihood of all bigrams in it. We do not consider longer n-grams as they require more resources with limited advantages. Tests show the likelihood feature can effectively distinguish random strings from authentic screen names.

Datasets

To train and test our model, we collect all public datasets of labeled human and bot accounts and create three new ones, all available in the bot repository (botometer.org/bot-repository). See overview in Table 2.

Let us briefly summarize how the datasets were collected. The **caverlee** dataset consists of bot accounts lured by honeypot accounts and verified human accounts (Lee, Eoff, and Caverlee 2011). This dataset is older than the others. To obtain the **varol-icwsm** dataset, the authors manually labeled accounts sampled from different Botometer score deciles (Varol et al. 2017). The dataset was designed to be representative of diverse types of accounts. The bot accounts in the **cresci-17** dataset contain a more fine-grained classification: traditional spambots, social spambots, and fake followers (Cresci et al. 2017). Traditional spambots are simple bots that tweet the same content repeatedly. Social spambots mimic the profiles and actions of normal users so they are not suspicious when inspected individually. But the authors found them promoting certain hashtags or content in a coordinated fashion. Fake followers are accounts paid to follow other accounts.

Table 1: List of features used in our framework. User metadata features are extracted directly from the user object fetched from the API. Twitter stopped providing the geo_enabled field as of May, 2019, so we remove it from the feature list although it proved helpful. Derived features are calculated based on the user metadata. Some are explained in the text, others are self-explanatory.

user metadata		derived features		
feature name	type	feature name	type	calculation
statuses_count	count	tweet_freq	real-valued	statuses_count / user_age
followers_count	count	followers_growth_rate	real-valued	followers_count / user_age
friends_count	count	friends_growth_rate	real-valued	friends_count / user_age
favourites_count	count	favourites_growth_rate	real-valued	favourites_count / user_age
listed_count	count	listed_growth_rate	real-valued	listed_count / user_age
default_profile	binary	followers_friends_ratio	real-valued	followers_count / friends_count
profile_use_background_image	binary	screen_name_length	count	length of screen_name string
verified	binary	num_digits_in_screen_name	count	no. digits in screen_name string
		name_length	count	length of name string
		num_digits_in_name	count	no. digits in name string
		description_length	count	length of description string
		screen_name_likelihood	real-valued	likelihood of the screen_name

The **pronbots** dataset consists of a group of bots that share scam sites. The dataset was first shared by Andy Patel (github.com/r0zetta/pronbot2) and then collected for study (Yang et al. 2019). The **celebrity** dataset is made of accounts selected among celebrities (Yang et al. 2019). Accounts in the **vendor-purchased** dataset are fake followers purchased by researchers from several companies (Yang et al. 2019). The **botometer-feedback** dataset was constructed by mannually labeling accounts flagged by feedback from Botometer users (Yang et al. 2019). The **political-bots** dataset is a group of politics-oriented bots shared by Twitter user `@josh_emerson` (Yang et al. 2019). For the **gilani-17** dataset, accounts collected using the Twitter streaming API were grouped into four categories based on the number of followers (Gilani et al. 2017). The authors then sampled accounts from the four categories and had four undergraduate students annotate them based on key information compiled in a table. For the **cresci-rtbust** dataset, the authors collected all Italian retweets between 17–30 June 2018, then manually annotated a roughly balanced set of about 1,000 human and bot accounts (Mazza et al. 2019). Bots in the **cresci-stock** dataset were isolated by finding accounts with similar timelines among tweets containing selected cashtags collected during five months in 2017 (Cresci et al. 2018b; 2019).

We created three more datasets. The **midterm-18** dataset was filtered based on political tweets collected during the 2018 U.S. midterm elections (Yang, Hui, and Menczer 2019). We manually identified some of the genuine human users that were actively involved in the online discussion about the elections. The bot accounts were spotted through suspicious correlations in their creation and tweeting timestamps. Most of the bot accounts have been suspended by Twitter after the elections, which confirms our labeling. The **botwiki** dataset is based on the botwiki.org archive of self-identified bot accounts. We manually removed inactive accounts and those from platforms other than Twitter. Finally, the **verified** dataset was generated by filtering the streaming API for verified accounts. This dataset is added as a supplement to balance `vendor-purchased` and `botwiki`, because the analyses in the following sections require the presence of both human and bot accounts. However, since the verified account flag is a feature of the classifier, we set this feature to 'false' for the human accounts; this is a conservative choice that prevents any bias in favor of our model.

