IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 18, 2019, accepted March 4, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904220

Contrast Pattern-Based Classification for Bot Detection on Twitter

OCTAVIO LOYOLA-GONZÁLEZ[1], RAÚL MONROY[2], JORGE RODRÍGUEZ[2], ARMANDO LÓPEZ-CUEVAS[3], AND JAVIER ISRAEL MATA-SÁNCHEZ[2]

[1]Tecnologico de Monterrey, School of Engineering and Science, Puebla 72453, Mexico
[2]Tecnologico de Monterrey, School of Engineering and Science, Estado de Mexico 52926, Mexico
[3]Tecnologico de Monterrey, School of Engineering and Science, Jalisco 45201, Mexico

Corresponding author: Octavio Loyola-González (octavioloyola@tec.mx)

This work was supported by the Google, Inc., under the APRU Project ''AI for Everyone''.

ABSTRACT Detecting non-human activity in social networks has become an area of great interest for both industry and academia. In this context, obtaining a high detection accuracy is not the only desired quality; experts in the application domain would also like having an understandable model, with which one may explain a decision. An explanatory decision model may help experts to consider, for example, taking legal action against an account that has displayed offensive behavior, or forewarning an account holder about suspicious activity. In this paper, we shall use a pattern-based classification mechanism to social bot detection, specifically for Twitter. Furthermore, we shall introduce a new feature model for social bot detection, which extends (part of) an existing model with features out of Twitter account usage and tweet content sentiment analysis. From our experimental results, we shall see that our mechanism outperforms other, state-of-the-art classifiers, not based on patterns; and that our feature model yields better classification results than others reported on in the literature.

INDEX TERMS Contrast patterns, bot detection, supervised classification, social networks.

I. INTRODUCTION

Nowadays, a large number of social networks, such as Twitter[1] and Facebook,[2] are provided to the public, free of charge. People are very much engaged in associating with a social network, and commonly follow others, they consider paramount, including singers, athletes and politicians.

A *bot* is a software, specially crafted to execute a task regularly [1]–[3]. In social media, some bots are used to carry out simple actions, such as ''re-post'' somebody else's posts, but others may do more complex tasks, like posting brand new messages or simulating human participation. Usually, bots are programmed to interact with one another, forming a *botnet* [1], [4] with the aim of achieving an even more complex goal.

Botnets have been recently developed to provoke a trending topic. They inject messages to social media so as to favor a political figure or as to misrepresent that figure's counterparts [5]. Botnets are considered to be highly influential in people's opinion, as has been reported in cases such as Brexit [6] and the federal elections of the United States in 2016 [7]. In addition, some botnets have been programmed to attract people with the objective of making them adopt extremists' ideologies of terrorism [5]–[9]. There are currently about 48 million bot accounts in Twitter, amounting approximately to 15% of all active client accounts [8]. It is therefore crucial to develop tools that help detect bots in social networks.

Current mechanisms for bot detection [10]–[14] present severe limitations, being the absence of an understandable model of paramount importance [15]. Understand-ability is a desirable property for a classifier [16]–[18]. Indeed, for some domains this property is even mandatory. For example, the Equal Credit Opportunity Act of the United States renders illegal to deny a credit to an individual for vague or undefined reasons. Hence, it is now recommended for a financial institution to use a classification mechanism that produces an explanatory model [17]. Bot detection in social media is yet another domain that demands an understandable classification mechanism, for example, to avoid blocking an account that belongs to a genuine user. Hence, in this paper, we will

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Gyu Kim.

[1]http://twitter.com
[2]http://facebook.com/

focus on developing an understandable classification model for bot detection.

There already exist various understandable classification mechanisms, which have been used for solving real-world applications [19]–[22]. From all, contrast pattern-based classification is about the most prominent [16], [18], because, for each decision, it provides an explanation, expressed in a language that is easy to understand for a human expert. Furthermore, for supervised learning, pattern-based classification has output more accurate results than other popular, state-of-the-art classifiers, including decision trees, naïve Bayes, nearest neighbor, bagging, boosting, and support vector machines [23].

The explanatory power of pattern-based classification may help identify what counts as non-human activity in an online service. Then, an explanation can be used to take legal action against, or to issue a formal complaint to an account holder. It can also be applied to forewarn a user about a possible misuse of the user account. Nevertheless, as far as we know, pattern-based classification has not been applied for social bot detection.

In this paper, we propose the use of a contrast pattern-based classifier for bot detection in Twitter, one of the most prominent social networks. From our results, we shall see that not only is our bot detection mechanism understandable, but that it also yields a classification performance that is competitive against other, well-known state-of-the-art classifiers. Moreover, we introduce a new feature model, which combines information extracted out of tweet content, tweet account, tweet account usage, and tweet content-sentiment analysis. We shall see that the use of our feature model improves the classification performance in a number of classifiers.

It is important to highlight that our proposed model, based on sentiment analysis and general information on the activity of an account, can be used to bot detection on Twitter, regardless whether the associated account tweets are written in either English or Spanish. As proof of the above, not only shall we provide classification results, but also a correlation analysis showing that the sentiment analysis for a text written in the English language does not change when it is applied to that text having been automatically translated into Spanish, and vice versa.

Note that a model having these characteristics would be useful for detecting bots in conversations that stem in communities from different countries, like Mexico and the US, where; for example, the construction of the wall, as proposed by the current US administration, has generated a strong debate in both English- and Spanish-speaking communities (https://twitter.com/hashtag/nowall). Also, multilingual models have proved to obtain better classification results than monolingual model because different languages contain different ambiguities and therefore present complementary views on the shared topic, which could be converging in the same expressed sentiment.

The remainder of the paper is organized as follows. Section II gives an overview of related research using either a supervised or an unsupervised approach to bot detection in social networks. Section III outlines contrast pattern-based classification. Section IV introduces our pattern-based proposal for bot detection in Twitter. Section V shows our experimental setup, where our feature model is presented. Section VI presents our experimental results as well as an in-depth result analysis. Finally, Section VII conveys the conclusions we have drawn from this research and indications for further work.

II. RELATED WORK

Since our research is to do with detecting the presence of a bot on a Twitter account, we shall focus this section's discussion on related research for bot detection in that social network.

Bots on Twitter are used with multiple motivations in mind. One group of bots, known as *spammers* or *content polluters*, are used to distribute spam, attract costumers to products, generate revenues, and disseminate malicious websites/programs [24]. Other groups are used to increase the popularity of an account, by adding followers or generating conversations about it, and to increase the influence the account has on the social network [25]. Another group is used for political purposes, to influence the population perception about a candidate, fake a grassroots movement [26], and to infiltrate social discussions to manipulate them [27]. Given the multiple uses of bots, it is a paramount problem to find methods to detect them.

Bot detection on Twitter is based on the premise that a genuine, human account displays a behavior that is different from the one output by a machine, regardless the machine level of automation or sophistication [24]. Commonly, to characterize human intervention behind a Twitter account, detection models make use of features extracted from five elements: the content of tweets posted from the account, how the poster is thought to feel about the topic contained in a tweet, general information about the account, account activity, and the associated account social network [28]. We will respectively refer to these kinds of features as: tweet content, tweet sentiment, account information, account usage, and social network structure. We outline these kinds of features below.

Tweet Content: a tweet content feature is one related with the message of a tweet and is obtained through message parsing analysis. Among others, it could be the number of words in a tweet, the use of capitalization and punctuation, the number and type of a sequence of n letters (called an n-gram), and whether the text resembles a known spam message [29]. Moreover, tweet content features also include special elements, such as the number of URLs, hashtags, or mentions to other users.

Tweet Sentiment: a sentiment feature is a piece of information extracted by applying sentiment analysis

to the text of a tweet [30]. This is used to determine how the poster feels about the topic of the associated tweet. We have separated this feature type from tweet content, as the feature is about how the poster feels, and not the text form per se. Sentiment features aim to measure different sets of sentiments about a topic, and the degree expressed to each sentiment. Sentiment analysis techniques are also used to compute whether the poster agrees or not with a topic, and the strength of such agreement [30].

Account Information: an account information feature is extracted from the Twitter account that has been allegedly used to post a message [14]. Features of this kind involve account creation date, Twitter username, account description, account language, account profile picture, description URL (if the account is verified), number of friends, number of followers, and the ratio between the two latter figures; among others. Since these features do not tend to change frequently, they do not need to be extracted with the same regularity than tweet content, or tweet sentiment features.

Account Usage: Account activity includes any measurement as to how regularly an account is used to post a tweet, the similarity between several tweets posted from the account, and how the user makes use of Twitter (for example, via a mobile device, a web interface, or an automated tool). In this type of feature, there are also statistical metrics computed from some of or all of the tweets in the account; these involve, for example, the number of different URLs and hashtags posted. Commonly, these statistical features are represented as a ratio between the number of occurrences of a distinctive element and the number of tweets.

Social Network Structure these features are used to measure message flow and account interaction. Here, we may have, for example, inter-account post similarity (in terms of content), or other typical behavior, including the number of replies or retweets, on a small-time period, to a single tweet, the number of bidirectional links between accounts (where a client's friend is also a client's follower), and general sentiment agreement about a topic in a group of accounts. The use of these features entails the observation of the behavior of multiple related accounts.

Twitter bot detection has been approached using a combination of these features, and, as expected, these appeared over time, being tweet content the earliest feature type supplied to a classifier. Bot detection has been approached using both types of learning: supervised (see Section II-A) and unsupervised (see Section II-B). In what follows, we shall discuss the most prominent bot detection mechanisms; in passing, for each mechanism, we also survey the feature space it has made use of, and the performance it has claimed to have attained. However, the reader should bear in mind that performance figures cannot be fairly compared because authors have used different experimental setups.

