FINAL PROJECT: BOT DETECTION AND ANALYSIS

Mengfei Du
18307130148@fudan.edu.cn

Guochao Jiang
18307110234@fudan.edu.cn

Jingcong Liang
18307110286@fudan.edu.cn

ABSTRACT

Online social networks are tended to be polarize into several communities with opposite perspectives for some controversial events. However, humans are not the only participants and computer-controlled accounts: bots are also taking parts.After analyzing large-scale social data collected during the 2020 American President Election with our neural-network-based bot detection system, we show that some social bots can play an important role in the information communication between two opposite communities. Our findings may give a different aspect of the influence of bots.

1 Introduction

Bots (computer-controlled accounts) exist in vast quantities in online social networks. For different purposes e.g. marketing, political infiltration, spamming, fun etc., different bots are produced. The rise of Twitter bots are evidenced by some researches[10]. Similarly to human interactions, bots might be able to affect the social structure and function of a social system[2]. To deal with this phenomenon, a growing body of research is addressing bot activity, its implications on the social network, and the detection of these accounts.

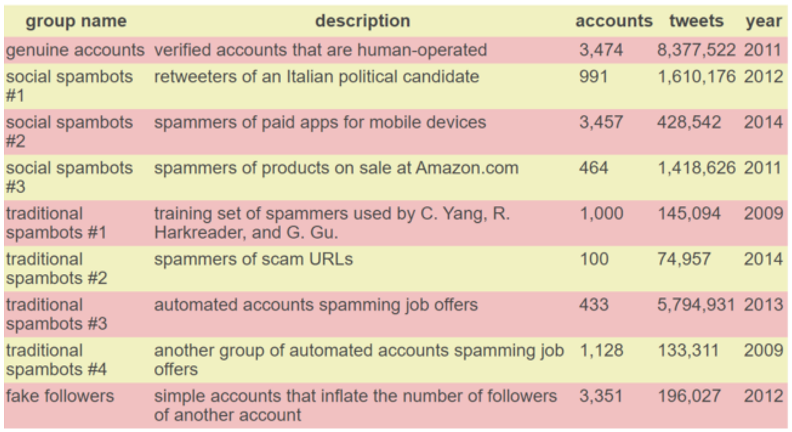
In this project, we try to design dot detection systems by traditional feature engineering and neural network and analyze the effect of bots in American President Election retweet network. This project makes the following contributions:

- We use Twitter API to crawl users' tweet and profile data during the American President Election and construct a retweet-based social network.
- We proposed feature-based model and neural-network-based end-to-end model of binary classification and multi-classification tasks respectively using a public data set and get a 0.99 AUC score.
- We use our neural-network-based model to detect the bot users in our retweet-based network and analyze the role they play during the controversial social event.

2 Related Work

Most bot detection methods are based on feature engineering with supervised machine learning. Davis et al.[8] proposed a framework using user, temporal, content, and social network features and random forest classifiers to detect bots. Varol et al.[16] analyzed the importance of features for bot detection and found the metadata of users is the most important. Yang et al.[19] simplified the process of feature extraction. They only use users profile information and some indicators like rate to construct machine learning model and get a good performance.Some researchers also try to use unsupervised learning to detect bot accounts. Chavoshi et al.[4] developed a novel lag-sensitive hashing technique to cluster user accounts into correlated sets in near real-time. Mazza et al.[14] modeled the time series of users' retweet behaviours and clustered the representations to get the bot clusters.