Table 2: Datasets of labeled bot and human accounts.

Dataset	#bots	#human
`caverlee`	15,483	14,833
`varol-icwsm`	733	1,495
`cresci-17`	7,049	2,764
`pronbots`	17,882	0
`celebrity`	0	5,918
`vendor-purchased`	1,087	0
`botometer-feedback`	139	380
`political-bots`	62	0
`gilani-17`	1,090	1,413
`cresci-rtbust`	353	340
`cresci-stock`	7,102	6,174
`midterm-18`	42,446	8,092
`botwiki`	698	0
`verified`	0	1,987
Total	94,124	43,396

By the time we collected the datasets, some of the accounts had been suspended already, so the numbers shown in Table 2 might be smaller than those in the original papers.

Beside the labeled datasets that served as ground truth, we also created a dataset of 100,000 random users collected from the streaming API in 2018. This data was used for model evaluation, as discussed in a later section.

Data Characterization

Since there are drastically different types of bot (and human) accounts (De Cristofaro et al. 2018), characterizing the accounts in different datasets can provide us with useful in-

sight into the design of a generalizable classifier.

Independent Dataset Analysis

The most intuitive way to inspect the different training datasets is to visualize them in feature space. To highlight the contrast between human and bot accounts, we merge some of the single-class datasets in Table 2. Specifically, `pronbots` and `celebrity` are combined to form the `pron-celebrity` dataset; `botometer-feedback` and `political-bots` are merged into `political-feedback`; `verified` is split into two parts, merged with `botwiki` and `vendor-purchased` to obtain the roughly balanced `botwiki-verified` and `vendor-verified`, respectively. The merging here is purely for analysis; we stick to the original datasets for data selection purposes below.

We apply PCA to project each dataset into the 2-D plane for visulization; t-SNE yields similar separation patterns. Since most of the features have wide ranges and skewed distributions, we first rescale them via log-transforms. To quantify the separation of human and bot accounts in each dataset, we apply a kNN classifier in the original feature space. For each account, we identify their nearest k neighbors and assign the majority label to the focal account. With the labels obtained from kNN and the ground truth, we are able to calculate the homogeneity score for each dataset (Rosenberg and Hirschberg 2007). The kNN algorithm is stable when $k > 3$; we choose $k = 9$ for our analysis. Because the datasets have various sizes and class ratios, we sample 500 bots and 500 humans (or fewer for datasets without enough accounts) to calculate the homogeneity score. This procedure is repeated 1,000 times for each dataset to generate a distribution of the scores.

Fig. 1 shows the PCA scatter plots and the homogeneity scores for all datasets. Five out of 11 datasets demonstrate a clearly clustered structure, suggesting bots and humans are easily separable using our features. The rest of the datasets have clusters that are not as easily separable. This is consistent with prior results showing no clear separation using t-SNE plots (Varol et al. 2017; De Cristofaro et al. 2018).

Among the five less separable datasets, `cresci-stock` was labeled based on the timeline similarity between accounts. Such coordination cannot be captured by our feature-based approach (Cresci et al. 2017; Yang et al. 2019). The other four datasets are manually annotated and include different types of accounts; many of them exhibit behaviors that even humans find difficult to distinguish. For example, `varol-icwsm` has 75% inter-annotator agreement. The mixed structure in feature space is understandable in light of these characteristics.

Cross-Dataset Analysis

Let us explore whether human and bot classes are consistent across different datasets. Visualizing all the data together is unfeasible because we have too many datasets with too many data points. Let us instead use generalization as a proxy for consistency, by training the models on one dataset and testing on another. If the datasets are consistent, we expect good generalization power.

The matrix in Fig. 2 maps cross-dataset AUC using random forest classifiers. When tested on cross-domain data, the generalization AUC varies. In some cases, it is very high indicating datasets with consistent classes. But no dataset can generalize well on all other datasets. In fact, the AUC can sometimes be below 0.5, suggesting that training and test datasets have contradictory labels. Many factors may contribute to this phenomenon. First, different datasets were annotated by different people with different standards using different methods. Considering the difficult nature of the problem, labeling noise is not surprising. Second, while our 20 features enable scalability, they only capture a tiny portion of an account's characteristics. The seemingly contradicting patterns in our feature space might be resolved by considering additional features.