A. SUPERVISED APPROACH TO BOT DETECTION

The earliest supervised methods for bot detection work under the premise that bots have more friends than followers, post more URLs, mention more users (whether friends or not), and post very similar messages [14]. Lee *et al.* [31] proposed a classifier, called *Decorate*, which makes use of features of the following types: tweet content, tweet account, and account usage. Authors showed that *Decorate* achieved a performance of 0.88 using the F1 measure; in their experiments, [31] considered a dataset developed by their own. Later, Wang [32] proposed a method that adopts a similar approach, but it focuses only on the last 20 tweets issued by the account under observation. Using a naïve Bayes classifier, Wang [32] has reported on to have achieved an F1 equal to 0.91. Later, Ahmed and Abulaish [33] tested the performance of naïve Bayes, J48, and Jrip (taken from Weka data mining tool [34]), when each classifier is restricted to different feature subsets. They found that account usage features are more discriminative than tweet content ones.

To increase the discrimination capabilities of the classifiers, other approaches exploited more tweet usage features. For example, Chu *et al.* [24] included elements, such as how the client logs into the Twitter platform or the client regularity of tweet posting. They reported to have achieved 96% of recall, using Random Forest [35] as classifier. Yang *et al.* [36] used features of type tweet usage and social network; where they have obtained an F1 equal to 0.9 using Random Forest, while Decorate and Bayesian network classifiers obtained an F1 of 0.88 and 0.83, respectively. Recently, Gilani *et al.* [37] proposed to use tweet usage features, including the use of multimedia elements. In their experiments, they separated the accounts into different datasets, depending on the number of followers. They found that using their feature representation, the Random Forest classifier attained an F1 close or equal to 1.0. In this way, Gilani *et al.* [37] showed that using their proposal it is easier to detect bots in accounts with more than 10 million followers; they also found it is more difficult to spot a bot for accounts with less than one thousand followers, where they have achieved an F1 of 0.84.

Dickerson *et al.* [38] reported that using sentiment features, in a social network-based Random Forest, improves Bot detection. After, Wang *et al.* [29] proposed to use tweet content, sentiment, tweet account, and tweet usage features. Wang showed that using a Random Forrest on the dataset created by [32], they obtained an F1 equal to 0.94. However, [32] reported to have obtained a result contradicting to [38]: using features that are either tweet or tweet usage only they obtained an F1 of 0.9.

Davis *et al.* [28] proposed a bot detector (called Bot or Not) that combines all the features previously mentioned. Bot or Not uses a Random Forest classifier [35] over 1,000 features from all the categories, obtaining a 0.95 of Area Under the ROC Curve (AUC). It is important to highlight that Bot or Not has an API available to the public.[3]

[3]https://botometer.iuni.iu.edu

Later, Cresci *et al.* [39] proposed that not only are the actions but the sequence of the actions which allows discerning if an account is a bot. They use a DNA-like analysis, where they assign a letter to each tweet on an account, depending on the presence of specific tweet content features: if the tweet has an URL, mentions, hashtags, images, a combination of these, or none of these. Then, the DNA fingerprint of an account is the string with all the assigned letters of each tweet, appearing in order of posting. They hypothesize that bots should have longer common substrings than normal users do. To test this, they propose to calculate the length of the longest common substrings and consider those accounts with a value higher than a threshold as bots. They report an F1 of 0.97 using this approach. Nevertheless, it is well-known that the DNA-like analysis approach has high computational complexity.

Recently, in [14] a follow-up to [39], authors compare their method against Bot or Not [28], [33], [36], [40], the bots countermeasures implemented by Twitter, and a set of human annotators in datasets they collected. These datasets contained information from bots that mimic the behavior of genuine accounts to evade bot detectors. They found that neither Twitter nor the human annotators where very good when dealing with these accounts. Furthermore, they found that their approach, and the proposed by [33], was better than Bot or Not in all cases.

From the supervised approaches for bot detection on Twitter, we can observe that combining features of multiple sources generally increases the performance of the classifier. While there is not conclusive evidence that some features are better than others, tweet usage seems to allow for good classification performance. Furthermore, we can also observe that approaches that use Random Forest [35] obtain higher performance than those which do not. Then, based on this last sentence, using tree-based approaches can help to improve the state-of-the-art classification results attained till the date. Hence, pattern-based classification, coming from decision trees, can be a approach to take into account for improving the state-of-the-art results.

B. UNSUPERVISED APPROACH TO BOT DETECTION

While less popular than their supervised counterpart, unsupervised methods have been used to find differences in groups of bots and genuine users.

Ahmed and Abulaish [33] propose that bots can be used for spam campaigns, that is, multiple bots disseminating the same information. To test this, [33] proposed to find groups of spammers from an undirected fully connected graph where each node is a suspected spam account. Their graph contains the weight of the links, which is the similarity between accounts using tweet usage features. To find groups of suspected spam campaigns, they use the Markov clustering algorithm [41]. While they claim that campaigns were successfully detected, they do not give any performance measure.

After, Miller *et al.* [40] have proposed to use the clustering approach to find groups based on features of either tweet account or account usage. However, given that account usage measures could vary over time, they used DenStream [42] and StreamKM++ [43], which respectively are versions of DBScan and K-means++ for data streams. Using these clustering techniques, they have obtained an F1 of 0.88.

Later, in [44], a group of authors observed that it is highly unlikely that several users post a message with a related trending topic in the same second, or retweet less than 20 seconds after the message is posted. Using social network features, when the accounts are related if they post about the same trending topic, they find groups of bots for Twitter with a precision of 0.91.

From the related work, we have observed that decision tree-based approaches [45] obtain higher performance than those which do not. Also, we have observed that those representations containing features of the types: account usage, sentiment analysis, and tweet content, allow obtaining better classification results than other features representations. Also, we have noted that there is not pattern-based classification for bot detection on social networks.

III. INTRODUCTION TO CONTRAST PATTERN-BASED CLASSIFICATION

Contrast pattern-based classifiers provide a model that it is easy to understand for a human, and they have demonstrated to be more accurate than other popular classification models [15], [23]. Contrast pattern-based classification has been used on several real-world applications, such as characterization for subtypes of leukemia, prediction of heart diseases, gene transfer and microarray concordance analysis, structural alerts for computational toxicology, classification of spatial and image data, and gene expression profiles; among others [23]. Nevertheless, as far as we know, contrast pattern-based classifiers have been not used for bot detection. Contrast pattern-based classification would be a powerful model for bot detection because the explanatory power of contrast patterns can help a Twitter user (whether a human being or an organization) for example, take legal action, or issue a formal complaint against a service provider, or forewarn user's friends from malicious activity going on that jeopardize the user endeavor.

A *contrast pattern-based classifier* uses a collection of contrast patterns to create a model that classifies a query object in a predefined class [15]. A *pattern* is represented by a conjunction of relational statements, each with the form: $[f_i \ \# \ v_j]$, where v_j is a value in the domain of feature f_i, and # is a relational operator from the set $\{=, \neq, \leq, >\}$ [15], [23]. For example, $[Hour\_in\_Server \in [0, 5]] \wedge [Number\_of\_URL > 5] \wedge [Number\_of\_Follower \leq 10] \wedge [Mobile = \text{``}False\text{''}]$ is a pattern describing a collection of tweets issued from a botnet. Let p be a pattern and D be a dataset; then, the support of p is the fraction resulting from dividing the number of objects in D described (support) by p by the total number of objects in D. Now, a *contrast pattern* (CP) for a class c is a pattern whereby the support of CP for c is significantly higher than any support of CP for every class other than c [16], [23], [46].

For building a contrast pattern-based classifier, there are three phases: *Mining*, *Filtering*, and *Classification* [23], [47]. Mining is dedicated to finding a set of candidate patterns by an exploratory analysis using a search-space, which is defined by a set of inductive constraints provided by the user. Filtering is dedicated to select a set of high-quality pattern coming from the before phase. Classification is responsible for searching the best strategy for combining the information provided by a subset of patterns and so builds an accurate model based on patterns.

Using contrast pattern mining based on decision trees has advantages regarding those approaches not based on trees. First, the local discretization performed by decision trees with numeric features avoids doing an a priori global discretization, which might cause information loss. Second, with decision trees, there is a significant reduction of the search space of potential patterns [48].

García-Borroto *et al.* [48] performed several experiments for comparing different algorithms for mining contrast patterns based on decision trees. Authors claim that those contrast pattern miners taking into account the diversity approach has shown better classification results than other approaches. In their experiments, authors have shown that Bagging, Random Forest, and LCMine [49] are the contrast pattern mining algorithms that allow obtaining a collection of patterns which produce better classification results than other state-of-the-art solutions based on contrast patterns.

LCMine, proposed by [49], extracts a collection of patterns from a set of decision trees. LCMine selects the best k splits in the first levels and the best split at lower levels. In this way, LCMine creates 120 different trees where patterns are extracted from the paths from the root node to the leaves.

García-Borroto *et al.* [48] presented Bagging miner, which creates diversity by generating each tree with a bootstrap replicate of the training set. Bagging miner is an excellent way to obtain a diverse collection of patterns because small changes in the training sample do not impact significantly in the produced model, which comes from different decision trees [50]. In the same paper, authors explain that Random Forest Miner (RFMiner) creates diverse trees by selecting a Random Subset of features at each node. The success of Random Forest can also be explained because injecting randomness at the level of nodes tends to produce higher accuracy models. Finally, García-Borroto *et al.* [48] concluded that RFMiner and Bagging miners, which are based on diversity, are the best for mining patterns because they obtained more high-quality patterns than other well-known pattern miners. Additionally, authors claimed that LCMine, as a non-random miner, allowed obtaining competitive classification results.

After mining contrast patterns, one must carry out a filtering step. The main idea behind it is to reduce the number of contrast patterns without degrading the accuracy of the output pattern-based model. For filtering, some prominent strategies have been proposed, such as removing duplicated patterns, removing redundant items from patterns, and selecting only general patterns [5], [51]–[53].