There also exists many researches to analyze the effect of bots in online social networks. Boshmaf et al.[3] stated the vulnerability of social media users to a social botnet designed to expose private information, like phone numbers and addresses. Davis et al.[8] found that bots contribute to the strong polarization of political discussion observed in social media. Kramer et al.[13] stated that emotions are contagious on social media. And elusive bots could easily infiltrate a population of unaware humans and manipulate them to affect their perception of reality, with unpredictable results[17]. Stella et al.[15] proposed bots can increase exposure to negative and inflammatory content in online social systems.



group name	description	accounts	tweets	year
genuine accounts	verified accounts that are human-operated	3,474	8,377,522	2011
social spambots #1	retweeters of an Italian political candidate	991	1,610,176	2012
social spambots #2	spammers of paid apps for mobile devices	3,457	428,542	2014
social spambots #3	spammers of products on sale at Amazon.com	464	1,418,626	2011
traditional spambots #1	training set of spammers used by C. Yang, R. Harkreader, and G. Gu.	1,000	145,094	2009
traditional spambots #2	spammers of scam URLs	100	74,957	2014
traditional spambots #3	automated accounts spamming job offers	433	5,794,931	2013
traditional spambots #4	another group of automated accounts spamming job offers	1,128	133,311	2009
fake followers	simple accounts that inflate the number of followers of another account	3,351	196,027	2012

Figure 1: Description for training dataset

Based on the current researches, with the knowledge of neural network, we propose a neural-network-based model together with features to detect bots and try to analyze the role that bots play in a information network about a controversial social event.

3 Dataset

3.1 Training Data

To train and test our model, we collect a public dataset of labeled human and bot accounts[7]. The dataset includes four types of accounts: genuine (human), social spam, traditional spam, and fake follower. The proportion of the four accounts are as Figure 1.

3.2 Election Data

Based on the situation of 2020 American President Election, we choose the time window from Nov. 1 to Nov. 7. According to the 2020 U.S. election tweets dataset[5], we use Twitter API to randomly crawl 20,000 tweets each hour during the time window. Finally, we have constructed a dataset with 2,075,936 tweets and 1,122,863 accounts.

4 Feature

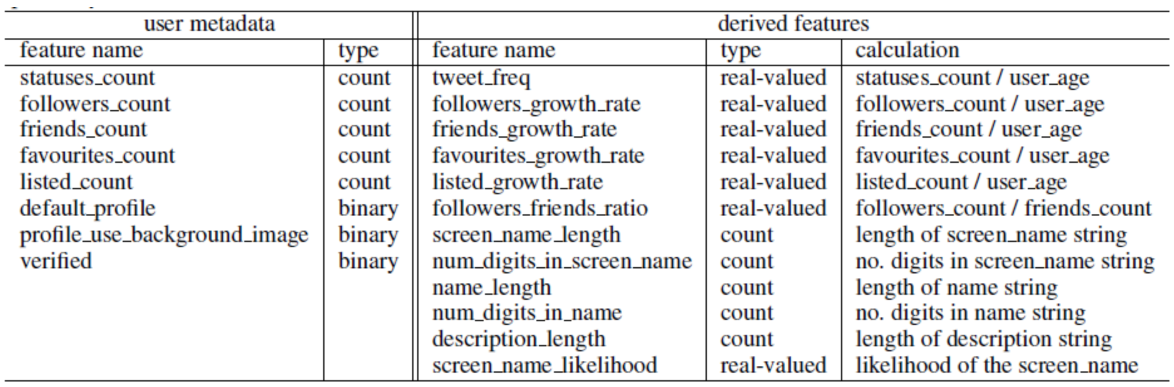
For a Twitter user, he has a lot of basic information such as the number of followers and the number of friends. In order to be able to predict whether the user is a robot better, it is necessary to obtain some extended information data from these basic information data, then combine the two, and extract some important features from them. On the one hand, it can reduce the number of features to avoid over-fitting, and on the other hand, it can reduce the amount of data.

Inspired by Yang et al.[19], the feature group shown in Figure 2 can be used to detect robot users. In order to save time and speed up model training, the LightGBM model is used in the feature selection part.

4.1 LightGBM

GBDT (Gradient Boosting Decision Tree) is a long-lasting model in machine learning. Its main idea is to use some weak classifiers (decision trees) to iteratively train to obtain the optimal model. This model has good training effects and is not easy to over-fit. GBDT is not only widely used in the industry, it is usually used for tasks such as multi-classification, click-through rate prediction, search ranking. Moreover, it is also a lethal weapon in various data mining competitions. LightGBM (Light Gradient Boosting Machine) is a framework that implements the GBDT algorithm, supporting high-efficiency parallel training. There are many advantages like faster training speed, lower memory consumption, better accuracy, distributed style.