Another important observation is the asymmetry of Fig. 2: for a pair of datasets, exchanging the training and test sets may lead to different results. We can define the *separability* of a dataset by using it to test models trained on other classifiers — averaging AUC across columns of the matrix in Fig. 2. For example, classifiers trained on most of the datasets achieve good performance on `botwiki-verified`, suggesting easy separability. On the other hand, `cresci-rtbust` is not easily separable because classifiers trained on other datasets perform poorly on it. Similarly, we can define the *generalizability* of a dataset by training a model on it and testing on other datasets — averaging AUC across rows of the matrix in Fig. 2. We find no clear correlation between separability and generalizability (Spearman's $r = 0.18$, $p = 0.6$): the fact that bots in one dataset can be easily detected does not imply that a classifier trained on that dataset can detect bots in other datasets.

Generalizability

Random forest achieves perfect AUC when trained and tested on any single dataset, and excellent AUC in cross-validation (Alothali et al. 2018; Yang et al. 2019); it is expressive enough to capture the non-linear patterns discriminating human and bot accounts even in the least separable datasets. Therefore, the proposed framework does not aim to provide a more expressive algorithm; we will stick with random forest. Instead, our goal is to address the poor cross-dataset generalization highlighted in the previous section and in the literature (De Cristofaro et al. 2018).

Model Evaluation

We have seen that good cross-validation performance, even on multiple datasets, does not guarantee generalization across unseen datasets. We propose a more strict evaluation system where, in addition to cross-validation on training data, we set some datasets aside for cross-domain validation. Those holdout datasets will act as unseen accounts for selecting models with best generalizability and give us a sense of how well the models perform when facing novel types of behavior. Specifically, we use the datasets listed in Table 3, on which Botometer was trained (Yang et al. 2019), as our candidate training datasets. Botometer therefore serves as a baseline. The rest of the datasets

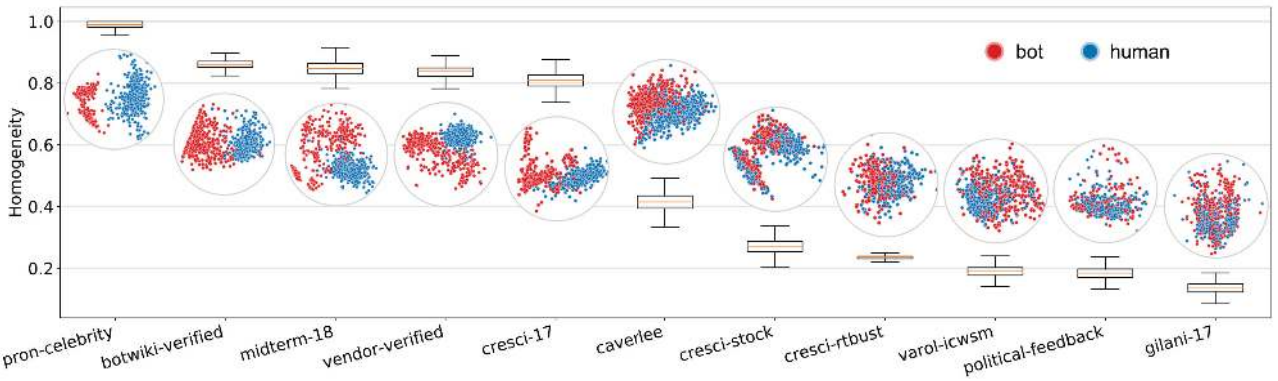

Figure 1: Visualization of human and bot accounts in different training datasets. The scatter plots visualize samples of 1,000 bot and human accounts after PCA. The box plots show the distributions of homogeneity scores across many samples from each dataset (see text for details).