Since contrast patterns are extracted from a collection of different decision trees using the same training dataset, the set of extracted contrast patterns usually contains duplicate patterns as well as patterns containing redundant items. On the one hand, a contrast pattern p_1 is labeled as duplicate if it contains the same items and covers the same objects than other contrast pattern p_2. On the other hand, a contrast pattern p_3 contains redundant items if at least one item is more general than another one, i.e., an item I_1 is more general than another item I_2 if all objects fulfilling I_1 also fulfill I_2 but not all objects fulfilling I_2 fulfill I_1.

A common filtering strategy for contrast patterns is to selected only those contrast patterns considered to be minimal [5], [51]–[53]. A contrast pattern is said to be minimal if if contains sub-patterns that are not contrast any more. These types of patterns are also the most general contrast patterns. Minimal patterns are interesting for describing a model because, in general, they contain a low number of items.

As stated above, filtering strategies allow having a set of contrast patterns with significantly fewer patterns than using all extracted patterns, but their main drawback is that, in some practical contexts, the number of selected patterns is yet impractical to be analyzed by experts.

After mining contrast patterns and filtering the extracted patterns, the next stage is classification. There are many contrast pattern-based classifiers reported on in the literature [18], [23], [54], [55]. One of the first strategies of classification based on patterns is using a scoring function; it has been adopted by many pattern-based classifiers, like CAEP [55]. This strategy uses a quality measure, like Support [46], for classifying a query object into the class with the highest sum of support. This sum is computed using all patterns covering the object to be classified.

Although, several contrast pattern-based classifiers have been proposed, PBC4cip, proposed by [23], has reported good results in both problems with and without class imbalance. In the training phase, PBC4cip weights the sum of supports in each class, for all patterns covering a query object, taking into account the class imbalance level. Another prominent pattern-based classifier is CAEP, proposed by [55], which aggregates the supports of the patterns matching the query object per class. Next, CAEP normalizes all the votes with the average votes per class for assigning the class with a higher vote to the query object. CAEP has shown good classification results in several contexts, but in a recent paper, PBC4cip improves the results obtained by CAEP [23].

IV. OUR APPROACH FOR BOT DETECTION IN TWITTER

Our approach for bot detection using pattern-based classification consists of four step: creating a new feature model based on a combination of tweet content-sentiment analysis and other pieces of information, stemming from tweets and the associated Twitter account (Section IV-A); inducing several diverse decision trees from our proposed feature representation by using different quality measures for

evaluating splits, where contrast patterns are extracted from each induced decision tree (Section IV-B); selecting the optimal contrast patterns covering all objects in the training dataset (Section IV-C); and classifying query objects by using a scheme based on the weighed sum of support of all patterns covering a query object (Section IV-D).

A. A NEW FEATURE MODEL FOR SOCIAL BOT DETECTION

The use of an appropriate feature model is fundamental for obtaining good classification results [56]. However, crafting a good feature model for social bot detection is not an easy task. For example, the core of a tweet is a piece of text, written in a given language. Then, any natural language processing statistical measure (such as *n*-gram frequency of occurrence) may vary from one language to the other. In addition, most datasets used in social bot detection contain posts written in the English language, which cannot be used to train a bot detection mechanism in a different language. As a consequence, it is complicated to detect bots on social media where people from different communities writing in different languages and referencing the same hashtag, for example, the hashtag *#NoWall* talking about the construction of the border wall, proposed by the current US administration (https://twitter.com/hashtag/nowall), contains posts written in both languages, English and Spanish. Another example is a blog of reviews or comments for a product, that could be written in different languages. Sentiment analysis that focuses only on a single language runs the risk of overlooking essential information in a collection of related texts.

To get around these issues, in this paper, we propose to use the sentiment analysis extracted from the text and the frequency data coming from tweets and twitter accounts. Specifically, we use sentiment analysis to determine whether a preprocessed tweet or the description of a tweet account expresses a *positive*, *negative*, or *neutral* sentiment, and what is its objectivity level and irony tone; for doing this, the local polarity of the different sentences in the text and the relationship between these are computed, resulting in a global polarity value for the entire text. To perform the sentiment analysis we used the cloud service API: ''Meaning Cloud''. Meaning Cloud computes local polarity of the different sentences in a text and the relationship between them, resulting in a global polarity value for the whole text. The service provides as output a JSON file with the following fields: *polarity, subjectivity, irony and emotional agreement* [30]. It performs sentiment analysis by using rule systems based on lexicons and deep linguistic analysis, offering the possibility of custom domain adaptations. Besides polarity at a sentence and a global level, the sentiment analysis service uses natural language processing techniques also to detect the polarity associated with both entities and concepts in the text (see www.meaningcloud.com for more information).

Then, similar to [30], for each text, we propose to extract the following five features based on sentiment analysis:

Sub: whether the text has an objective text.
Irony: whether the text has an ironic text.
Agreement: whether the text has an agreement text.
Confidence:]the percent of confidence regarding to the global polarity (Feeling) obtained from a text.
Feeling: the type of feeling issued by the posted tweet or the text describing a tweet account, which may be neutral (NEU), NONE, very negative (N+), negative (N), positive (P) or very positive (P+)

Also, we propose to compute the frequency of the last 200 tweets issued by a Twitter account for each hour and day of the week, taking as reference the time of the analyzed tweet. From this, 31 frequency features are extracted, which vary on the time.

In addition, we propose to extract eight additional feature regarding the time and date of the tweet and the Twitter account. These are:

tweet_year: the year when the tweet was created
tweet_month:the month when the tweet was created
tweet_day:the day when the tweet was created
tweet_hour:the hour when the tweet was created
user_created_year:the year when the Twitter account was created
user_created_month:the month when the Twitter account was created
user_created_day:the day when the Twitter account was created
user_created_hour:the hour when the Twitter account was created

Finally, we propose a new feature representation containing 49 features. Our representation is based on sentiment analysis extracted from the text posted in a tweet and the frequency data coming from tweets and twitter accounts.

B. MINING CONTRAST PATTERNS

Our proposal begins by inducing a decision tree, for doing this, it builds a root node with all objects of a training dataset D. Then, it splits the root node into two disjoint subsets (left child D^l and right child D^r), and repeats this process recursively over the children nodes until a certain *stopping criterion* is met. For splitting each node, we randomly select a subset of features F and, using the selected features, generate as many binary *splitting criteria* as possible, depending on the feature type.

Our proposal evaluates each binary candidate split at each decision tree level using three different measures (Hellinger distance [57], Bhattacharyya [58], and information gain [59], because these measures have demonstrated effective results in inducing decision trees [23], [59]–[61]. Our proposal continues to generate binary candidate splits until a stop criterion is met (*stopping criterion*). We used the following two stop criteria: (i) whether the node is pure (which means that all objects belong to the same class), and (ii) the quality measure takes the lowest value for all candidate splits [59].

The procedure explained above allows inducing only one random decision tree. However, extracting contrast patterns

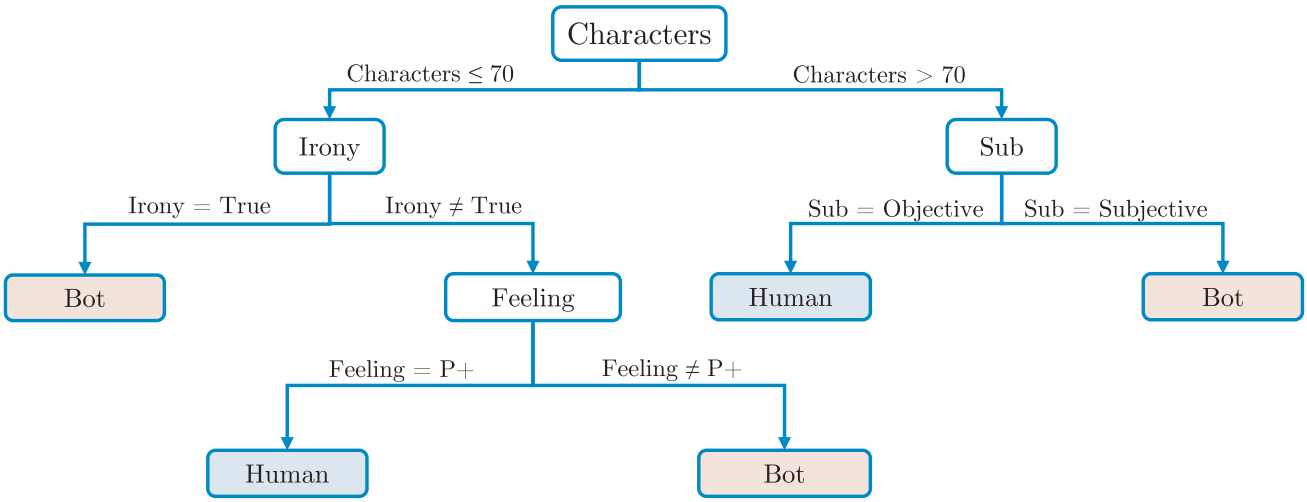

FIGURE 1. Decision tree example.

from only one random decision tree generates very few contrast patterns that, when used by a contrast pattern-based classifier, attain inferior classification results to those obtained when using contrast patterns extracted from several decision trees [49]. However, extracting patterns from several equal decision trees generates several duplicate patterns, leading to the same problem as when using only one decision tree. Extracting contrast patterns from a collection of diverse decision trees can mitigate these problems [49]. Therefore, we induce K decision trees by following our decision tree induction procedure, which, owing to the above random feature subset selection, allows for obtaining a collection of K diverse decision trees.