In one iteration step, GBDT needs to traverse the entire training data multiple times. If the entire training data is loaded into the memory, the size of the training data will be limited. If it is not loaded into the memory, it will consume a lot of time to read and write the training data repeatedly. Especially in the face of industrial-grade massive data, the ordinary



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

Figure 2: Generalizable social bot features

GBDT algorithm can not meet needs. The main reason LightGBM put forward is to solve the problems encountered by GBDT in massive data so that GBDT can be used in industrial practice better and faster.

LightGBM is a decision tree algorithm based on histogram. A common histogram-based decision tree algorithm discretizes continuous floating-point feature values into a series of integers. At the same time it constructs a histogram with a suitable width. When traversing the data, the discretized value is used as an index to accumulate statistics in the histogram. After traversing the data once, we can find the optimal split point according to the discrete value of the histogram.

LightGBM has made many optimizations here, the first is histogram difference acceleration. The histogram of a leaf can be obtained by the difference between the histogram of its parent node and the histogram of its siblings. Using this prior knowledge can double the speed of training. And it uses only non-zero feature values to construct the histogram.

The second is that it abandons the level-wise decision tree growth strategy used by most GBDT tools and uses a leaf-wise algorithm with depth restrictions. Compared with level-wise, the advantage of leaf-wise is that it can reduce errors and get better accuracy under the same number of splits. The disadvantage is that it may grow a deeper decision tree, resulting in over-fitting. Therefore, LightGBM adds depth limitation on the basis of leaf-wise to ensure high efficiency while preventing over-fitting.

The third is that it uses GOSS (Gradient-based One-Side Sampling). From the perspective of reducing samples, the GOSS algorithm excludes most of the samples with small gradients and only uses the remaining samples to calculate the information gain. It is an algorithm that reduces the amount of data and ensures the accuracy.

The fourth is that it uses the EFB (Exclusive Feature Bundling) algorithm. High-dimensional data is often sparse. This sparseness inspired us to design a lossless method to reduce the dimension of features. Usually the bundled features are mutually exclusive. If there are two features bundled together, no information will be lost. If two features are not completely mutually exclusive, an index can be used to measure the degree of feature non-exclusion, which is called the conflict ratio. When the conflict ratio is small, we can choose to bundle two features that are not completely mutually exclusive without affecting the final accuracy. EFB pointed out that if some features are merged and bound, the number of features can be reduced. In this way, the time complexity of constructing the histogram will be greatly reduced.

4.2 Feature Selection and Bot Detection

For decision tree algorithms, the importance of model features can often be directly calculated after model training. Feature importance can measure the importance of each feature. Therefore, the feature importance in the decision tree algorithm is used as the index of feature screening. So several features with high feature importance are selected. Calling Python's `lightgbm` package, we can directly use functions for model training and evaluation.

Table 1 and Figure 3 show the training evaluation results and feature importance of the LightGBM model.

From the results, it is not difficult to find that there are about 11 selected features with high importance. These features are also very relevant to robot detection from an intuitive point of view such as `verified`.

The above-mentioned training model is based on two classifications, that is, a classification model of real users and robot users. So the training results can be used for classification. Using the same method to construct test set, Table 1 shows different model evaluation results. It is obvious that our model is better than other models. But our model does

Table 1: Models Comparison

Methods	Detection Results			
	Precision	Recall	Accuracy	F_1 Score
Twitter Countermeasures	**1.000**	0.094	0.691	0.171
Human Annotators	0.267	0.080	0.698	0.123
Cresci et al.[6]	0.982	0.972	0.976	0.977
Ahmed and Abulaish[1]	0.945	0.944	0.943	0.944
Our Model	0.996	**0.989**	**0.996**	**0.992**

