Proceedings of the Fifteenth International AAAI Conference on Web and Social Media (ICWSM 2021)

Posting Bot Detection on Blockchain-based Social Media Platform using Machine Learning Techniques

Taehyun Kim,[1] Hyomin Shin,[1] Hyung Ju Hwang,[1*] Seungwon Jeong[2*]
[1]Pohang University of Science and Technology
[2]University of Bristol
{taehyun3401,zhainl,hjhwang}@postech.ac.kr, eugene.jeong@gmail.com

Abstract

Steemit is a blockchain-based social media platform, where authors can get author rewards in the form of cryptocurrencies called STEEM and SBD (Steem Blockchain Dollars) if their posts are upvoted. Interestingly, curators (or voters) can also get rewards by voting others' posts, which is called a curation reward. A reward is proportional to a curator's STEEM stakes. Throughout this process, Steemit hopes "good" content will be automatically discovered by users in a decentralized way, which is known as the Proof-of-Brain (PoB). However, there are many bot accounts programmed to post automatically and get rewards, which discourages real human users from creating good content. We call this type of bot a posting bot. While there are many papers that studied bots on traditional centralized social media platforms such as Facebook and Twitter, we are the first to study posting bots on a blockchain-based social media platform. Compared with the bot detection on the usual social media platforms, the features we created have an advantage that posting bots can be detected without limiting the number or length of posts. We can extract the features of posts by clustering distances between blog data or replies. These features are obtained from the Minimum Average Cluster from Clustering Distance between Frequent words and Articles (MAC-CDFA), which is not used in any of the previous social media research. Based on the enriched features, we enhanced the quality of classification tasks. Comparing the F_1-$scores$, the features we created outperformed the features used for bot detection on Facebook and Twitter.

Introduction

Despite the interest in blockchain technology, the usage of the so-called *decentralized application* (DApp) is still limited. Except for transferring and trading cryptocurrencies, one of the most widely used applications is *Steemit*,[1] a blockchain-based social media platform. Based on a DApp ranking site,[2] on January 2020, Steemit had ranked first among all DApps for a long time, and it still ranks sixth, and most DApps with high ranks are based on the *Steem* blockchain (Steemit 2017), on which Steemit also runs.

On Steemit, authors get *author rewards* in the form of cryptocurrencies called STEEM and SBD (Steem Blockchain Dollars) if their posts are upvoted. Interestingly, curators (or voters) also get rewards by voting others' posts, which is called a *curation reward*. A user is an author if she writes a post, and is a curator if she votes a post (including her own posts). Rewards are proportional to a curator's staked amount of STEEMs, which is called STEEM POWER.[3] That is, an *upvote* from a user with more STEEM POWER has a higher value. Each vote consumes a *voting power*, which is regenerated as time goes by. There is also a *downvote* which decreases the reward of a post, which is intended to prevent spams and any malicious content. Throughout this process, Steemit hopes "good" content to be automatically discovered by users in a decentralized way, which is called the Proof-of-Brain (PoB).

However, as on other traditional social media platforms such as Facebook and Twitter, there are many bot accounts that post automatically. We call this type of bot a *posting bot*. Detection of posting bots may be more critical on Steemit than other platforms, because posting bots on Steemit also get rewards, which discourages real human users from creating good content. Due to downvoting, bots that spam frequently cannot survive in terms of rewards. Therefore, posting bots have evolved in a way that they can write more meaningful posts hence appearing like human accounts.

There are many papers that studied bots on traditional social media platforms. In particular, some studies detected posting bots on Twitter. Twitter is a microblogging site on which users post messages called Tweets. A Tweet has a 140 (or 280 since November 2017) character limit. Thus, a text in a tweet is short and relatively easy to analyze compared with other social media platforms such as Facebook and Steemit. On Steemit, there is no restriction on the length of a post, but it is limited by the block size, which is currently 64KB. This is quite enough for most social media posts.[4] Another complication of detecting bots on Steemit

*Corresponding authors.

[1]https://steemit.com
[2]https://www.stateofthedapps.com
[3]Converting STEEM to STEEM POWER is instant, but the reverse takes 4 weeks.
[4]Any media files, e.g., pictures, videos, are uploaded to a tradi-

is its high level of anonymity because of the decentralized nature of blockchain. Due to its financial rewards, relatively long texts, high level of anonymity, it is both important and challenging to detect posting bots on Steemit.

To the best of our knowledge, this study is the first to investigate posting bots on a blockchain-based social media platform. Compared with the bot detection on the traditional social media platforms, the features we created have an advantage that they can be obtained without limiting the number and length of posts. We extract the features by clustering distance between the blog data or replies. These features are obtained from the MAC-CDFA (Minimum Average Cluster from Clustering Distance between Frequent words and Articles), which has not been used in any of the previous social media research. This feature shows similarity between blog data by clustering distances between blog data. Based on the enriched features, we enhanced the classification quality. Comparing the F_1-$scores$, the features we created outperformed the features used for the bot detection on Facebook and Twitter.

Related Work

Detecting bots on social media platforms has become an important issue with the growth of social media platforms (Allcott and Gentzkow 2017; Ferrara et al. 2016). Many researchers have tried to detect bots with machine learning algorithms (Abu-El-Rub and Mueen 2019; Chu et al. 2012; Clark et al. 2016; Dickerson, Kagan, and Subrahmanian 2014; Santia, Mujib, and Williams 2019; Varol et al. 2017; Wang 2010). They have focused on extracting features that represent patterns of behavior of each account. In particular, features that represent the regularity have played a major role. Some researchers computed the similarity of texts posted by each user, and others measured an entropy of time intervals to express regularity of behavior patterns. In a recent study, (Li and Palanisamy 2019) pointed out the prevalence of a different type of bot from which users buy votes on Steemit. Most of these bots are easily found from their own advertisements or by examining transfer memos that contain the post URL to be upvoted. In contrast, our focus is a posting bot.

There have been several attempts to extract regularity of texts, especially around Twitter (Abu-El-Rub and Mueen 2019; Clark et al. 2016; Wang 2010). They used a method that considers all pairs of tweets, by defining the similarity between two tweets. Similarity between two tweets is defined in many ways. (Wang 2010) identified if a tweet is duplicated by another, using the Levenshtein distance. (Abu-El-Rub and Mueen 2019) defined the similarity using the Jaccard index of hashtags contained in each text. (Clark et al. 2016) considered the longest common sequence of two texts. However, Twitter is different from Steemit in terms of the text length. In this respect, Facebook is a good example to compare with Steemit. Users of both Steemit and Facebook can write long articles. (Santia, Mujib, and Williams 2019) tried to detect social bots on Facebook with six features including the content-based features. However, they did not extract the features that represent pairwise text similarity. Rather, they computed the innovation rate that represents the vocabulary of each account and also is used on a Twitter dataset (Clark et al. 2016). In addition, they proposed six features which are used on a Facebook dataset containing the innovation rate.