Once K diverse decision trees have been induced, our contrast pattern miner extracts all contrast patterns from each decision tree. Each pattern is the conjunction of the properties $f_i \ \# \ v_j$ in a path from the root node to a leaf node; that is, any path from the root to a leaf determines a conjunction of properties, which forms a pattern. Finally, only those patterns fulfilling the contrast pattern condition are preserved. For example, from the decision tree illustrated in Fig. 1 , the procedure extracts the following five contrast patterns:

$P_1 = [Characters \leq 70] \wedge [Irony = True]$
$P_2 = [Characters \leq 70] \wedge [Irony \neq True] \wedge [Feeling \neq P+]$
$P_3 = [Characters > 70] \wedge [Sub \neq Objective]$
$P_4 = [Characters \leq 70] \wedge [Irony \neq True] \wedge [Feeling = P+]$
$P_5 = [Characters > 70] \wedge [Sub = Objective]$,

from which P_1, P_2, and P_3 correspond to the *Bot* class and the remaining patterns (P_4 and P_5) correspond to the *Human* class.

C. CONTRAST PATTERN FILTERING

The forementioned phase allows mining several contrast patterns, but there could be duplicate patterns. In addition, this can produce specific contrast patterns and redundant items which can degrade the classification results of a contrast pattern-based classifier. For these reason, the intermediate step in our proposal is to eliminate duplicate and specific contrast patterns; additionally, redundant items are also removed from contrast patterns. For doing that and following the ideas of [5], [51]–[53], we take into account the following steps:

Removing duplicated contrast patterns: Several contrast patterns containing the same items and covering the same objects (duplicate patterns) are extracted because they come from several decision trees by using the same training dataset. Then, to reduce the size of the outcome, only one contrast pattern is selected from those containing the same items and covering the same objects.

Removing specific contrast patterns: Commonly, many specific patterns are extracted, which can be eliminated without degrading the classification results. Let P_1 and P_2 two contrast patterns from the same class, P_1 is more specific than P_2 if P_1 contains all the items in P_2 and at least one more. For example, let $P_1 = [Hour\_in\_server \leq 5] \wedge [Have\_photo\_profile = True] \wedge [Country = Cuba]$ and $P_2 = [Hour\_in\_server \leq 5] \wedge [Country = Cuba]$ be two patterns from the same class. Since all the items belonging to P_2 also belong to P_1 but P_1 has one more item, then P_1 is more specific. Therefore, as P_1 is more specific than P_2 and both are contrast patterns from the same class, and as was stated in [5] and [51]–[53], then P_1 should be removed.

Removing redundant items: An item I_1 is more general than another item I_2 if all objects fulfilling I_1 also fulfill I_2, but not all objects fulfilling I_2 fulfill I_1. We also say that I_2 is redundant with I_1. If two items in a pattern are redundant, the most general item is eliminated. An example of a pattern with redundant items is: $[[Hour\_in\_server > 5] \wedge [Hour\_in\_server > 6]]$,

which is simplified to [*Hour_in_server* > 6]; since a record with an hour greater than 6 is also greater than 5.

D. CONTRAST PATTERN-BASED CLASSIFICATION

As we have stated in Section III, there are several contrast pattern-based classifiers reported on in the literature [18], [23], [54], [55] but PBC4cip [23] has reported competitive results in both problems with and without class imbalance. Also, PBC4cip has obtained better classification results than other contrast pattern-based classifiers, such as CAEP, proposed by [55], iCAEP introduced by [54], SMOTE-TL+LCMine proposed by [18], and SJPE presented by [62].

Usually, bot detection is labeled as a class imbalance problem; in this type of problem, the objects (or post in our case) are not equally distributed among classes (human and bot), which produces a bias of classification results to the class with more objects. Also, frequently, the most interesting class, i.e., bot class, is the one that contains significantly fewer objects [23]. For this reason, we will use PBC4cip as the contrast pattern-based classifier in our experiments.

At the training stage, PBC4cip weights the sum of supports in each class by taking into account all contrast patterns covering a query object and class imbalance level. This training scheme is different from traditional classifiers, which only sum the supports. The weighted expression is:

$$w_c = \left(1 - \frac{|c|}{|T|}\right) \Big/ \sum_{p \in P} \textsf{support}\,(p, c) \tag{1}$$

where $|c|$ represents the number of objects belonging to the class c, $|T|$ is the number of objects in the training dataset, P is the set of all the patterns for the class c, and $\textsf{support}(p, c)$ is the support of the pattern p into the class c. This expression punishes the high sum of supports computed for the majority class.

$$W_{Sum\_Supp}(o, c) = w_c \sum_{\substack{p \in P \\ p \text{ covers } o}} \textsf{support}(p, c) \tag{2}$$

At the classification stage, the sum of supports in each class for all patterns matching with the query object o is computed by PBC4cip. This sum is also multiplied by the weight w_c of its corresponding class c. Thus, the query object is classified in the class where it reaches the highest value, according to (2).

V. EXPERIMENTAL SETUP

This section is devoted to showing data acquisition (Section V-A), and the databases used in our experiments (Section V-B), as well as, tested supervised classifiers (Section V-C), performance measure, and statistical methods (Section V-D) used to compare our proposal regarding other well-known classifiers.

A. DATA ACQUISITION

In our method, data acquisition was designated to work with collections of tweet accounts. As a consequence, we use datasets extracted from several databases proposed by [14] and other ones collected by us.

In order to corroborate our hypothesis that our proposed feature model is suitable for detecting bots coming from accounts posting tweets in both English and Spanish language, we used tweets written in English coming from the databases proposed by [14] and also, we collected tweets issued by prominent people talking in the Spanish language.

The databases proposed in [14] contain several tweets, which coming from political figure twitter accounts talking in the English language. In order to conserve the same domain, we select Mexican political figure twitter accounts for collecting tweets written in the Spanish language. For doing this, we selected the most prominent Mexican political figure twitter accounts,[4] namely: Andrés Manuel López Obrador (@*lopezobrador_*), José Antonio Meade (@*JoseAMeadeK*), Ricardo Anaya (@*RicardoAnayaC*), and Jaime Rodríguez Calderón (a.k.a "El Bronco") (@*JaimeRdzNL*); where their accounts were tracked during a period before the elections in Mexico.

For each Mexican political figure, we have collected the following data: all tweets posted by their Twitter account and other features, similar to the ones used in [14], coming from the issued tweet and the user accounts. The features selected are: 22 tweet content, 46 tweet account, 10 sentiment, and 31 tweet usage; for a total of 109 features. All of the data were gathered using the Twitter API[5] version 1.1.

We defined four time intervals for collecting tweets from the Mexican political figures: (i) from the start of electoral campaigns (March 31) to the first political debate (April 22), (ii) from the first political debate to the second political debate (May 20), (iii) from the second political debate to the third political debate (June 12), and (iv) from the third political debate to the final elections (July 1). As a consequence, we have extracted 3,519 tweets issued by the Mexican political figures during the aforementioned four time intervals.

In Table 1, we show the other databases used for bot detection, which contain millions of collected tweets coming from both human and bot. From these databases, we select a significant representation of the objects for each database shown in Table 1 (28,135 from genuine accounts database and 19,803 from the bot databases, equal to 47,938 in total) because processing all tweets with the the API for the tweet sentiment analysis is not suitable for this research (see developer.twitter.com for more information about the time break restriction).

B. DATABASES

To evaluate the performance of our proposal, we will use a set of 51,457 tweets from which 31,654 belong to human (28,135 from the genuine dataset shown in Table 1 and 3,519 collected by us) and the remaining (19,804) belong to bots from Table 1. This data follows a distribution of

[4]https://twitter.com/TwitterMexico/status/1070336958608609280
[5]https://developer.twitter.com

TABLE 1. Databases used in [14].

Database	Description	accounts	tweets	year
genuine accounts	Verified accounts that are human-operated	3,474	8,377,522	2011
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 in [11]	1,000	145,094	2009

TABLE 2. Databases used in our experiments.

Database	Description	#Features
Original	Containing all the original features proposed in [14]	50
DB1	Containing all the original features proposed in [14] and our proposed features	109
DB2	Containing a subset of the original features proposed in [14] and our proposed features	89
Our Feat.	Containing only our proposed features	59

imbalanced classes, which it is normal in the context of bot detection.

In order to see if our proposed features (stated in Section IV-A) allow obtaining better classification results than the original features proposed by [14] or a combination of them, we have tested all classifiers using different datasets, which contain combinations of the features proposed in [14] and those proposed by us (see Table 2 for more details).

C. STATE-OF-THE-ART CLASSIFIERS

For comparing the classification results obtained by our proposal regarding other state-of-the-art classifiers, we have selected the following classifiers:

BN:*Bayesian Network* [63] is a directed acyclic graph that models probabilistic relationships among a set of random variables from a datatset. Each vertex (node) of the graph represents a random variable and the edges capture the direct dependencies between the variables. The network encodes the conditional independence relationships that each node is independent of its non-descendants given its parents [64].

kNN: *k-Nearest Neighbor* [65] is used to evaluate new instances using the decision output by the closest k neighbors of the training set to a query instance, q. In our experiments, we have used $k = 3$ because several authors as [66] and [67] have reported it yields better classification results than other k values. Using $k = 3$ (3NN), the classifier uses the 3 closest neighbors to q and each one gives a class vote; the query instance then is classified adopting a majority-based policy.

C4.5: This classifier was proposed by [59]. It induces a decision tree by selecting, on each node, the best pair feature/value that tends to produce pure nodes. After the induction, a decision tree has a root node, decision nodes, and leaf nodes. The tree can be pruned to avoid overfitting where unnecessary branches are replaced by leaf nodes. Then, for classifying a query instance q through a decision tree, first, q is evaluated in the root node, and as a consequence, it decides which decision node follows, this procedure is repeated for each decision node until a leaf node is reached. This last node decides the class to be assigned to the query instance.

NBayes: *Naïve Bayes* [68] is a simple but efficient algorithm for supervised learning that has performed remarkably well in many applications. Naïve Bayes uses the Bayes rule, but making the (strong) assumption that all features are statistically independent.

RF: *Random Forest* [35] is an extension of the bagging approach. Random forest is a combination of tree-based classifiers; each tree is built with a random selection of instances of the training set containing different subsets of features. As in bagging, the query instance is classified into the class obtaining the majority vote.