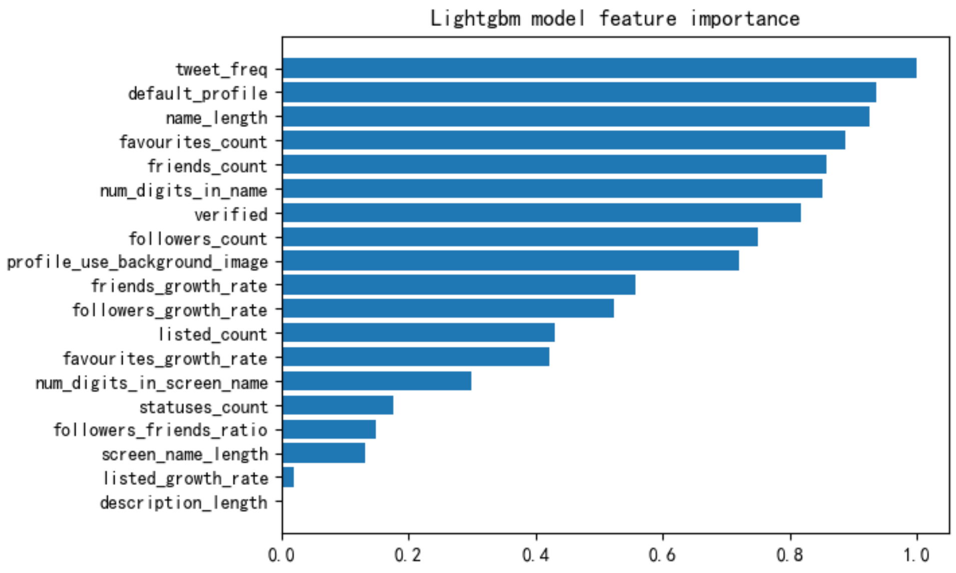


Figure 3: LightGBM model feature importance

not use the user's tweet data. User tweets are also a very important dimensional data for detecting robot users. So next step we will process user tweets and combine the extracted features to perform robot user detection.

5 Neural-network-based Bot Detection

To furthermore incorporate tweet information, it is necessary to digest the contents in a tweet as well as all tweets by a single user. Hence, we also propose a neural-network-based bot detection model, which considers both user features and tweet features.

5.1 Model Structure

The model consists of three major parts. First, the text of each tweet is converted into a tweet vector by BERT and concatenated with other tweet content and social features; in this way we generate a vector sequence to represent a user's tweet timeline, where tweet vectors are arranged chronologically. Next, the sequence is processed by a Bi-LSTM, whose last output is chosen as the representation vector of a user's all tweets. Finally, this vector is again concatenated with user features selected in section 4, and then passed through two linear layers and one softmax layer to obtain the final classification results.

In the first part, we introduce BERT[9] to create feature vectors based on tweet texts. BERT is a language model capable to provide word and sentence vector representation from a given sentence or a sentence pair. Since the length of a tweet is limited to 280 characters, we can directly apply BERT to obtain the feature vector of the entire tweet text. After converting the tweet text into a tweet vector, extra tweet features are included to finish a full tweet representation. These extra tweet features include content features (number of mentioned users, hashtags and links) and social features (post time since user being created, total retweeted times and number of favorites by other users).

With the above process, for each user we now have a chronological tweet sequence. This sequence is passed into a Bi-LSTM to comprehend the user's tweet behaviors and habits. Although LSTM outputs a feature for each time step, we only take its last output as the user's tweet feature, as it contains information across the whole tweet timeline. This tweet feature then incorporates with the selected user features. Only the most important features mentioned in section 4 are involved. Finally, this hybrid feature is converted into classification results by linear and softmax layers. The final output are probabilities of the four classes for both balancing sample sizes and providing finer classification results on bots.

5.2 Implementation Details

In order to reduce model training time, we use `bert-as-service`[18] to calculate vector representations of tweet texts with a pretrained multi-lingual BERT model. We freeze the pretrained model during the training process, and no further fine-tuning is applied. The dimension of a tweet vector is 768. This vector is then fed into a Bi-LSTM with hidden dimension 1024 after batch normalizing. Next, the last output of the LSTM is concatenated with user features and fed into the linear layers. Each linear layer consists of a batch normalization, a fully-connected layer and a ReLU activator. The hidden sizes of linear layers are 256 and 64 respectively. Finally, the 64-D output is passed through a batch normalization layer and a fully-connected softmax layer to obtain probabilities of the four classes.