Features related with the behavioral regularity also give important information. (Chu et al. 2012) proposed an automated account detection algorithm on Twitter, which measures the entropy of tweeting time intervals. Showing the difference in the distributions of the entropy for each type of account, they emphasized that entropy measures are important for detecting automated accounts. In addition, (Chavoshi, Hamooni, and Mueen 2017) emphasized that it is important to consider temporal data to detect twitter bots. In this regard, we compute entropy from the sequence of the various activities including transfer of an account.

Other studies have tried to extract various types of features. (Dickerson, Kagan, and Subrahmanian 2014) used sentiment scores to design a social bot classifier by applying the Random Forest algorithm, using several features including the sematic metric. Feature importance extracted from Random Forest algorithm has revealed that semantic metrics play an important role in detecting social bots on Twitter. (Varol et al. 2017) extracted six different types of features: metadata of users and friends, tweet content and sentiment, network patterns, and activity time series. They highlighted the fact that human and bot accounts have diverse behavior patterns and concluded that 8 - 15 percent of accounts on Twitter are social bots.

There are different approaches to detect bots on social media platform. (Cresci et al. 2017; Feng et al. 2017; Lee and Kim 2014) defined different types of similarities used to detect social bots. (Cresci et al. 2017) defined the Digital DNA, which is the sequence of behaviors of a user, and computed the similarity of two sequences. (Lee and Kim 2014) considered the similarity of user names. In addition, (Feng et al. 2017) defined the similarity of users' relationships. Both applied each similarity to the hierarchical clustering method. (Chavoshi, Hamooni, and Mueen 2016) is a model to detect twitter bot based on the clustering method. They used the lag-sensitive hashing technique and computed the Pearson correlation of posting time series between each user. Some applied anomaly detection methods (Castellini, Poggioni, and Sorbi 2017; Minnich et al. 2017). (Castellini, Poggioni, and Sorbi 2017) extracted features and applied them to a denoising autoencoder, a deep learning algorithm, and (Minnich et al. 2017) employed an ensemble method of anomaly detection. Some studies are based on the graph structure (Cao et al. 2012; Wang, Zhang, and Gong 2017). Their approaches are based on the Random Walk (Cao et al. 2012) or Loop Belief Propagation (Wang, Zhang, and Gong 2017). (Boshmaf et al. 2015; El-Mawass, Honeine, and Vercouter 2018; Höner et al. 2017) mixed machine learning algorithms with graph based methods. They defined the similarity between users (El-Mawass, Honeine, and Vercouter 2018) or adjusted each edge weight using a machine learning method (Boshmaf et al. 2015; Höner et al. 2017). For

tional cloud service, and only links to the media are included in the post.

other studies on Steemit, see (Thelwall 2018; Casadesus-Masanell, White, and Elterman 2019; Jeong 2020).

Feature Generation

In the feature generation section, we describe the features used in the classification. The features are divided into four categories. First, we develop the CDFA group that describes the distance between frequently used words and articles. Second, (Santia, Mujib, and Williams 2019) analyzed the social bots on Facebook. Unlike Twitter, one can write a blog post on Facebook with unlimited characters, similar to Steemit. Therefore, we benchmark the features in (Santia, Mujib, and Williams 2019) and call them a Santia-2019 group. Third, (Chu et al. 2012) classified the accounts in Twitter into human, bot, cyborg using entropy rate, spam detection, and account properties. However, some of the features are not available or not meaningful to detect the posting bots on Steemit. For example, the kind of twitting device or account verification features are not available on Steemit, and spam detection is not meaningful because out of fourteen spammers, only two appear to be posting bots, and the remaining twelve are humans. Consequently, we benchmark the entropy rate and some of the account properties from (Chu et al. 2012), and we denote them as Chu-2012 group. Finally, we added more features related to blockchain in order to observe a relation between blockchain and posting bots. We denote them as blockchain-oriented feature group. In this section, we generate four groups of features. We aim to study the difference in the effect of posting bot detection between the features we created and the features in the previous study (Chu-2012, Santia-2019). Also we want to observe the difference in performances between features with and without the blockchain-oriented ones.

CDFA Group

We introduce the new features called the CDFA group to represent the characteristics of words of a given account. First, to develop the new features, we introduce a clustering method that considers a similarity between articles. Clustering Distance between Frequent words and Articles (CDFA) is a method that transforms word data into real values. We consider the frequent words used by an account and measure the distance between the frequent words and the articles written by the account.

For the given data of the m articles written by an account, to extract the frequent words, we split the articles into words with a space. Let W_j be the set of words in the j-th article, $W = \bigcup_{1\leq j\leq m} W_j$ be the set of all words used in the articles, and w_i, $1 \leq i \leq n$, be the words in W. Further, for each article, we determine whether a word w_i is used or not. Then we obtain the occurrence vectors V_j, $1 \leq j \leq m$, with the length n in which an element in each vector V_j represents the word occurrence in the j-th article. We describe the occurrence vectors as follows:

$$V_j[i] = \begin{cases} 1, & w_i \in W_j \\ 0, & w_i \notin W_j \end{cases}.$$

Next, we sum up the occurrence vectors and obtain a total occurrence vector T. More precisely, for each element in T, the value represents the number of articles where the word appears. Among the values in T, words having an occurrence value of 10% or more of the maximum value in T are defined as frequent words F and we obtain a vector V_{freq} of length n that has value 1 on the frequent words, and 0 otherwise:

$$V_{freq}[i] = \begin{cases} 1, & w_i \in F \\ 0, & w_i \notin F \end{cases}.$$

After we determine the vector V_{freq}, we compute the Euclidean distance between V_{freq} and V_j and define the distance as d_j. For the m distances, we cluster them with the Dirichlet Process Gaussian Mixture Model introduced in (Rasmussen 2000). In detail, a maximum number of clusters is set to five. Note that some of the clusters may not contain enough data and the clusters are not appropriate to represent the writing patterns. In this case, we choose the clusters such that sizes of the clusters are at least $\frac{m}{5}$, where the denominator comes from the maximum number of clusters.

Using CDFA, we obtain the clusters of distances. Among them, we choose a cluster that has the minimum average of the distances. Because posting bots tend to write articles with a fixed form, words in the form would be in the frequent words, and the distances between the frequent words and articles with fixed forms would be small. Therefore, we select the cluster with the minimum average of the distance and denote the cluster as MAC-CDFA. From the MAC-CDFA, we extract the mean, variance of the MAC-CDFA, and the number of clusters that have the size at least $\frac{m}{5}$.

Figure 1 shows the procedure of CDFA in brief. From the articles, we extract frequent words and calculate the distance between the frequent words and articles. After that, we cluster the distances and choose the cluster that has the least mean among the clusters.

CDFA can be applied to various datasets. For a blog, there are the title, content, and replies. We applied the CDFA to the title, content, and replies that are written by an account, and we denote them as CDFA-T, CDFA-C, and CDFA-R, respectively. Similarly, we define the MAC-CDFA-T, MAC-CDFA-C, and MAC-CDFA-R. For each MAC-CDFA, we extract three features, thus there are nine features from the CDFA. We call the nine features CDFA feature group (or simply CDFA features). For accounts with less than five blogs, the value of mean and variance of MAC-CDFA-T and MAC-CDFA-C are 0. In addition, the number of clusters via CDFA-T and CDFA-C are 0. Similarly, for accounts with less than five replies, the mean and variance of MAC-CDFA-R and the number of clusters via CDFA-R are 0.