SVM: *Support Vector Machine* [69] applies a kernel function to map the input instances into a high-dimensional feature space that splits the different classes and separates them as much as possible. For classification, each query instance is mapped to the same space, and the class is predicted by its position in the space.

LogRegression: *Logistic Regression* [70] is a statistical model whose objective is to predict the influence of different features using the probability of an event appearance. Logistic Regression can only be used to predict two-class problems.

AdaBoost.M1: *Adaptive Boosting* [71] is based on the idea of creating a highly accurate prediction rule classifier using many weak accurate rule classifiers. Each weak classifier should follow the weak learning condition, which means that the accuracy has to be a little bit over 50%; the random guessing. Generally, misclassification drops very quickly with each round of boosting but each round makes the system more complex, causing overfitting. We used the M1 version of AdaBoost composed by decision trees.

Bagging: It is a bootstrap aggregating classifier [72] that generates n claasifiers using a subset of the training dataset. This subset is obtained by sampling with replacement. For our experimental, decision tree classifiers was used as base classifier. Each tree has just about 63.2% of instances from the training dataset, while the rest is fulfilled with repeated instances. For classification, Bagging returns the most voted class among the n decision trees regarding a query instance.

PBC4cip: It is a contrast Pattern-based classifier designed for class imbalance problems [23]. PBC4cip, first, extracts a collection of contrast patterns from several decision trees. After, using the class imbalance level of the training dataset, it weights the support of each pattern. Finally, a query instance is classified into the class with the highest sum of supports.

MLP: *Multilayer Perceptron* [73] is an artificial neural network that uses back-propagation to classify query instances. An MLP is a fully connected network where each node contains neurons with logistic activation that belongs to a specific layer of the network. The input of the first layer are all features, for the next layers the input are the processed features on the previous layer with a certain weight, the last layer is known as output layer.

D. PERFORMANCE MEASURE AND STATISTICAL TESTS

There are several measures to evaluate the performance of a classifier. Nevertheless, the most used measures for class imbalance problems are the Area Under the Receiver Operating Characteristic curve (AUC) [74] and the Matthews correlation coefficient (MCC), which are suitable for assessing the classification results on bot detection. All our classification results are based on both AUC and MCC measures, averaged over 10-fold-cross-validation. AUC is computed as follows:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \tag{3}$$

where TP_{rate} is the ratio of objects belonging to the minority class (bot) that are well-classified and FP_{rate} is the ratio of misclassified objects belonging to the majority class (human).

By contrast, MCC computes a correlation coefficient for two-class classification results; it returns a value into the range $[-1, +1]$. A result equal to $+1$ represents a perfect classification result, 0 no better than a random result and -1 indicates the worst classification result. MCC is given by:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4}$$

where TP represents the number of true positives, FP represents the number of false positives, TN represents the number of true negatives, and FN represents the number of false negatives.

In machine learning, it is important to know if the obtained results are statistically different. As a consequence [75], [76] suggested some non-parametric tests and post-hoc procedures to be used in machine learning to perform a comparison among different classification results. In this paper, we applied a Friedman's test (as a non-parametric test), as suggested in [75] for knowing whether exist statistical difference among compared results. Nevertheless, the Friedman's test does not show in a detailed form those results having statistical differences, as a consequence, a post-hoc procedure should be executed. Then, we performed the Shaffer static procedure (as a post-hoc procedure), because it is more powerful than the classical Nemenyi and Holm procedures and it is less computationally expensive than the Bergmann-Hommel's dynamic procedure [76].

A compact way, proposed in [75], to show the statistical results is through CD (critical distance) diagrams. These diagrams present the order of the compared classifiers based on the Friedman ranking, the magnitude of the differences between them, and the significance of the observed differences, all in a compact form. In a CD diagram, the rightmost classifier is the best classifier, the position of a classifier within the segment represents its rank value, and if two or more classifiers share a thick line it means they have statistically similar behavior.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

In order to analyze the obtained experimental results, we split this section into three subsections: first, we present a correlation analysis based on the Kendall τ procedure for knowing whether our feature model based on sentiment analysis is suitable for detecting bots in posts written in both English and Spanish language (Section VI-A); after, we show all the classification results of those classifiers based on patterns and other state-of-the-art classifiers not based on patterns (Section VI-B); and finally, we present an in-depth analysis of the extracted patterns (Section VI-C).

A. CORRELATION ANALYSIS

In [77], two main rules of thumb for a correlation analysis were proposed:

- Sample sizes larger than 30 and less than 500 are appropriate for most research.
- In multivariate research, the sample size should be several times (preferably 10 times or more) as large as the number of variables in the study.

Following the above rules, first, we have taken 1,000 random tweets from the database containing genuine accounts and others 1,000 random tweets from those databases containing bot accounts (see Table 1). Next, for each one of the selected tweets, we translate its text using the API provided by Google translate[6]; and after, we obtain the sentiment analysis from each one of the 4,000 tweets (2,000 in Spanish and 2,000 in English). After that, we performed 40 correlation analysis based on the Kendall τ procedure. For each feature extracted from sentiment analysis (Sub, Irony, Agreement, and Feeling), we performed 10 correlation analysis of 200 tweets (without replacement), as explained Fig. 2.

Based on the three evaluation measures for computing the correlation coefficient proposed in [78], there is a linear relationship when $C_{reults} \geq 2/\sqrt{n}$ (Rule 1), $C_{reults} \geq 2/\sqrt{n+1}$ (Rule 2), and $C_{reults} \geq 2/\sqrt{n+2}$ (Rule 3). Where C_{reults} is the correlation results and n the size of the sample.

Table 3 shows for each class and each analyzed feature, its averaged correlation result (Corr.), its standard deviation (Std.), and if there exist a significant correlation (Sig.) for all feature analyzed in each class. From this table, we can

[6]https://cloud.google.com/translate/docs/translating-text

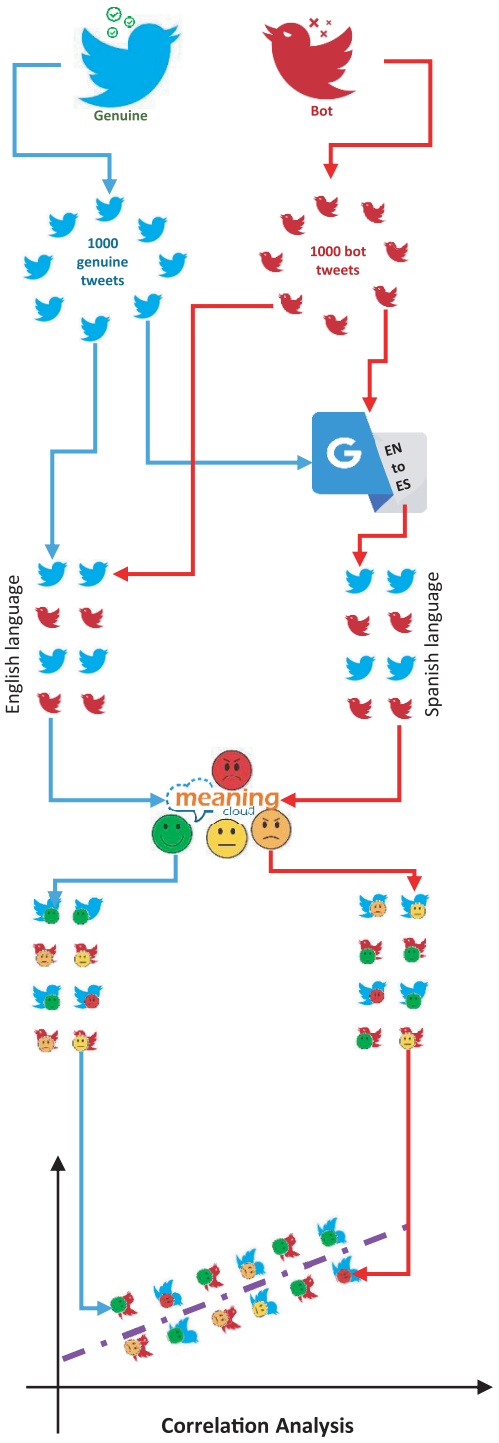

FIGURE 2. Visual explanation of the executed correlation procedure.

see that there is a significant positive correlation between the sentiment analysis extracted from a tweet posted in English language and the sentiment analysis extracted from its translation to the Spanish language. Finally, from this analysis, we can conclude that our feature model based on the sentiment analysis is suitable for analyzing tweets coming from both English and Spanish language.

TABLE 3. Averaged correlation results.

Class	Feature	Corr.	Std.	Sig.
Geniune	Sub	0.44	0.04	✓
	Irony	0.39	0.24	
	Agreement	0.39	0.14	
	Feeling	0.46	0.09	
Bot	Sub	0.41	0.05	✓
	Irony	0.57	0.01	
	Agreement	0.31	0.14	
	Feeling	0.45	0.07	

B. CLASSIFICATION RESULTS

In this section, first, we show the average AUC and standard deviation values obtained by the tested classifiers and after, we show the statistical results obtained from classification results of each tested classifier.

Table 4 and Table 5 the average AUC and the average MCC, respectively; they also include the associated standard deviation (STD) output by running the classifiers on the datasets shown in Table 2. In these tables, we also show each tested combination based on patterns such as a method for mining contrast patterns (LCMine [49] or RFm [48]), a measure for evaluating a split criteria on a decision tree (Hellinger distance [57], Bhattacharyya [58], and information gain [59]), if it was filtered (Fil), and the contrast pattern-based classifier used (PBC4cip [23]).