We randomly split the training set to training, validation and test sets with a ratio of $8 : 1 : 1$. The optimization target is to minimize the cross-entropy of all samples in the training set, and SAM[11] with Adam[12] as the base optimizer is used to optimize the model for regularization. The initial learning rate of Adam is 10^{-4} (we keep this unchanged during training), with a weight decay of 10^{-5}. In order to prevent gradient explosion, we set the gradient norm clipping threshold to 1. We train the model until validation loss does not improve for 10 epochs.

5.3 Experiment Results

Table 2 illustrates classification results of the model on the test set. We also measured the accuracy and AUC of the model. The total accuracy of the model is 0.95, while the AUC could reach 0.99. From Table 2 we can see that the model has excellent performance in detecting fake followers and social spam bots, while having some confusion between genuine accounts and traditional spam bots.

Table 2: Neural-network-based Bot Detection Results

Class	Precision	Recall	F_1 Score
Genuine Accounts	0.94	0.89	0.92
Social Spam Bots	1.00	0.99	0.99
Traditional Spam Bots	0.84	0.93	0.88
Fake Followers	0.99	0.96	0.98
Average	0.94	0.94	0.94

6 Network Analysis

To analyze the effect of bots during the process of information exchange between the two communities with different political opinions. According to the observed social interactions in retweets (who re-shares the content posted by whom),we construct a network for 2020 U.S with 8353 nodes and 103851 edges.By using community partition algorithm with information dissemination, we divide the nodes into two groups.The result is demon in Figure 4.

Based on the tweet content we can get the blue group supports democratic party while red group supports republican party. The proportion of different types of accounts are listed in Table 3.

By observing the location of bots, we can find that Social spam mainly exists in three parts of the network: The bots in network center has a high degree and is a transit point for the dissemination of opinion leaders' tweets. Some social spams act as a bridge between the two communities. The retweet actions of bot are mainly controlled by computer programs and will not be affected by political tendencies. To a certain extent, the two opposing groups are contacted for information exchange by these bots. And some bots in the edge of the network are no longer retweeted by others and have a decreasing importance.