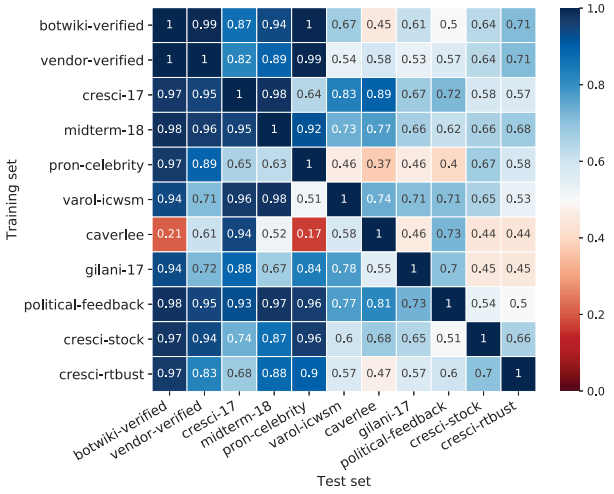

Figure 2: AUC scores of random forest classifiers trained on one dataset and tested on another. The datasets are ordered by separability (defined in the text).

are holdout accounts. The `cresci-stock` dataset is excluded from this experiment because the coordinated nature of the bots makes it unsuitable for training feature-based methods. In addition to cross-validation and cross-domain validation, we also want to evaluate generalization to a broader, more representative set of accounts. To this end we employ the 100,000 random accounts as described earlier. The random accounts lack labels, therefore we use Botometer as a reference, since it has been tested and adopted in many studies (Vosoughi, Roy, and Aral 2018; Shao et al. 2018).

Data Selection

Throwing all training data into the classifier should give us the best model according to learning theory, if the labels are correct and the accounts are independent and identically distributed in the feature space (Shalev-Shwartz and Ben-David 2014). Unfortunately, our experiments in Fig. 2 show that those assumptions do not hold in our case. This suggests that generalization might be improved by training on a selected subset of the data.

Our goal is to find a set of training accounts that optimizes our three evaluation metrics: cross-validation accuracy on training data, generalization to unseen data, and consistency with a more feature-rich classifier on unlabeled data. Similar data selection methods have proven effective in other domains with noisy and contradictory data (Wu, Zhang, and Rudnicky 2007; Erdem et al. 2010; Zhang et al. 2014).

We treat every dataset listed in Table 3 as a unit and end up with 247 possible combinations with both human and bot accounts present. Random forest classifiers with 100 trees are trained on those 247 combinations, yielding as many candidate models. We record the AUC of each model via five-fold cross-validation. The classifiers are then applied to unseen datasets (see list in Table 4) and random accounts. Results are reported in Fig. 3. Cross-validation yields high AUC for most of the models, as expected. For unseen accounts, datasets with high homogeneity tend to yield better AUC, whereas performance is worse on datasets with lower homogeneity. Most of the models have positive correlation with Botometer on random accounts. Botometer performs well in all AUC tests, although some candidate models beat Botometer in each test. No candidate model, however, outperforms the baseline on all unseen datasets.

To select a model that performs consistently well in all tests, we first rank the models in each of the six tests shown in Fig. 3 based on their performance, with the top model first. We then select the model with minimal product of the six ranks. Models selected in this way may not achieve the best results in every single test, but will do well in all tests, ensuring stability in applications. The datasets selected by the top 3 models are shown in Table 3; `M196` is the best model. Detailed performance metrics are reported in Table 4.

The winning models achieve very high AUC in cross-validation as well as on unseen datasets. Further analysis reveals that the winning models all have much higher precision than the baseline in all holdout tests at the 0.5

Table 3: Datasets used to train selected candidate models and Botometer. M196, M125 and M191 are the top three models according to our selection method.

Dataset	Botometer	M196	M125	M191	M246
caverlee	✓				✓
varol-icwsm	✓	✓	✓	✓	✓
cresci-17	✓	✓	✓	✓	✓
pronbots	✓				✓
celebrity	✓	✓	✓	✓	✓
vendor-purchased	✓			✓	✓
botometer-feedback	✓	✓	✓	✓	✓
political-bots	✓	✓			✓

Table 4: AUC scores on unseen datasets, five-fold cross-validation AUC, and correlation with Botometer for selected candidate models. Performance of the Botometer baseline and M246 (trained on all data) is shown for comparison. Metrics significantly better than the baseline ($p < 0.01$) are highlighted in bold.