From tables 4-5, we can see that our proposal of using contrast patterns obtains the best average results for both measures (AUC and MCC) and the lowest standard deviation (STD) considering all the validation set. For example, the combinations (LCMine+Hell)+PBC4cip and (LCMine+Hell+Fil)+PBC4cip attained more than 0.9935 of AUC for each tested database, obtaining the best AUC final average (0.9976) and the lowest standard deviation (0.0024). Also, taking into account all tested classifiers, we can see that the representation proposed by us jointly with the representation proposed in [14] obtained the best average AUC (0.9384) and the representation proposed by us attained the lowest standard deviation (0.0833). In the same vein, notice that the combinations (LCMine+Hell)+PBC4cip and (LCMine+Hell+Fil)+PBC4cip attained the best average MCC (0.955) and the lowest standard deviation (0.0048).

Also, based on tables 4-5, we can see that there are not significant differences among our proposals when they use, or without using, the filtering method. Also, we have noted that the classifier Random Forest [35] obtains the best classification results for the databases Original and DB1, which is in correspondence with the stated in the literature (see Section II). In similar way, we note that the classifier Adaboost obtains the best classification results for the databases DB2 and Our Feat.

Notice, from Tables 4-5, that all the classifiers used in our experimentation output a good average AUC for all databases; the only exception to this is (LCMine+Bhatt) + PBC4cip with and without filtering, which obtained bad classification results for the databases Original, DB1, and DB2.

TABLE 4. Averaged AUC values obtained by the tested classifiers on databases shown in Table 2.

Classifiers	Original	DB1	DB2	Our feat.	Average	STD
BN	0.9995	0.9993	0.9918	0.7940	0.9455	0.0951
C4.5	0.9746	0.9871	0.9971	0.9899	0.9914	0.0042
kNN	0.9995	0.9994	0.9948	0.9125	0.9758	0.0399
NBayes	0.9365	0.9535	0.9018	0.6601	0.8642	0.1279
Adaboost.M1	0.9746	0.9909	**0.9999**	**0.9963**	0.9904	0.0097
Bagging	0.9745	0.9887	0.9964	0.9920	0.9879	0.0082
LogRegression	0.9995	**0.9999**	0.9697	0.8110	0.9450	0.0783
SVM	0.9323	0.9355	0.8409	0.8231	0.8830	0.0513
RF	**0.9999**	**0.9999**	0.9985	0.9841	0.9956	0.0066
MLP	NA	NA	NA	0.9554	NA	NA
(LCMine+Bhatt)+PBC4cip	0.5016	0.5016	0.5020	0.9937	0.6247	0.2130
(LCMine+Hell)+PBC4cip	0.9977	0.9995	0.9996	0.9936	**0.9976**	**0.0024**
(LCMine+InfoGain)+PBC4cip	0.9740	0.9733	0.9906	0.9936	0.9829	0.0093
(LCMine+Bhatt)+Filt+PBC4cip	0.5016	0.5016	0.5020	0.9937	0.6247	0.2130
(LCMine+Hell)+Filt+PBC4cip	0.9977	0.9995	0.9996	0.9936	**0.9976**	**0.0024**
(LCMine+InfoGain)+Filt+PBC4cip	0.9740	0.9733	0.9906	0.9936	0.9829	0.0093
(RFm+Bhatt)+PBC4cip	0.9086	0.9636	0.9961	0.9539	0.9555	0.0313
(RFm+Hell)+PBC4cip	0.9944	0.9967	0.9987	0.9508	0.9851	0.0199
(RFm+InfoGain)+PBC4cip	0.9666	0.9878	0.9980	0.9535	0.9765	0.0174
(RFm+Bhatt)+Filt+PBC4cip	0.9239	0.9684	0.9966	0.9537	0.9607	0.0262
(RFm+Hell)+Filt+PBC4cip	0.9883	0.9956	0.9983	0.9499	0.9830	0.0195
(RFm+InfoGain)+Filt+PBC4cip	0.9754	0.9913	0.9984	0.9536	0.9797	0.0172
Average	0.9283	**0.9384**	0.9363	0.9362	-	-
STD	0.1409	0.1427	0.1459	**0.0833**	-	-

TABLE 5. Averaged MCC values obtained by the tested classifiers on databases shown in Table 2.

Classifiers	Original	DB1	DB2	Our feat.	Average	STD
BN	0.9990	0.9983	0.9847	0.6098	0.8980	0.1665
C4.5	0.9582	0.9789	0.9949	0.9796	0.9779	0.0130
kNN	0.9990	0.9988	0.9904	0.8208	0.9523	0.0760
NBayes	0.8540	0.8913	0.7838	0.3261	0.7138	0.2271
Adaboost.M1	0.9582	0.9805	**0.9998**	0.9922	0.9827	0.0157
Bagging	0.9578	0.9816	0.9941	0.9843	0.9795	0.0133
LogRegression	0.9989	0.9999	0.9488	0.6234	0.8928	0.1569
SVM	0.8883	0.8938	0.7511	0.6465	0.7949	0.1030
RF	**0.9999**	**0.9999**	0.9972	0.9698	0.9917	0.0127
MLP	NA	NA	NA	0.9603	NA	NA
(LCMine+Bhatt)+PBC4cip	0.0423	0.0423	0.0487	**0.9877**	0.2803	0.4084
(LCMine+Hell)+PBC4cip	0.9963	0.9992	0.9992	0.9874	**0.9955**	**0.0048**
(LCMine+InfoGain)+PBC4cip	0.9581	0.9577	0.9847	0.9874	0.9720	0.0141
(LCMine+Bhatt)+Filt+PBC4cip	0.0423	0.0423	0.0487	**0.9877**	0.2803	0.4084
(LCMine+Hell)+Filt+PBC4cip	0.9963	0.9992	0.9992	0.9874	**0.9955**	**0.0048**
(LCMine+InfoGain)+Filt+PBC4cip	0.9581	0.9577	0.9847	0.9874	0.9720	0.0141
(RFm+Bhatt)+PBC4cip	0.8415	0.9414	0.9932	0.9125	0.9221	0.0548
(RFm+Hell)+PBC4cip	0.9905	0.9943	0.9974	0.9069	0.9723	0.0378
(RFm+InfoGain)+PBC4cip	0.9400	0.9800	0.9962	0.9108	0.9568	0.0335
(RFm+Bhatt)+Filt+PBC4cip	0.8784	0.9492	0.9941	0.9115	0.9333	0.0431
(RFm+Hell)+Filt+PBC4cip	0.9794	0.9927	0.9968	0.9054	0.9686	0.0370
(RFm+InfoGain)+Filt+PBC4cip	0.9597	0.9857	0.9970	0.9107	0.9633	0.0332
Average	0.8599	0.8783	0.8750	**0.8898**	-	-
STD	0.2715	0.2748	0.2778	**0.1687**	-	-

Nevertheless, for our proposed feature representation these proposals obtained good classification results.

It is important to highlight that in Tables 4-5 the classifier MLP continues executing, after eight months, the experiments for the databases: Original, DB1, and DB2. This is because MLP takes as an input numerical data, and to convert the attributes of these databases from categorical to numerical, we use one-hot encoding. This results in a feature vector of 125 attributes for DB3, 24,493 for DB2, 117,529 for DB1, and 117,411 for Original. The length of the encoded feature vector heavily affects the execution time, making MLP unsuitable for this problem. As a consequence, these results appear as NA in tables 4-5 and this classifier was not taken into account in the statistical tests. However, note that for our representation this classifier obtained more than 0.95 of AUC and its execution time was similar to other tested classifiers, making our feature representation suitable for the MLP-based approach.

Table 6 shows the average ranks obtained by each tested classifier in the Friedman's test (as a non-parametric test) by using the AUC results. The p-value computed by the Friedman's Test was 0.009413, which reveals that there are significant differences among the classification results of the compared classifiers. The Friedman's Test reveals whether differences exist (as a Boolean output) among the compared results but it does not offer which results are different among

TABLE 6. Average rankings based on the friedman's procedure for all tested classifiers.

Algorithm	Ranking
(LCMine+Hell)+PBC4cip	4.25
(LCMine+Hell)+Filt+PBC4cip	4.25
RF	4.25
Adaboost.M1	5.625
(RFm+Hell)+PBC4cip	8.25
(RFm+InfoGain)+Filt+PBC4cip	9.25
(RFm+Hell)+Filt+PBC4cip	9.75
kNN	9.75
LogRegression	10.25
C4.5	10.375
BN	10.5
Bagging	10.5
(LCMine+InfoGain)+PBC4cip	12.25
(LCMine+InfoGain)+Filt+PBC4cip	12.25
(RFm+InfoGain)+PBC4cip	12.25
(RFm+Bhatt)+Filt+PBC4cip	14
(RFm+Bhatt)+PBC4cip	14.75
(LCMine+Bhatt)+PBC4cip	16
(LCMine+Bhatt)+Filt+PBC4cip	16
Nbayes	18.25
SVM	18.25

them, as a consequence, a post-hoc procedure should be executed for obtaining, in a detailed way, which classifiers are different among them. For these reasons, we performed the Shaffer statistic test, which is more powerful than other classical procedures like Nemenyi and Holm and it is less computationally expensive than others like the Bergmann-Hommel's dynamic procedure [79].

In Fig. 3, we show the post-hoc results in a CD diagram. In this type of diagram the rightmost classifier is the best classifier. The position of the classifier within the segment represents its rank value, and if two or more classifiers share a thick line it means that they have statistically similar behavior.

From Table 6 and Fig. 3 we ca see that RF, (LCmine+Hell)+Fil+PBC4cip, (LCmine+Hell)+PBC4cip, and Adaboost.M1 are the best ranked and they have statistical differences among the remaining tested classifiers. Note that among these classifiers, Adaboost.M1 has a slightly worse performance than the remaining three classifiers better ranked.

Additionally, note that the top ranked classifiers are RF, (LCmine+Hell)+Fil+PBC4cip, and (LCmine+Hell)+ PBC4cip. These last two contain the same contrast pattern miner, measure for evaluating node splitting, and contrast pattern-based classifiers. The only difference is that one applies a filtering method for patterns (+Fil) and the other one does not. For pattern-based classification is better a model applying filtering because it guarantees to obtain a model with fewer patterns, which make it more easy for understanding. As a consequence, we focus our discussion between the model provided by RF and our proposal of using contrast patterns, which are tied as the best-ranked classification results.