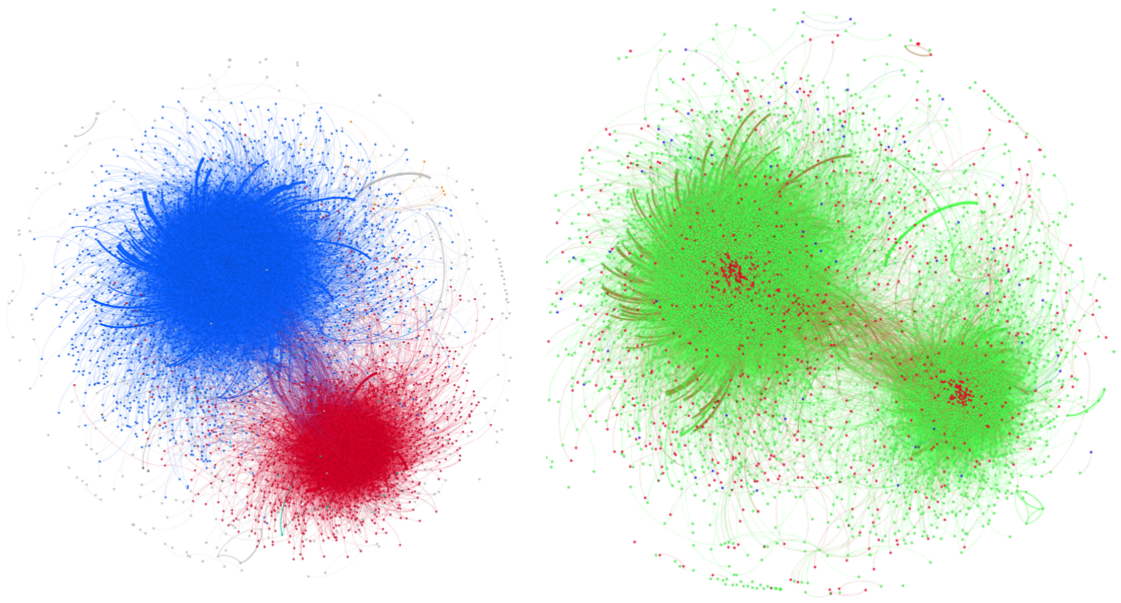


Figure 4: Network of Retweet Relations:The left figure is the two communities of users. The right figure is the network with four types of accounts (green for human, red for social spam, blue for fake follower and black for traditional spam)

Table 3: Proportion of different types of accounts

Type	Number	Proportion
Human	7408	88.80%
Social Spam	839	10.04%
Fake Follower	61	0.73%
Traditional Spam	45	0.54%

These findings can also be proved by betweenness centrality of social spams. We can find the proportion of bot-human edge between two communities is up to 53.9%. The mean of social spams' betweenness centrality is 1.25×10^{-3} which is much larger than human's 1.91×10^{-4}.

7 Conclusion

Diverse types of bots in social media create challenges for traditional bot detection systems making inferences and predictions on online data. Based on the existing researches, We proposed two bot detection systems: one based on traditional feature engineering and the other based on LSTM. We compare our models with some existing method and the performance of our models in some metrics is better.

According to the results of our neural-network-based model, we analyze the effect of bots in the retweet network of 2020 U.S. election. Our findings support the hypothesis that bots may influence information diffusion in social media systems[2], and it can serve as a bridge between two opposite social community.

Due to the current field has few researches for multi-category bot detection, we need to do more comparing experiments to evaluate performance of our system. We also try to use multi-language sentiment analysis, network analysis and semantic network analysis to explore the effect of bots from comprehensive perspective.

Division of Work

- Mengfei Du: Data crawl , some machine learning codes writing and network analysis.
- Guochao Jiang: LightGBM model implementation and feature selection.
- Jingcong Liang: Neural network design and implementation.

References

[1] Faraz Ahmed and Muhammad Abulaish. A generic statistical approach for spam detection in online social networks. *Computer Communications*, 36(10-11):1120–1129, 2013.

[2] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. *First monday*, 21(11-7), 2016.

[3] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. Design and analysis of a social botnet. *Computer Networks*, 57(2):556–578, 2013.

[4] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. Debot: Twitter bot detection via warped correlation. In *Icdm*, pages 817–822, 2016.

[5] Emily Chen, Ashok Deb, and Emilio Ferrara. # election2020: the first public twitter dataset on the 2020 us presidential election. *Journal of Computational Social Science*, pages 1–18, 2021.

[6] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, 31(5):58–64, 2016.

[7] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 963–972. International World Wide Web Conferences Steering Committee, 2017. ISBN 978-1-4503-4914-7. doi: 10.1145/3041021.3055135. URL https://doi.org/10.1145/3041021.3055135.

[8] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. Botornot: A system to evaluate social bots. In *Proceedings of the 25th international conference companion on world wide web*, pages 273–274, 2016.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[11] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[14] Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM Conference on Web Science*, pages 183–192, 2019.

[15] Massimo Stella, Emilio Ferrara, and Manlio De Domenico. Bots increase exposure to negative and inflammatory content in online social systems. *Proceedings of the National Academy of Sciences*, 115(49):12435–12440, 2018.

[16] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the international AAAI conference on web and social media*, volume 11, 2017.

[17] Xian Wu, Ziming Feng, Wei Fan, Jing Gao, and Yong Yu. Detecting marionette microblog users for improved information credibility. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 483–498. Springer, 2013.

[18] Han Xiao. bert-as-service. https://github.com/hanxiao/bert-as-service, 2018.

[19] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1096–1103, 2020.