Figure 2 shows distributions of features in the CDFA group. The top three graphs represent the mean of MAC-CDFA, three graphs in the middle represent the variance of MAC-CDFA, and the bottom three graphs represent the number of clusters via CDFA. To observe meaningful data, we make histograms using accounts with five or more blogs for the left six graphs, and accounts with five or more replies for the right three graphs because the excluded accounts have the value of 0.

In addition to our features, there are numerous other ones related to text similarity and natural language processing.

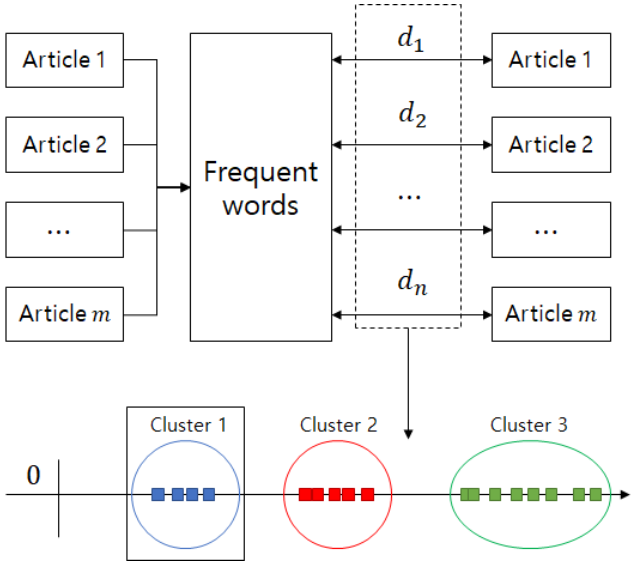

Figure 1: CDFA and MA-CDFA

Index	Feature Name
1	Average of MAC-CDFA-T
2	Variance of MAC-CDFA-T
3	Number of clusters in CDFA-T
4	Average of MAC-CDFA-C
5	Variance of MAC-CDFA-C
6	Number of clusters in CDFA-C
7	Average of MAC-CDFA-R
8	Variance of MAC-CDFA-R
9	Number of clusters in CDFA-R

Table 1: Features in CDFA Group

In this section, we will compare the CDFA features with other text-related features: (i) frequent word counts, (ii) term frequency–inverse document frequency (TF–IDF), (iii) Levenshtein edit distance, and (iv) word embedding.

We obtained the frequent word count in the CDFA process. Concurrently, one of the standard methods to analyze text content is TF–IDF, which assigns a weight to each word in a document. In general, in TF–IDF, common words are allotted low weights, whereas uncommon ones have high weights. However, TF–IDF does not represent the features of an account. Assuming that an account frequently uses a word, whereas other accounts scarcely employ it, then the TF–IDF weight of this word will be higher than those of the other words employed in the content written by that account. However, the TF–IDF weight of a word will be relatively low if the word was found in other documents. We calculated the TF–IDF weights of those words occurred for ten times or more.

The Levenshtein edit distance is one of the standard approaches to calculate the distance between two texts. Owing to the long computing time of the model, we sampled the accounts that wrote less than 500 posts and less than 500 replies. Moreover, for a given account, we divided data into titles of blogs, content of blogs, and replies, and calculated the mean of the pairwise distance of each categorized data.

Features	Score	Score of CDFA
Frequent word counting	62.78	83.72
TF-IDF	79.38	83.72
Levenshtein edit distance	73.01	83.22
Word embedding	59.05	78.14

Table 2: Comparison between text related features and CDFA

Word embedding is used in natural language processing. To conduct the word embedding, we collected pre-trained dataset in English, German, Spanish, Korean, French, and Russian from Fasttext(Joulin et al. 2016). In the pre-trained dataset, a word is assigned to a 300-dimensional vector. We denote the set of words in the pre-trained dataset as a *bag of words*. For a given text, we split it into words, and calculate the average of vectors corresponding with the words which are in the bag of words. We denote the average of vectors as a *text vector*. For each account, we calculate a text vector for each blog content or replies, obtain an *account vector* as the average of the text vectors, and use the account vector as a feature. We sampled accounts that the number of words contained in both the bag of words and the words that the account used is 500 or more.

In Table 2, we compare all the text-related features to the CDFA features. We used Random Forest classifier with Gini index in the classification. The left scores in the table are the F_1-$scores$ of the features, and the right scores are the corresponding F_1-$scores$ of the CDFA features. Because an account set varies, the corresponding scores of the CDFA features also vary. We observe that the CDFA features outperform the other features.

Santia-2019 Group

In the Santia-2019 group, there are six features; average response time, average comment length, innovation rate, maximum daily comments, number of links and thread deviation.

Average Response Time Steemit users can leave comments on a blog or may leave replies to the comments left on the blog. Moreover, users can leave replies to the replies. We introduce a depth of comments to explain this process. Blogs in Steemit are comments of depth 0. Comments left on blogs are comments of depth 1. If you leave a reply on a comment of depth n, then your reply is of depth $n+1$. Then, the comment of depth n you left a reply to is the parent reply to your reply. Response time measures how long each reply has been created since a previous reply was made. Here, the previous reply means a reply written just before the reply among replies whose parent reply is the same. If a reply is the first among them, we compute the time difference from the parent reply. Then, we obtain the response time of each reply. Given a user, average response time is the average of the response times for all replies written by the user.

Average Comment Length We generate features related to blogs and replies. One of them is the average comment