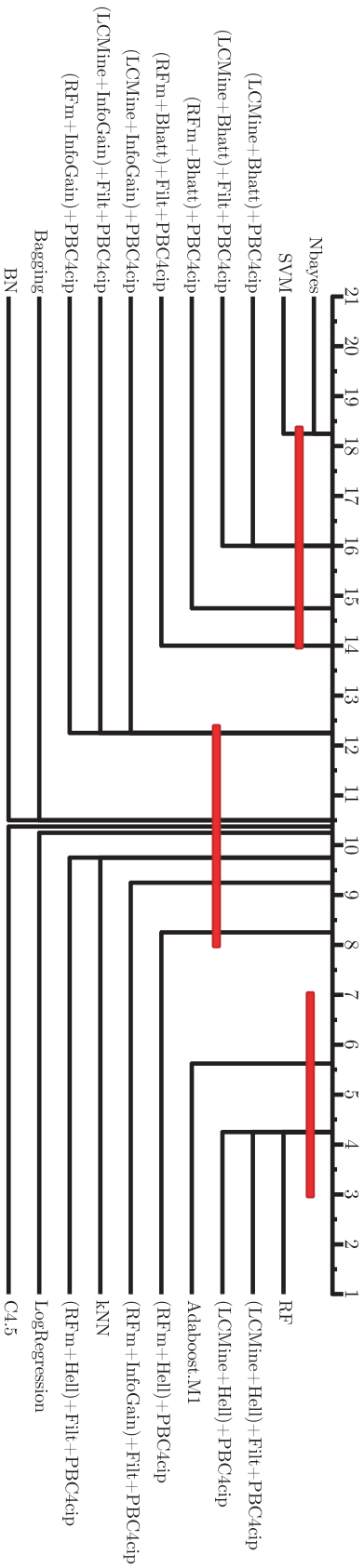

FIGURE 3. CD diagram with a statistical comparison of the AUC results for all classifiers over all the tested databases.

TABLE 7. Examples of extracted contrast patterns.

Class	ID	Items	Supp
Bot	CP_1	$[tweet\_source = \text{“}TweetAdderv4\text{”}]$	0.66
	CP_2	$[user\_friends\_count > 1054.50] \wedge [user\_favourites\_count \leq 2.50]$	0.62
	CP_3	$[tweet\_num\_mentions \leq 0.50] \wedge [user\_friends\_count > 1054.50] \wedge [user\_favourites\_count \leq 2.50]$	0.60
	CP_4	$[user\_favourites\_count \leq 10.50 \wedge [tweet\_year \leq 2014] \wedge [user\_friends\_count \in [1054.50, 2089.50]]$	0.59
	CP_5	$[user\_H16\_TweetDist > 6.50] \wedge [tweet\_year \leq 2014] \wedge [tweet\_retweet\_count \leq 0.50] \wedge [user\_favourites\_count \leq 4.50] \wedge [user\_H04\_TweetDist \leq 10.50] \wedge [tweet\_day > 1.50]$	0.50
Human	CP_6	$[tweet\_year > 2014]$	0.56
	CP_7	$[user\_followers\_count > 12.50] \wedge [user\_geo\_enabled = \text{“}1\text{”}] \wedge [user\_favourites\_count > 380.50] \wedge [user\_screen\_name\ != \text{“}\_iHATEMOON\text{”}] \wedge [user\_statuses\_count > 571.50]$	0.48
	CP_8	$[user\_geo\_enabled = \text{“}1\text{”}] \wedge [user\_favourites\_count > 380.50] \wedge [user\_profile\_text\_color\ != \text{“}F70A0A\text{”}] \wedge [user\_profile\_sidebar\_border\_color\ != \text{“}CC3366\text{”}]$	0.48
	CP_9	$[user\_geo\_enabled = \text{“}1\text{”}] \wedge [user\_time\_zone! = \text{“}EasternTime(US\&Canada)\text{”}] \wedge [user\_favourites\_count > 170.50]$	0.37
	CP_{10}	$[tweet\_num\_mentions > 0.50] \wedge [user\_descrip\_sa\_score\_tag = \text{“}NONE\text{”}] \wedge [tweet\_month \leq 5] \wedge [user\_H05\_TweetDistt \leq 1] \wedge [user\_favourites\_count > 208]$	0.26

On the one hand, as we have stated in Section II, RF have achieved good classification results in the bot detection context. On the other hand, contrast pattern-based classification has showed good classification results in several contexts. Nevertheless, as far as we know, there is not a study showing the use of contrast pattern-based classification in the bot detection context. From our experimental results, we can see that both approaches (decision tree and contrast pattern) obtain the best position into the Friedman's ranking. Also, they have statistical differences with many state-of-the-art classifiers, such as SVM, NBayes, C4.5, Bagging, BN and kNN.

Note that RF is based on building several decision trees, which provides a model that could be converted into rules, but it contains significantly more rules than those pattern extracted from a contrast pattern-based model. A model with fewer patterns or rules is more understandable by a human decision-maker. Hence, we recommend to use our approach based on contrast pattern for bot detection.

C. EXTRACTED PATTERNS

It is essential to know the extracted patterns by the contrast pattern-based approach to provide an explanatory model that help experts to take legal action or forewarn a company from malicious activity. In this section, we show a study about the contrast patterns extracted by the most accurate pattern-based model shown in Section VI-B.

In Section III we have stated that contrast pattern-based classifiers provide a model that it is easy for a human to understand, and they have demonstrated to be more accurate than other popular classification models [15], [23]. In order to show the potential of the patterns, we have selected a random subset of the extracted patterns in our experiments. In Table 7, we show five contrast patterns, selected randomly, by class; where, we also show the support of each pattern as well as their items.

From Table 7, we can see that contrast patterns from the bot class have more support than those patterns from the human class. Also, we can see that some patterns, like CP_1 and CP_6, have only one item and more than 0.55 of support. These patterns can be easily understood by an expert and can describe more than the 55% of the objects in a dataset. Also, we can see that the feature *user_favorites_count* is a frequent feature in several patterns shown in Table 7. This frequent feature describes the number of accounts that the user has bookmarked as favourite in the tweet account. Note that based on the examples of contrast patterns shown in Table 7, the item [*user_favorites_count* $\leq$ 3] defines a bot access while [*user_favorites_count* $>$ 170] defines a human access. Finding bots based on the number of accounts that the user has bookmarked as favorite is consistent with the method proposed in [28]. Note however, that contrast patterns can be combined with other features and as a result, we can obtain contrast patterns (see CP_2) describing a bot behavior for more than the 60% of the objects previously classified as bot in the dataset. In addition, note that our proposed feature model, containing information from data usage and sentiment analysis, is into some items of the extracted patterns (See CP_5 and CP_10); although, commonly these patterns have more amount of items than those whose not contain our proposed features.

In Table 8, we show statistics of the extracted contrast patterns from the bot class with the aim of seeing the characteristics of the models produced by each combination of classification based on contrast patterns tested in our experiments. From Table 8, we can see that every combination of the RFm miner extracts significantly more patterns with more items and more support than those combinations obtained using the LCMiner. Also, we have noted that there is not a statistical difference between combinations using filtering methods for patterns and those combinations not using filtering methods.

In addition, we can note that those combinations using Bhattacharyya, as a split criteria on a decision tree, extracted fewer patterns and with lower support than the other tested combinations. As a result, those models using the Bhattacharyya measure attained the lowest classification results.

Also, from Table 8, we can see that combining the RFm miner jointly with the Hellinger distance allows obtaining the best trade-off between the number of extracted patterns and average support for each tested database. These results are consistent with other ones presented in for a set of problems, which do not include the bot detection context [23].

TABLE 8. Statistics of extracted contrast patterns from the bot class.