Metric	Botometer	M196	M125	M191	M246
botwiki-verified	0.92	**0.99**	**0.99**	**0.99**	0.91
midterm-18	0.96	**0.99**	**0.99**	**0.98**	0.83
gilani-17	0.67	**0.68**	**0.69**	**0.68**	0.64
cresci-rtbust	0.71	0.60	0.59	0.58	0.57
5-fold cross-validation	0.97	**0.98**	**0.98**	**0.98**	0.95
Spearman's r	1.00	0.60	0.60	0.62	0.60

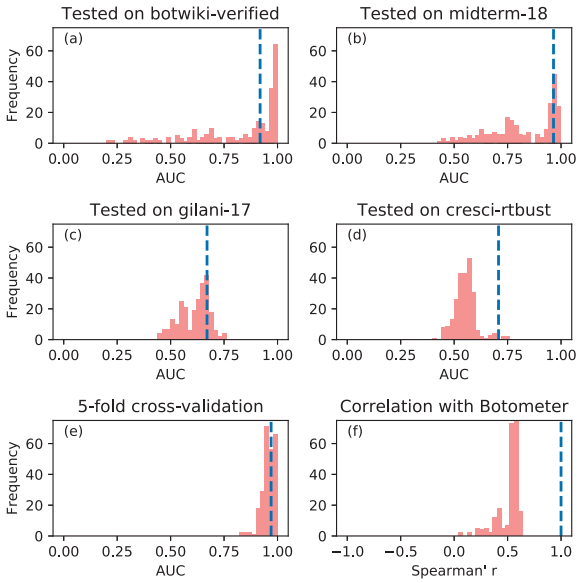

Figure 3: (a-d) AUC distributions of candidate models tested on 4 unseen datasets. (e) Five-fold cross-validation AUC distribution of candidate models. (f) Distribution of Spearman's rank correlation coefficients between candidate models and Botometer on 100,000 random accounts. Baselines from Botometer are highlighted by dashed blue lines.

threshold. Those models also achieve better recall exept for gilani-17 and cresci-rtbust. Accounts in these two datasets are annotated with different types of behaviors. In fact, when we train random forest models on these two datasets alone with five-fold cross-validation, we obtain

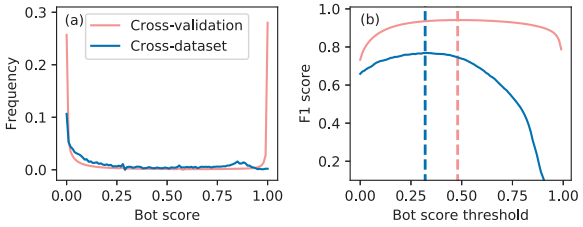

Figure 4: (a) Bot score distributions and (b) F_1 score versus bot score threshold, for the M196 model. The best F_1 is obtained with thresholds of 0.48 and 0.32 for cross-validation and cross-domain testing, respectively (dashed lines).

AUC scores of 0.84 and 0.87 respectively, indicating challenging classification tasks. Botometer's generalization performance is also lower on these two datasets. All things considered, the generalizability of the winning models seems acceptable.

Random forest generates a score between 0 and 1 to estimate the likelihood of an account exhibiting bot-like behavior. If we need a binary classifier, we can use a threshold. Fig. 4 illustrates the thresholds that maximize precision and recall (via the F_1 metric). Different thresholds yield best accuracy in cross-validation ($F_1 = 0.94$, $R = 0.93$, $P = 0.94$) or cross-domain testing ($F_1 = 0.77$, $R = 0.68$, $P = 0.88$), underscoring that the choice of threshold depends on the holdout datasets used for validation. Depending on the topic of interest, practitioners can choose or create new datasets of annotated accounts for cross-domain validation and use the rest for training data selection.

Scalability

In this section, we quantify the scalability of the proposed framework. The user metadata required for classification can be obtained through the users lookup endpoint or the streaming API, since every tweet carries the user object. Users lookup allows checking 8.6M accounts per day with a user API key. The maximum streaming volume is provided by the Firehose, which delivers all public tweets — currently 500 millions per day on average (Fedoryszak et al. 2019).