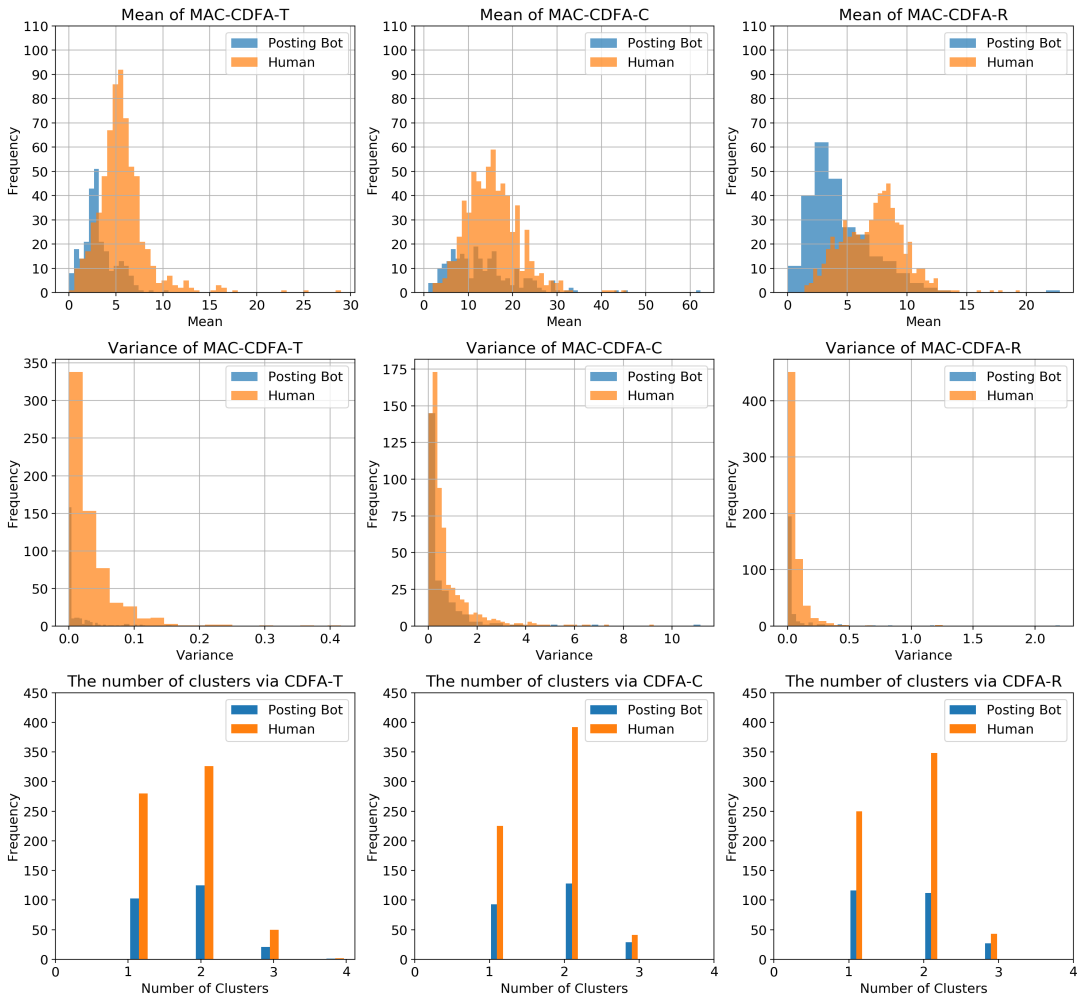

Figure 2: Distribution of Features in CDFA group

length. Same as (Santia, Mujib, and Williams 2019), some of the posting bots generate blog content or replies that are long. We generate the average comment length by averaging lengths of all the blog content and replies written by an account.

Innovation Rate One of the criteria for identifying humans and bots is the diversity of words. To measure the diversity of words, (Santia, Mujib, and Williams 2019) used the innovation rate that represents the decay rate of diversity of words. In (Clark et al. 2016), to detect the automation on Twitter, they used the word introduction decay rate $\alpha(n)$. In our case, the whole procedure is the same except the shuffling. Because there are many blogs and replies for some bots, we shuffled the words based on the articles. That means, for m articles, we shuffle the order of the articles, split them with space, and make the sequence of the words. Further, we shuffle three times to obtain the innovation rate.

Maximum Daily Comments Unlike ordinary accounts, bots can write many articles in a day using automated programs. To deal with the bots that generate a massive number of blogs or replies in a short period, we extract the maximum daily comments.

Number of Links We use a regular expression to extract the strings that http or https contain. Even though a regular expression is used, some strings could not be included in the URL in the middle, thus a URL validator is used to filter them out.

Thread Deviation This feature represents a regularity in a user's response patterns. We compute response times of all replies left on a blog. Next, we check the average response time corresponding to the blog. Then, for each reply left on the blog, we calculate a difference between the response time of the reply and the average response time of the blog. This difference is called deviation. We calculate the deviation of replies written by a user. Finally, a thread deviation of a user is defined as the average of deviation of replies written by the user.

Index	Feature Name
10	Average Response Time
11	Average Comment length
12	Innovation Rate
13	Maximum Daily Comments
14	Number of Links
15	Thread deviation

Table 3: Features in Santia-2019 Group

Index	Feature Name
16	Entropy rate
17	Hashtag ratio
18	Mention ratio
19	URL ratio
20	FF ratio
21	The age of an account

Table 4: Features in Chu-2012 Group

Chu-2012 Group

In Chu-2012 group, there are six features; entropy rate, hashtag ratio, mention ratio, URL ratio, FF ratio and the age of an account.

Entropy rate The entropy rate $\bar{H}(X)$ is the conditional entropy of an infinite random process $X = \{X_i\}$,

$$\bar{H}(X) = \lim_{n\to\infty} H(X_n|X_{n-1}, \cdots, X_1),$$

where the conditional entropy is computed as follows:

$$H(X_n|X_{n-1}, \cdots, X_1) = H(X_1, X_2, \cdots, X_n) - H(X_1, X_2, \cdots, X_{n-1}),$$

and an entropy of a sequence of random variables is defined as

$$H(X_1, \cdots, X_n) = -\sum_{i=1}^{n} P(X_i = x_i) \log P(X_i = x_i).$$

Here, we denote the above equation as an *entropy formula*. Because real data sets are finite, (Chu et al. 2012) used a corrected conditional entropy, denoted as CCE, to estimate the entropy rate. First, they derived the joint probabilities, $P(X_1 = x_1, \cdots, X_n = x_n)$, empirically. Then, they computed the conditional entropy based on the empirically derived joint probability. This conditional entropy is denoted by CE. Then, they added corrective terms $per(X_n) \cdot EN(X_1)$, where $per(C_n)$ is the percentage of unique sequences of length n, and $EN(X_1)$ is the entropy of X_1 as follows:

$$CCE(X_n|X_{n-1}, \cdots, X_1) = CE(X_n|X_{n-1}, \cdots, X_1) + per(X_n) \cdot EN(X_1).$$

They determined n that minimizes CCE, and also computed the entropy rate of the sequence of tweeting intervals of each user. We measured the entropy rate of the sequence of comment time intervals and the time difference between comment actions.

Account Properties As we mentioned at the beginning of the feature generation section, some features are available. In the case of blogs, tag data contains the tags of blogs that represent the main topic of the blogs. Thus, we use a regular expression to extract the hashtags. After obtaining the hashtags, we calculate the *hashtag ratio* by dividing the number of blogs and replies that contain mentions to the number of blogs and replies. We also extract the *mention ratio* in a similar way to the hashtag ratio. In the case of the *URL ratio*, we calculate it via processed data used to extract the number of links by using a similar approach to the hashtag ratio.

Next, using the follower and following data, we calculate the *FF ratio*. We obtain the FF ratio by dividing the number of followers by the sum of the number of followers and followings. If the number of followers and followings are 0, the FF ratio is 0. Finally, the *age of an account* is the difference between the time the account was created and the time at the end of the dataset.

Blockchain-Oriented Feature Group

Based on the blockchain system, we added 12 features, which are listed in Table 5, and term them blockchain-oriented (or simply blockchain) features. In this section, we introduce the blockchain features. *Number of transfers* is the sum of the transfers; *Daily time entropy of transfer* is the entropy setting transfers per day as a random variable in the entropy formula; *Transfer activation time* is the interval between the first and last transfer times; *Daily transfer* is obtained by dividing the number of transfers by the transfer activation time; *In-degree of transfer* of an account is the number of accounts that transferred to the account; *Out-degree of transfer* of an account is the number of accounts that the account transferred; *Entropy of the in-degree accounts* of an account is the entropy setting accounts that are transferred to the account as a random variable in the entropy formula; *Entropy of the out-degree accounts* of an account is the entropy setting accounts that the account transferred as a random variable in the entropy formula, and *Steem-created account* determines whether the account is created by Steem. Initially, to obtain *average transfer per blog or reply*, we calculate the number of blogs or replies and the number of transfers on each day. Subsequently, we divide the number of blogs or replies into the number of transfers on each day and obtain a feature by taking an average. *Average transfer per blog* and *Average transfer per reply* are obtained similarly.