We conducted an offline experiment to estimate the classification speed. Our classifier was implemented in Python with scikit-learn (Pedregosa et al. 2011) and run on a machine with an Intel Core i7-3770 CPU (3.40GHz) and 8GB RAM. It takes an average of $9{,}612 \pm 6 \times 10^{-8}$ seconds to evaluate each tweet, which means almost 900M tweets per day, well beyond the Firehose volume.

Model Interpretation

With only 20 features in our best model (M196), it is possible to interpret its logic using the SHAP model explanation technique (Lundberg, Erion, and Lee 2018). Fig. 5 shows that, for example, long screen names (red) have positive SHAP values, meaning more bot-like. Conversely, verified accounts (red) have negative SHAP values, meaning more human-like. The SHAP analysis tells us that high favorites

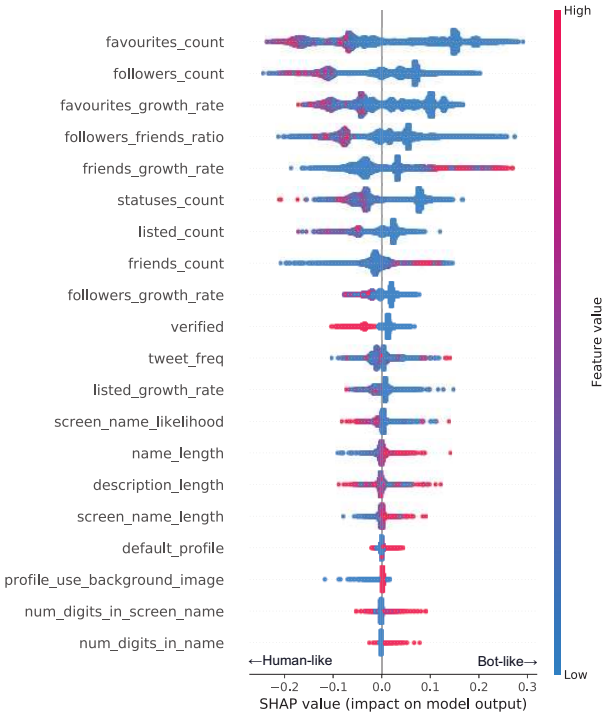

Figure 5: The summary plot generated by SHAP for model `M196`. The features are ordered by importance from top to bottom. The x-axis shows the SHAP value of each feature. A positive SHAP value means the feature is pushing the result to 1 (bot-like), a negative SHAP value means the feature is pushing the result to 0 (human-like). The feature value is indicated by color: red means the feature value is large, blue means it's small. Binary features are coded so that true is represented by 1.

and followers counts, favorites growth rate, and followers-friends ratio are signs of organic accounts. High friends count and growth rate, however, suggest an account is suspicious. This confirms the intuition that bots are eager to expand their friend lists, but do not attract as many followers.

Naturally, Fig. 5 presents an over-simplified picture of the effects of different features. The actual logic is more complex due to interactions among features. For example, a low favourites count generally means bot-like, but if the account also has a high follower growth rate, then the effect of the favorites count is reduced. Moreover, the interpretation depends on the specific data used to train the model. Yet, this example illustrates an advantage of simplified feature sets.

Conclusion

We proposed a bot detection framework that scales up to real-time processing of the full public Twitter stream and that generalizes well to accounts outside the training data. Analysis of a rich collection of labeled data reveals differences in separability and generalization, providing insights into the complex relationship among datasets. Instead of training on all available data, we find that a subset of training data can yield the model that best balances performance on cross-validation, cross-domain generalization, and consistency with a widely adopted reference. Thanks to the simplicity of the model, we are able to interpret the classifier. The scalability opens up new possibilities for research involving analysis on massive social media data. Our framework can also be embedded in other algorithms or systems for detecting more complex adversarial behaviors (Hui et al. 2019), such as coordinated influence campaigns, stock market manipulation, and amplification of misinformation.

Our work shows that data selection is a promising direction for dealing with noisy training data. Developing smarter algorithms that can perform fine-grained data selection for better performance is an open challenge for future work.

Acknowledgments. We thank Emilio Ferrara for suggesting the botwiki.org archive. K.-C. Y. and F. M. were supported in part by Craig Newmark Philanthropies. P.-M. H. and F. M. were supported in part by DARPA contract W911NF-17-C-0094.

References

Alothali, E.; Zaki, N.; Mohamed, E. A.; and Alashwal, H. 2018. Detecting social bots on twitter: A literature review. In *IEEE Intl. Conf. on Innovat. in Inf. Tech. (IIT)*, 175–180.

Berger, J. M., and Morgan, J. 2015. The ISIS Twitter Census: Defining and describing the population of ISIS supporters on Twitter. *The Brookings Project on US Relations with the Islamic World* 3(20):4–1.

Beskow, D. M., and Carley, K. M. 2019. Its all in a name: detecting and labeling bots by their name. *Comp. and Math. Org. Theory* 1–12.

Bessi, A., and Ferrara, E. 2016. Social bots distort the 2016 US presidential election online discussion. *First Monday* 21(11).

Broniatowski, D. A.; Jamison, A. M.; Qi, S.; AlKulaib, L.; Chen, T.; Benton, A.; Quinn, S. C.; and Dredze, M. 2018. Weaponized health communication: Twitter bots and russian trolls amplify the vaccine debate. *Am. J. of Public Health* 108(10):1378–1384.

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

Chen, Z., and Subramanian, D. 2018. An unsupervised approach to detect spam campaigns that use botnets on Twitter. Preprint 1804.05232, arXiv.

Cresci, S.; Di Pietro, R.; Petrocchi, M.; Spognardi, A.; and Tesconi, M. 2016. Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intel. Sys.* 31(5):58–64.

Cresci, S.; Di Pietro, R.; Petrocchi, M.; Spognardi, A.; and Tesconi, M. 2017. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proc. 26th Intl. WWW Conf. Companion*, 963–972.

Cresci, S.; Di Pietro, R.; Petrocchi, M.; Spognardi, A.; and Tesconi, M. 2018a. Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling. *IEEE Trans. on Depend. and Secur. Comp.* 15(4):561–576.

Cresci, S.; Lillo, F.; Regoli, D.; Tardelli, S.; and Tesconi, M. 2018b. $ fake: Evidence of spam and bot activity in stock microblogs on twitter. In *12th Intl. AAAI Conf. on Web and Soc. Media (ICWSM)*.

Cresci, S.; Lillo, F.; Regoli, D.; Tardelli, S.; and Tesconi, M. 2019. Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on twitter. *ACM TWEB* 13(2):11.

Davis, C. A.; Varol, O.; Ferrara, E.; Flammini, A.; and Menczer, F. 2016. BotOrNot: A system to evaluate social bots. In *Proc. 25th Intl. WWW Conf. Companion*, 273–274.

De Cristofaro, E.; Kourtellis, N.; Leontiadis, I.; Stringhini, G.; Zhou, S.; et al. 2018. Lobo: Evaluation of generalization deficiencies in twitter bot classifiers. In *Proc. 34th ACM Ann. Comp. Secur. Applic. Conf.*, 137–146.

Deb, A.; Luceri, L.; Badaway, A.; and Ferrara, E. 2019. Perils and challenges of social media and election manipulation analysis: The 2018 us midterms. In *Companion Proc. WWW Conf.*, 237–247.

Erdem, C. E.; Bozkurt, E.; Erzin, E.; and Erdem, A. T. 2010. Ransac-based training data selection for emotion recognition from spontaneous speech. In *Proc. 3rd Intl. Workshop Affect. Interac. in Nat. Environments*, 9–14.

Fedoryszak, M.; Frederick, B.; Rajaram, V.; and Zhong, C. 2019. Real-time event detection on social data streams. In *Proc. 25th ACM SIGKDD Intl. Conf. on Knowl. Discov. & Data Min.*, 2774–2782.

Ferrara, E.; Varol, O.; Davis, C.; Menczer, F.; and Flammini, A. 2016a. The rise of social bots. *CACM* 59(7):96–104.

Ferrara, E.; Wang, W.-Q.; Varol, O.; Flammini, A.; and Galstyan, A. 2016b. Predicting online extremism, content adopters, and interaction reciprocity. In *Proc. Intl. Conf. on Soc. Info.*, 22–39.

Ferrara, E. 2017. Disinformation and social bot operations in the run up to the 2017 french presidential election. *First Monday* 22(8).