Posting Bot Classification

We explain how to classify the posting bots. First, we introduce a dataset. Second, we clarify an annotation process. Finally, we describe the procedure of classification using several classifiers.

Dataset

Steemit is a social media platform based on the Steem blockchain. The Steem blockchain is a public blockchain;

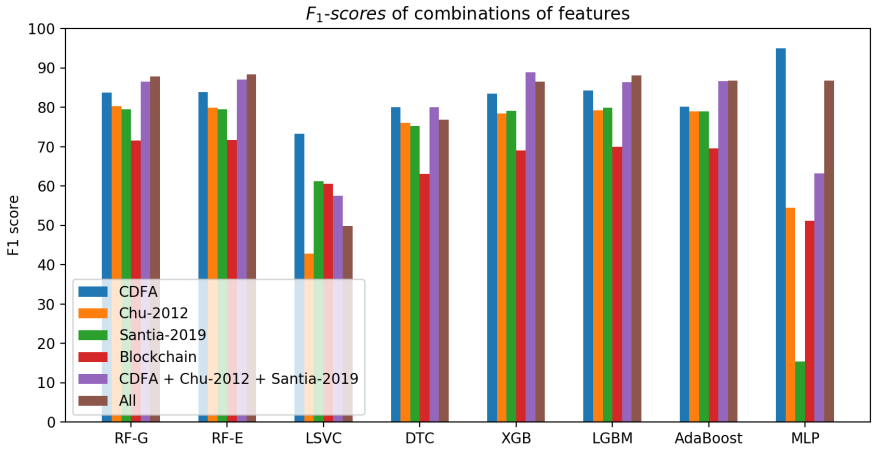

Figure 3: F_1-$score$ comparison

Index	Feature Name
22	Number of transfers
23	Daily time entropy of transfer
24	Transfer activation time
25	Daily transfer
26	In-degree of transfer
27	Out-degree of transfer
28	Entropy of the in-degree accounts
29	Entropy of the out-degree accounts
30	Steem-created account
31	Average transfer per blog or reply
32	Average transfer per blog
33	Average transfer per reply

Table 5: Features in the Blockchain-oriented Feature Group

therefore, all the data are publicly available.[5] Using the data from February 2019 to December 2019, we manually classified humans and bots. A total number of 984 accounts were divided into 325 bot accounts and 659 human accounts. For sampling, we collected the users that write blogs or replies that are 40 times or more. We describe the detailed labeling process in the annotation section.

Annotation

Because Steem is less explored in previous studies, annotation is one of the challenging tasks in our research. Two annotators participated in the annotation, which consists of two stages. In the first stage, both the annotators label the accounts independently using the same dataset. Table 6 summarizes the results of the first stage. Subsequently, in the second stage, the annotators compare and discuss their labels. Remark that Cohen's Kappa value is 90.23. Some accounts write several posts or replies like humans, but they

[5]However, there are some types of data that are not stored on this blockchain. First, any media (e.g., pictures, videos, etc.) is stored in a typical centralized cloud service. Second, broadly, certain activities (e.g., login, logout, and read) that are not publicly available on steemit.com are not stored on the blockchain.

Annotators		Annotator 1	
		Bot	Human
Annotator 2	Bot	283	26
	Human	15	660

Table 6: Annotation in the First Stage. The Cohen's Kappa value is 90.23.

are suspicious of using the automated program in some posts or replies. We denote them as *semi-automated accounts*. The annotators analyzed that the Cohen's Kappa value is high because the subjective opinions of two annotators coincide in labeling semi-automated accounts as bots. The annotators agreed to establish the criteria for labeling to deal with the semi-automated accounts and consider the disagreement of 4.17% for all the accounts.

Basic criteria for labeling bots were established to be general characteristic of a blog and reply data of the accounts, which are labeled as bots from both the annotators. To set the basic criteria, we define the *form*. When multiple texts have the same form, only the numbers, accounts, and links in the texts change, and the rest is the same. For example, some accounts have the form of "You got a [n]% upvote from [account]." In this case, the changes are only in the number n and/or the account part. In addition, some accounts have the form of a table with the ranking of accounts according to some criteria. In this scenario, changes occur only in the accounts in the table. Second, if an account has several forms and writes repeatedly using them, the account is labeled as a bot. For example, when an account runs a gambling app, it is necessary to set the forms such as the winner, amount won, amount they can bet on, and remained funds for gambling. In summary, our basic criterion is that a bot has a certain form or forms in blogs or replies and writes ten or more times in a row using the form or forms. If the blog content or replies of an account match the basic criterion, the account is labeled as a bot.

However, there are accounts that our basic criterion may not be adequately applied. Therefore, we establish some ex-

ceptions. An account that satisfies one of the following cases is labeled as a human: (i) leaving replies to participate in an event or use a service that a bot cannot participate or use easily, (ii) writing a link related to a game-play live streaming and having ten or more replies which do not satisfy the basic criteria, (iii) reporting one's workout records using workout app that has its own abusing detection system, (iv) posting a personal game app status and having ten or more replies which do not satisfy the basic criteria, and (v) posting pictures and having ten or more replies which do not satisfy the basic criteria. In contrast, an account that satisfies one of the following cases is labeled as a bot: (i) copying news and having less than ten replies; (ii) randomly rearranging short sentences.

The created CDFA features focus on text similarity. In the labeling process, the basic criteria are related to text similarity. However, our labeling does not entirely depend on text similarity. It also considers the opinions or experiences of users. In addition, the labeling process considers copying the news or other types of bots that do not depend on text similarity.

Classification Procedure

For the classification, we used several classifiers. In (Santia, Mujib, and Williams 2019) and (Chu et al. 2012), Random Forest classifiers with Gini index and Entropy (Breiman 2001), Linear Support Vector classifier (Cortes and Vapnik 1995), and Decision Tree classifier (Breiman 2017) are used to detect bots. In addition to them, we also used more classifiers based on boosting algorithms such as XGBoost (Chen and Guestrin 2016), LightGBM (Ke et al. 2017), and AdaBoost (Freund, Schapire, and Abe 1999). Also, we applied the multi-layer perceptron (MLP) classifier (Windeatt 2006) as a representative neural network. We denote Random Forest classifier with Entropy as RF-E, Random Forest classifier with Gini index as RF-G, Linear Support Vector classifier as LSVC, XGBoost classifier as XGB, Decision Tree classifier as DTC, LightGBM as LGBM.

In case of scores, we used the four traditional measurements; $Accuracy$, $Precision$, $Recall$, and $F_1\text{-}score$.

To generate the results, we performed five-fold cross-validation. First, we shuffled our dataset and divided it equally into five sets. Next, we choose the first set as the test set, with the remainder becoming the training set. In the training set, we optimized the hyperparameters for each classification algorithm using a grid search via the five-fold cross-validation to obtain a high $F_1\text{-}score$. Applying the optimized hyperparameters to the classification algorithms, we obtained the models and fit them to the test set and acquired the results. From the divided sets, we can choose five different test sets. Repeating the above procedure, we realized five different results for each model and obtained the final result for each model by taking the average.

Results and Discussion

We compare the results obtained by employing the four feature groups, and observe that the CDFA group outperforms the other ones. In addition, we derive the results corresponding to the presence and absence of the blockchain features, to ensure that blockchain-oriented features are effective. To interpret the results, we consider the feature importance of each model and rank the features accordingly. For highly ranked features, we further analyze their characteristics.

Results

Note that we categorized the features into the four feature groups: CDFA, Santia-2019, Chu-2012 and blockchain features. For convenience, Santia-2019 is denoted as S, and Chu-2012 as C. Table 7 shows the results when only one of the four feature groups is applied and the results when blockchain-oriented features are excluded and included. In Table 7, we highlight the best scores among the four feature groups for each classifier and the best scores among the classifiers in the cases of including and excluding blockchain-oriented features respectively. We observe that the CDFA group classifies posting bots better than the other feature groups. In addition, including blockchain-oriented features is more effective in detecting posting bots except for linear support vector classifier, decision tree classifier and XGBoost classifier. Finally, as we see in Table 7, Random Forest classifier with entropy gives the best score in $Accuracy$ and $F_1\text{-}score$, Random Forest classifier with Gini index gives the best score in $Precision$, and AdaBoost classifier gives the best score in $Recall$.

Feature Importance

The tree-based ensemble models (Random Forest, Decision Tree, XGBoost, LightGBM, AdaBoost) provide the feature importance. For a tree-based model, the classification is done based on the features in the data set. Feature importance provides information on how the features contributed to improve scores. In the classification procedure, we obtain six different tables of feature importance from the six classifiers. Because we calculate the $F_1\text{-}score$ by averaging the $F_1\text{-}scores$ of the five different test sets for each model, we also calculate the feature importance by averaging the importance of the five different train sets.

Figure 4 shows the results of the top 15 feature importance of each model. The x axis represents the feature names. We observe that some of features in the CDFA group have high importance. As we see in Figure 4, only the qualitative analysis on the feature rank is available due to the scattered graphs. To determine the overall ranking, we consider the Borda count in (Borda 1784) that is one of the popular election methods. The Borda count changes the rank of the relative points and determines the final rank by summing the relative points. There are many methods that change the rank of the relative points. In this study, we use the Dowdall system that calculates the relative points with the reciprocal of the rank. For the average of the feature importance of each model, we determine the rank with respect to the importance. Furthermore, we get the reciprocal of the ranks and sum them up. Finally, we obtain the sum of the relative points of the features and determined the rank. Table 8 shows the top five features that have relative points higher than 1. We observe that three of them are in the CDFA group. The FF ratio and the innovation rate are also in Table 8. We will analyze the features in Table 8 further in the next section.

Models	Scores	CDFA	Santia-2019	Chu-2012	Blockchain	CDFA + S + C	All
RF-G	Accuracy	**89.63**	87.40	87.10	82.42	91.46	92.28
	Precision	**81.73**	78.66	76.23	66.93	83.48	**85.12**
	Recall	**85.94**	82.25	83.35	77.32	89.81	90.81
	F_1	**83.72**	80.31	79.53	71.52	86.44	87.83
RF-E	Accuracy	**89.64**	87.20	87.20	82.42	91.77	**92.68**
	Precision	**82.10**	77.74	75.58	67.51	84.50	84.77
	Recall	**85.75**	82.43	84.06	76.85	89.81	92.28
	F_1	**83.83**	79.83	79.51	71.70	87.03	**88.32**
LinearSVC	Accuracy	**83.64**	57.83	75.31	74.90	64.94	63.83
	Precision	**67.70**	50.45	63.43	58.81	71.08	54.18
	Recall	**80.56**	49.62	65.99	64.29	56.23	56.98
	F_1	**73.22**	42.76	61.21	60.59	57.45	49.77
DTC	Accuracy	**87.50**	85.67	84.96	76.83	87.40	85.67
	Precision	**76.30**	69.73	70.19	60.27	76.54	72.34
	Recall	**84.33**	84.01	81.29	67.76	83.78	82.59
	F_1	**79.99**	76.00	75.29	63.04	79.97	76.82
XGBoost	Accuracy	**89.64**	86.38	87.09	81.40	**92.99**	91.46
	Precision	**79.39**	75.03	74.05	62.97	**85.73**	83.32
	Recall	**88.02**	82.57	84.92	76.92	**92.36**	90.02
	F_1	**83.47**	78.36	79.06	69.06	**88.88**	86.48
LightGBM	Accuracy	**89.94**	87.09	87.50	81.91	91.36	92.48
	Precision	**81.60**	74.98	75.34	63.89	83.23	84.49
	Recall	**87.13**	83.99	85.09	77.58	89.83	91.92
	F_1	**84.23**	79.19	79.88	69.97	86.33	88.02
AdaBoost	Accuracy	**87.91**	87.10	87.09	81.30	91.36	91.77
	Precision	**74.12**	73.48	73.71	64.79	84.61	81.79
	Recall	**87.33**	85.33	85.17	75.64	88.91	**92.50**
	F_1	**80.09**	78.89	78.94	69.58	86.61	86.70
MLP	Accuracy	**85.77**	70.93	60.98	70.94	76.93	75.81
	Precision	**72.60**	53.54	22.76	50.54	62.77	63.94
	Recall	**82.54**	60.65	37.01	63.84	70.09	65.19
	F_1	**94.97**	54.50	15.40	51.19	63.21	63.74

Table 7: Scores of the Models

Rank	Feature Name
1	Variance of MAC-CDFA-R
2	Variance of MAC-CDFA-T
3	Mean of MAC-CDFA-T
4	Innovation Rate
5	FF Ratio

Table 8: Rank of Features (Top 5)

Feature Interpretation

In this section, we check the distributions of important features showed in Table 8, and a feature that has the highest rank among the blockchain features. Top left graph in the Figure 2 shows the distribution of the mean of MAC-CDFA-T for active users who post blogs five times or more. The users in the graph are normally distributed, and the distribution of posting bots has smaller means in the graph. This shows that posting bots tend to post with some forms in titles.

The top two histograms in Figure 5 show the log scaled distributions of variance of MAC-CDFA-R and MAC-CDFA-T for active users. We see that the variances of MAC-CDFA-R and MAC-CDFA-T of posting bots are zero more often than humans. In contrast, the log scaled distributions for humans resemble the normal distributions. Consequently, we infer that an active user is a posting bot when the variance of MAC-CDFA-T or MAC-CDFA-R is zero.