Gilani, Z.; Farahbakhsh, R.; Tyson, G.; Wang, L.; and Crowcroft, J. 2017. Of bots and humans (on twitter). In *Proc. IEEE/ACM Intl. Conf. Adv. in Soc. Netw. Anal. and Min.*, 349–354.

Gilani, Z.; Kochmar, E.; and Crowcroft, J. 2017. Classification of twitter accounts into automated agents and human users. In *Proc. IEEE/ACM Intl. Conf. Adv. in Soc. Netw. Anal. and Min.*, 489–496.

Grimme, C.; Preuss, M.; Adam, L.; and Trautmann, H. 2017. Social bots: Human-like by means of human control? *Big data* 5(4):279–293.

Hui, P.-M.; Yang, K.-C.; Torres-Lugo, C.; Monroe, Z.; McCarty, M.; Serrette, B.; Pentchev, V.; and Menczer, F. 2019. Botslayer: real-time detection of bot amplification on twitter. *Journal of Open Source Software* 4(42):1706.

Jiang, M.; Cui, P.; Beutel, A.; Faloutsos, C.; and Yang, S. 2016. Catching synchronized behaviors in large networks: A graph mining approach. *ACM TKDD* 10(4):35.

Kudugunta, S., and Ferrara, E. 2018. Deep neural networks for bot detection. *Inf. Sci.* 467(October):312–322.

Lee, K.; Eoff, B. D.; and Caverlee, J. 2011. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. In *Proc. AAAI Intl. Conf. on Web and Soc. Media (ICWSM)*.

Lundberg, S. M.; Erion, G. G.; and Lee, S.-I. 2018. Consistent individualized feature attribution for tree ensembles. Preprint 1802.03888, arXiv.

Mazza, M.; Cresci, S.; Avvenuti, M.; Quattrociocchi, W.; and Tesconi, M. 2019. Rtbust: Exploiting temporal patterns for botnet detection on twitter. Preprint 1902.04506, arXiv.

Minnich, A.; Chavoshi, N.; Koutra, D.; and Mueen, A. 2017. Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proc. IEEE/ACM Intl. Conf. Adv. in Soc. Netw. Anal. and Min.*, 467–474.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; et al. 2011. Scikit-learn: Machine learning in Python. *J. of Mach. Learning Res.* 12:2825–2830.

Rosenberg, A., and Hirschberg, J. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proc. Joint Conf. Empiric. Methods in Nat. Lang. Proc. and Comp. Nat. Lang. Learning (EMNLP-CoNLL)*.

Shalev-Shwartz, S., and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shao, C.; Ciampaglia, G. L.; Varol, O.; Yang, K.-C.; Flammini, A.; and Menczer, F. 2018. The spread of low-credibility content by social bots. *Nat. Commun.* 9(1):4787.

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

Subrahmanian, V.; Azaria, A.; Durst, S.; Kagan, V.; Galstyan, A.; Lerman, K.; Zhu, L.; Ferrara, E.; Flammini, A.; Menczer, F.; et al. 2016. The DARPA Twitter bot challenge. *Computer* 49(6):38–46.

Varol, O.; Ferrara, E.; Davis, C. A.; Menczer, F.; and Flammini, A. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *Proc. Intl. AAAI Conf. on Web and Soc. Media (ICWSM)*.

Vosoughi, S.; Roy, D.; and Aral, S. 2018. The spread of true and false news online. *Science* 359(6380):1146–1151.

Wu, Y.; Zhang, R.; and Rudnicky, A. 2007. Data selection for speech recognition. In *IEEE Workshop Autom. Speech Recognit. & Underst. (ASRU)*, 562–565.

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. *Hum. Behav. and Emerg. Tech.* 1(1):48–61.

Yang, K.-C.; Hui, P.-M.; and Menczer, F. 2019. Bot electioneering volume: Visualizing social bot activity during elections. In *Companion Proc. WWW*, 214–217.

Zhang, Z.; Eyben, F.; Deng, J.; and Schuller, B. 2014. An agreement and sparseness-based learning instance selection and its application to subjective speech phenomena. In *Proc. 5th Intl. Workshop Emotion Soc. Signals, Sentiment & Linked Open Data (ES3LOD)*.