(Chu et al. 2012) analyzed that automated bots on Twitter follow numerous users, expecting that humans will follow them in return. However, this scenario is reversed on Steemit. The lower left graph in Figure 5 presents the distribution of the FF ratio. We observe that FF ratios of various posting bots are close to 1 each. This suggests that most of the posting bots do not follow other users. In contrast, humans follow other users actively.

A user who has a limited vocabulary has a high innovation rate, whereas a creative user has a low innovation rate. This is well-illustrated in the lower right in Figure 5. This distribution displays that users with high innovation rates are posting bots, whereas those with low innovation rates are humans.

In contrast, the feature with the highest-ranking among the blockchain features is the out-degree of transfer, and it

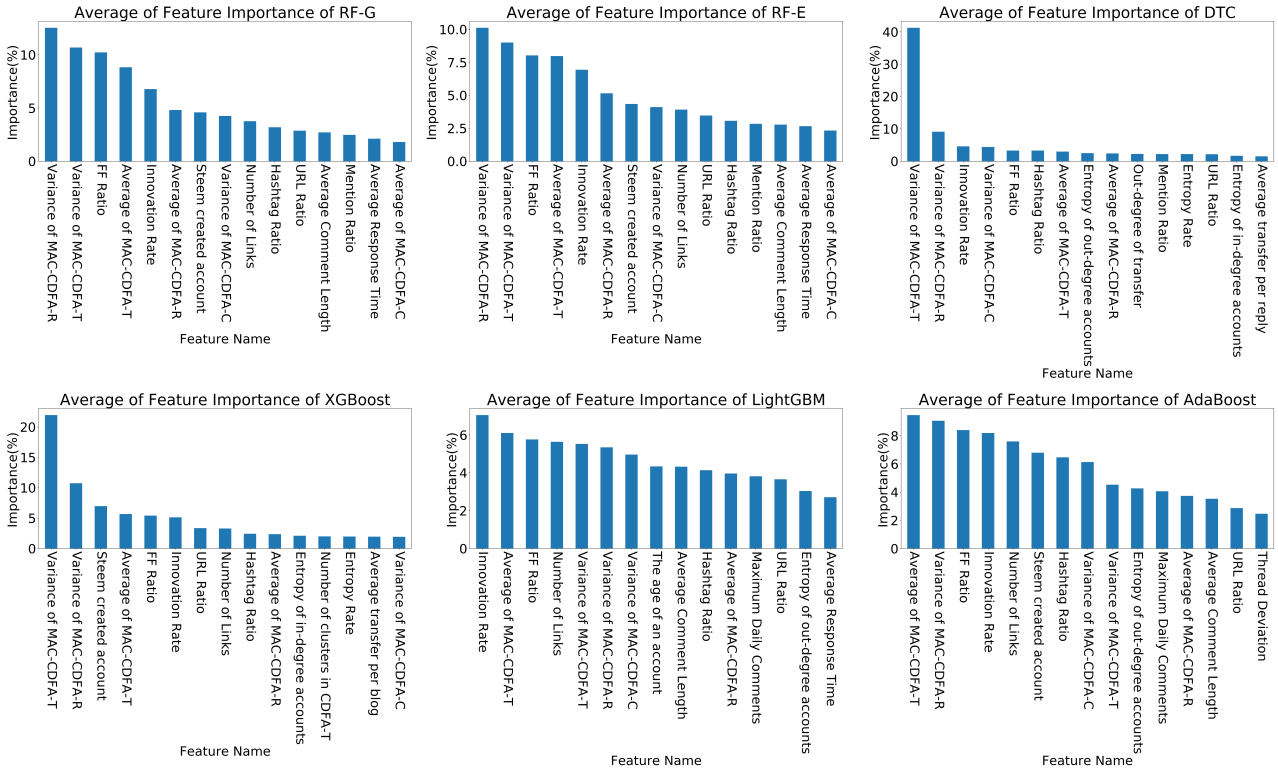

Figure 4: Top 15 feature importance of classifiers

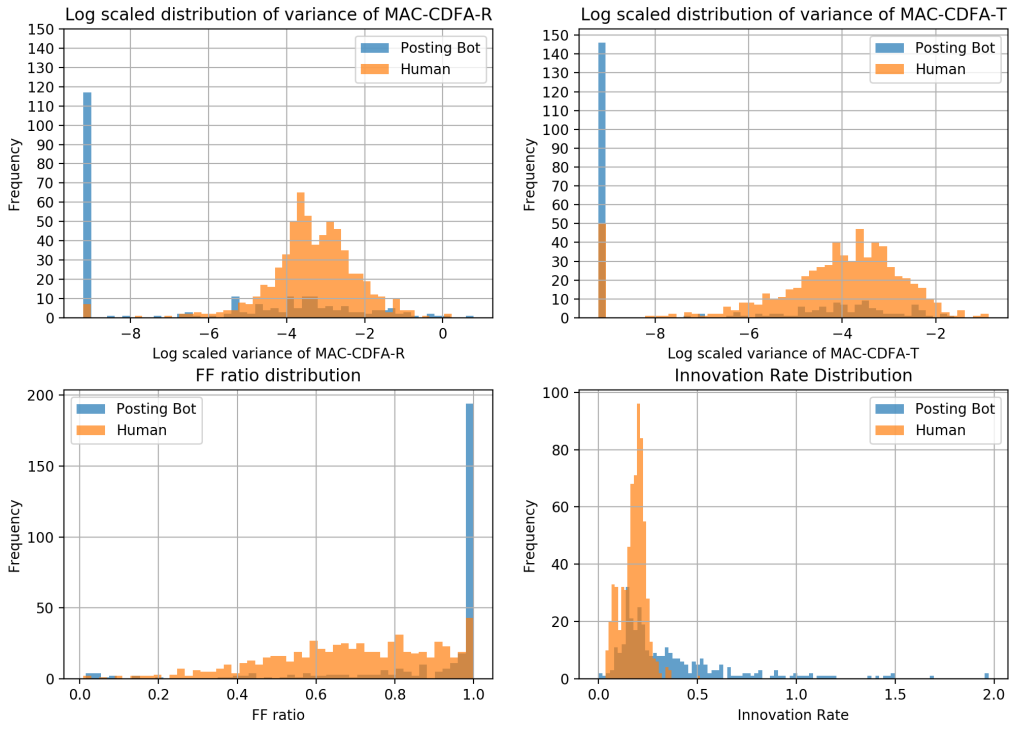

Figure 5: Distributions of important features

is ranked 12th. Analysis of this feature demonstrated that 39 accounts had an out-degree of transfer of more than 200, of which 92.3% or 36 accounts were bots, and 7.7% or three accounts were humans. In addition, 11.1% of the bots and only 0.5 % of the humans had an out-degree of transfer of more than 200. Observation of these accounts suggests that they need to transfer tokens to other accounts, such as running games and events, or manage their tokens within the Steem blockchain. Therefore, from the out-degree of transfer feature, we infer that it assists in detecting these types of bots.

Overall, we observed that the behaviors of posting bots are different from those of humans in numerous aspects. Based on the CDFA features, we obtain the information of the texts close to the representative text structure of each account. In fact, our results demonstrate that using a representative text structure is essential to detect posting bots. Considering the innovation rate, bots produce the same texts with little variations and have restricted vocabularies. In addition, we find an extreme distribution of the FF ratio. From the distribution, we infer that developing a relationship with other accounts is not a primary objective of posting bots. Among the blockchain features, the out-degree of transfer is essential in classification, and we detect some bots that transfer a large amount of cryptocurrencies for running games or managing their tokens.

Conclusion

The problem of detecting posting bots is one of the essential issues to avail more rewards to human users and motivate them to generate good content. In this paper, we developed features in a CDFA group to detect the posting bots. The CDFA method is used to find frequent words in articles and to measure the distance between the frequent words and the articles. Note that Steemit users can write blogs or replies without limit of length of words like on Facebook. To analyze the posting bot, it is necessary to deal with a large number of blogs or replies with unlimited length because they can generate many articles in a short period of time. Therefore, we calculate the similarity of articles by transforming the articles into real numbers and using a clustering method that can deal with many blogs and replies. With CDFA, we select the MAC-CDFA among the clusters obtained from CDFA and extract features from MAC-CDFA. To compare the performance of features, we benchmark the features introduced in (Santia, Mujib, and Williams 2019) and (Chu et al. 2012), and use the F_1-$score$ as a comparison measure. The results show that the features in the CDFA group are more effective than other feature groups. To interpret the results, we calculated the feature importance and its rank and performed further analysis of feature distribution.

There is a limitation of our research. In our labeling process, annotations were rarely proceeded for languages that the annotators were not familiar with. In the future research, we expect that new features are developed and detect other kinds of bots in blockchain-based social media platforms. For example, bid voting bots receive cryptocurrency and upvote posts or replies. Although a list of such bots is available, detecting such bots systematically and using them to improve posting bot detection quality would be of interest. Also, we will be able to improve the results by developing customized CDFA features for each language. Finally, we expect that the CDFA features will be used to detect posting bots on social media platforms other than Steemit.

Acknowledgements

H.J. Hwang was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2017R1E1A1A03070105) and by Institute for the Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)) and by the ITRC(Information Technology Research Center) support program(IITP-2018-0-01441).

References

Abu-El-Rub, N.; and Mueen, A. 2019. BotCamp: Bot-driven Interactions in Social Campaigns. In *The World Wide Web Conference*, 2529–2535. ACM.

Allcott, H.; and Gentzkow, M. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives* 31(2): 211–36.

Borda, J. d. 1784. Mémoire sur les élections au scrutin. *Histoire de l'Academie Royale des Sciences pour 1781 (Paris, 1784)* .

Boshmaf, Y.; Logothetis, D.; Siganos, G.; Lería, J.; Lorenzo, J.; Ripeanu, M.; and Beznosov, K. 2015. Integro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs. In *NDSS*, volume 15, 8–11.

Breiman, L. 2001. Random forests. *Machine learning* 45(1): 5–32.

Breiman, L. 2017. *Classification and regression trees*. Routledge.

Cao, Q.; Sirivianos, M.; Yang, X.; and Pregueiro, T. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 15–15. USENIX Association.

Casadesus-Masanell, R.; White, A.; and Elterman, K. 2019. Steemit: A New Social Media? Harvard Business School Case 720-428.

Castellini, J.; Poggioni, V.; and Sorbi, G. 2017. Fake twitter followers detection by denoising autoencoder. In *Proceedings of the International Conference on Web Intelligence*, 195–202. ACM.

Chavoshi, N.; Hamooni, H.; and Mueen, A. 2016. DeBot: Twitter Bot Detection via Warped Correlation. In *ICDM*, 817–822.

Chavoshi, N.; Hamooni, H.; and Mueen, A. 2017. Temporal patterns in bot activities. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 1601–1606. International World Wide Web Conferences Steering Committee.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794. ACM.

Chu, Z.; Gianvecchio, S.; Wang, H.; and Jajodia, S. 2012. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6): 811–824.

Clark, E. M.; Williams, J. R.; Jones, C. A.; Galbraith, R. A.; Danforth, C. M.; and Dodds, P. S. 2016. Sifting robotic from organic text: a natural language approach for detecting automation on Twitter. *Journal of Computational Science* 16: 1–7.

Cortes, C.; and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3): 273–297.

Cresci, S.; Di Pietro, R.; Petrocchi, M.; Spognardi, A.; and Tesconi, M. 2017. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing* 15(4): 561–576.

Dickerson, J. P.; Kagan, V.; and Subrahmanian, V. 2014. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 620–627. IEEE Press.

El-Mawass, N.; Honeine, P.; and Vercouter, L. 2018. Supervised classification of social spammers using a similarity-based markov random field approach. In *Proceedings of the 5th Multidisciplinary International Social Networks Conference*, 14. ACM.

Feng, B.; Li, Q.; Pan, X.; Zhang, J.; and Guo, D. 2017. GroupFound: An effective approach to detect suspicious accounts in online social networks. *International Journal of Distributed Sensor Networks* 13(7): 1550147717722499.

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

Freund, Y.; Schapire, R.; and Abe, N. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14(771-780): 1612.

Höner, J.; Nakajima, S.; Bauer, A.; Müller, K.-R.; and Görnitz, N. 2017. Minimizing trust leaks for robust sybil detection. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1520–1528. JMLR. org.

Jeong, S. E. 2020. Centralized Decentralization: Does Voting Matter? Simple Economics of Governance Attacks on the DPoS Blockchain. https://ssrn.com/abstract=3575654. Accessed: 2020-05-05.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* .

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154.

Lee, S.; and Kim, J. 2014. Early filtering of ephemeral malicious accounts on Twitter. *Computer Communications* 54: 48–57.

Li, C.; and Palanisamy, B. 2019. Incentivized Blockchain-based Social Media Platforms: A Case Study of Steemit. In *Proceedings of the 10th ACM Conference on Web Science*, 145–154.

Minnich, A.; Chavoshi, N.; Koutra, D.; and Mueen, A. 2017. BotWalk: Efficient adaptive exploration of Twitter bot networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 467–474. ACM.

Rasmussen, C. E. 2000. The infinite Gaussian mixture model. In *Advances in neural information processing systems*, 554–560.

Santia, G. C.; Mujib, M. I.; and Williams, J. R. 2019. Detecting Social Bots on Facebook in an Information Veracity Context. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, 463–472.

Steemit. 2017. Steem: An incentivized, blockchain-based, public content platform. https://steem.com/SteemWhitePaper.pdf. Accessed: 2019-08-03.

Thelwall, M. 2018. Can social news websites pay for content and curation? The SteemIt cryptocurrency model. *Journal of Information Science* 44(6): 736–751.

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

Wang, A. H. 2010. Detecting spam bots in online social networking sites: a machine learning approach. In *IFIP Annual Conference on Data and Applications Security and Privacy*, 335–342. Springer.

Wang, B.; Zhang, L.; and Gong, N. Z. 2017. SybilSCAR: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, 1–9. IEEE.

Windeatt, T. 2006. Accuracy/diversity and ensemble MLP classifier design. *IEEE Transactions on Neural Networks* 17(5): 1194–1211.