Miner	Database	Eval.	Fil.	Average length	STD length	Average total	STD total	Average support	STD support
LCMiner	Original	Bhatt	Yes	2.46	0.97	9.20	1.17	0.0006	0.0001
			No	2.46	0.97	9.20	1.17	0.0006	0.0001
		Hell	Yes	6.69	2.02	537.60	27.86	0.0111	0.0056
			No	6.69	2.02	537.60	27.86	0.0111	0.0056
		InfoQuin	Yes	5.42	1.53	235.60	15.15	0.0537	0.1294
			No	5.48	1.53	235.60	15.15	0.0537	0.1294
	DB1	Bhatt	Yes	2.46	0.97	9.20	1.17	0.0006	0.0004
			No	2.46	0.97	9.20	1.17	0.0006	0.0004
		Hell	Yes	7.43	2.11	694.80	34.74	0.0565	0.0108
			No	7.43	2.11	694.80	34.74	0.0565	0.0108
		InfoQuin	Yes	5.66	1.57	250	13.46	0.1424	0.0535
			No	5.66	1.57	250	13.46	0.1424	0.0535
	DB2	Bhatt	Yes	2.97	1.07	40.80	2.86	0.0006	0.0005
			No	2.97	1.07	40.80	2.86	0.0006	0.0005
		Hell	Yes	7.51	2.08	887.9	52.47	0.0575	0.0123
			No	7.51	2.08	887.9	52.47	0.0575	0.0123
		InfoQuin	Yes	5.97	1.71	358	46.16	0.0956	0.0329
			No	5.97	1.71	358	46.16	0.0956	0.0329
	Our feat.	Bhatt	Yes	11.12	2.76	9693.90	360.28	0.0218	0.0045
			No	11.12	2.76	9693.90	360.28	0.0218	0.0045
		Hell	Yes	10.23	2.78	10246.5	375.72	0.0210	0.0043
			No	10.23	2.78	10246.5	375.72	0.0210	0.0043
		InfoQuin	Yes	10.93	2.78	10246.5	375.72	0.0210	0.0043
			No	10.93	2.78	10246.5	375.72	0.0210	0.0043
RFm	Original	Bhatt	Yes	4.92	2.46	214.90	25.50	0.0523	0.0082
			No	4.93	2.49	230.90	27.91	0.0538	0.0085
		Hell	Yes	9.40	3.79	2664.30	260.27	0.0361	0.0056
			No	9.35	3.76	2667.50	221.70	0.0380	0.0059
		InfoQuin	Yes	7.48	2.96	1628.50	93.59	0.0047	0.0009
			No	7.48	2.97	1665.50	130.63	0.0494	0.0098
	DB1	Bhatt	Yes	7.31	3.51	671.70	104.97	0.0434	0.0078
			No	7.37	3.47	691.90	45.20	0.0434	0.0080
		Hell	Yes	11.09	4.18	4977	342.34	0.0304	0.0052
			No	11.13	4.18	5239.10	290.12	0.0281	0.0048
		InfoQuin	Yes	9.30	3.47	3314.60	225.54	0.0380	0.0073
			No	9.21	3.39	3352.80	174.93	0.0382	0.0075
	DB2	Bhatt	Yes	9.59	3.78	1658.50	191.67	0.0362	0.0061
			No	9.51	3.64	1856.90	217.94	0.0338	0.0058
		Hell	Yes	11.36	3.75	8795.10	278.27	0.0269	0.0044
			No	11.34	3.75	8777.10	398.18	0.0271	0.0045
		InfoQuin	Yes	10.40	3.40	6790.70	336	0.0309	0.0050
			No	10.42	3.39	6996.10	331.62	0.0304	0.0048
	Our feat.	Bhatt	Yes	13.80	4.12	41735.40	674.80	0.0109	0.0018
			No	13.77	4.10	41647.80	700.59	0.0109	0.0019
		Hell	Yes	13.67	4.10	43540.50	347.55	0.0107	0.0018
			No	13.64	4.09	43586.60	583.17	0.0107	0.0018
		InfoQuin	Yes	13.31	3.82	43949.80	751.49	0.0121	0.0018
			No	13.33	3.84	44078.40	814.61	0.0120	0.0018

A possible explanation is that the Hellinger distance is unaffected by the class imbalance problem because it rewards, in a better way than the other compared measures, those candidate splits that maximize the TP_{rate} while minimizing the FP_{rate}. The higher the TP_{rate} value, the more bot objects are well detected.

Additionally, note that for each tested combination, our proposed feature model allows obtaining more patterns but they show lower support regarding the other tested representation. Finally, we have noted that the representation used in DB2, which is a combination of our proposed representation and the one proposed by [14], allows obtaining the best model based on patterns. This model contains patterns with high support and lowest standard deviation support for each tested combination shown in Table 8.

From the extracted contrast patterns from the human class, we have noted a similar behavior in each tested combination. Nevertheless, we can highlight that the patterns extracted from the human class have more items, lower support, and higher standard deviation support than those patterns extracted from the bot class, for each tested combination. As a consequence, we have noted that each pattern-based model produced contains significantly more patterns from the human class than the bot class. As a result, we believe more easy for an expert to understand those patterns coming from bot accesses than those coming from human accesses.

VII. CONCLUSIONS AND FUTURE WORK

Bot detection problems have become a significant research area due to many companies are investing resources in detecting abnormal behavior in their accesses. As a consequence, several classifiers have been proposed for bot detection, but there is no proposal using understandable models for detecting bots on social networks. Hence, in this paper, we propose to applying contrast pattern-based classifiers for detecting bot behaviors on the social network, Twitter.

From classification results obtained by 21 classifiers, we show that Random Forest and the contrast pattern-based classification are the proposals with the best classification results. Although, Random Forest have proved good classification results on in the literature and in our experimental results; it generates significantly more rules than the patterns extracted by using our proposal.

In addition, contrast pattern-based approach obtained classification results over 0.90 of AUC and 0.91 of MCC for all tested combinations except when the Bhattacharyya function was used. Besides, from our analysis of the extracted patterns, we can conclude that the contrast pattern filtering method used did not provide better classification results, and the reduction of the number of patterns was not significant. Also, the average support of the patterns is low for all combination based on patterns; nevertheless, the average length is good (short) for all combination. For a pattern: the fewer items, the best understandable.

From our experimental results, we have shown by means of a correlation analysis based on the Kendall τ procedure that the sentiment analysis is suitable for bot detection, indistinctly if the text was issued in English or Spanish language. Additionally, we showed that our proposed feature model jointly with the feature model proposed by [14] allows obtaining better classification results for each tested classifier than using them alone.

Finally, it is important to highlight that the model provided by us can be understandable by experts in the application domain. Also, the explanatory power of contrast patterns can help social networks to take legal action, issue a formal complaint against a service provider, or forewarn another company from malicious activity going on that may jeopardize its reputation.

As future work, we plan to improve the filtering method to tune up it to the bot detection problems and, as a consequence, obtaining a reduced set of good-quality patterns for bot detection. Also, we want to extend this work to other social networks, such as Facebook, Instagram, and Google Plus, with the aim of creating a model more general for bot detection on social networks.

REFERENCES

[1] M. Debashi and P. Vickers, "Sonification of network traffic for detecting and learning about botnet behavior," *IEEE Access*, vol. 6, pp. 33826–33839, 2018.

[2] I. Ghafir *et al.*, "BotDet: A system for real time botnet command and control traffic detection," *IEEE Access*, vol. 6, pp. 38947–38958, 2018.

[3] J. Woo, S. W. Kang, H. K. Kim, and J. Park, "Contagion of cheating behaviors in online social networks," *IEEE Access*, vol. 6, pp. 29098–29108, 2018.

[4] J. Á. Cid-Fuentes, C. Szabo, and K. Falkner, "An adaptive framework for the detection of novel botnets," *Comput. Secur.*, vol. 79, pp. 148–161, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404818309805

[5] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000923

[6] A. Mostrous, M. Bridge, and K. Gibbons. (2017). *Russia used Twitter bots and trolls 'to disrupt' Brexit Vote*. [Online]. Available: https://www.thetimes.co.uk/edition/news/russia-used-web-posts-to-disrupt-brexit-vote-h9nv5zg6c

[7] A. Bessi and E. Ferrara. (Nov. 2016). *Social Bots Distort the 2016 U.S. Presidential Election Online Discussion by Alessandro Bessi and Emilio Ferrara*. [Online]. Available: http://firstmonday.org/ojs/index.php/fm/article/view/7090

[8] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini. (2017). "Online human-bot interactions: Detection, estimation, and characterization." [Online]. Available: https://arxiv.org/abs/1703.03107

[9] S. Kramer. (2017). *Identifying Viral Bots and Cyborgs in Social Media*. [Online]. Available: https://www.oreilly.com/ideas/identifying-viral-bots-and-cyborgs-in-social-media?imm_mid=0f81cc&cmp=em-data-na-na-newsltr_20171115

[10] K. S. Perera, B. Neupane, M. A. Faisal, Z. Aung, and W. L. Woon, "A novel ensemble learning-based approach for click fraud detection in mobile advertising," in *Mining Intelligence and Knowledge Exploration*, R. Prasath and T. Kathirvalavakumar, Eds. Cham, Switzerland: Springer, 2013, pp. 370–382.

[11] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving Twitter spammers," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1280–1293, Aug. 2013.

[12] M. Taneja, K. Garg, A. Purwar, and S. Sharma, "Prediction of click frauds in mobile advertising," in *Proc. 8th Int. Conf. Contemp. Comput. (IC)*, 2015, pp. 162–166.

[13] M. Iqbal, M. Zulkernine, F. Jaafar, and Y. Gu, "Fcfraud: Fighting click-fraud from the user side," in *Proc. 17th Int. Symp. High Assurance Syst. Eng. (HASE)*, 2016, pp. 157–164.

[14] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW)*, Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 963–972

[15] X. Zhang and G. Dong, "Overview and analysis of contrast pattern based classification," in *Contrast Data Mining: Concepts, Algorithms, and Applications.* (Data Mining and Knowledge Discovery Series), G. Dong and J. Bailey, Eds. New York, NY, USA: CRC Press, 2012, ch. 11, pp. 151–170.

[16] G. Dong, "Preliminaries," in *Contrast Data Mining: Concepts, Algorithms, and Applications Data Mining and Knowledge Discovery Series*, G. Dong and J. Bailey, Eds. London, U.K.: Chapman & Hall, 2012, ch. 1, pp. 3–12.

[17] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen, "Comprehensible credit scoring models using rule extraction from support vector machines," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1466–1476, 2007.

[18] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases," *Neurocomputing*, vol. 175, pp. 935–947, Jan. 2016.

[19] J. Rodríguez, M. A. Medina-Pérez, A. E. Gutierrez-Rodríguez, R. Monroy, and H. Terashima-Marín, "Cluster validation using an ensemble of supervised classifiers," *Knowl.-Based Syst.*, vol. 145, pp. 134–144, Apr. 2018.

[20] L. J. González-Soler, L. Chang, J. Hernández-Palancar, A. Pérez-Suárez, and M. Gomez-Barrero, "Fingerprint presentation attack detection method based on a bag-of-words approach," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, M. Mendoza and S. Velastín, Eds. Cham, Switzerland: Springer, 2018, pp. 263–271.

[21] Y. Martínez-Díaz, H. Méndez-Vázquez, L. López-Avila, L. Chang, L. E. Sucar, and M. Tistarelli, "Toward more realistic face recognition evaluation protocols for the youtube faces database," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 526–5268.

[22] Y. Martínez-Díaz, N. Hernández, R. J. Biscay, L. Chang, H. Méndez-Vázquez, and L. E. Sucar, "On fisher vector encoding of binary features for video face recognition," *J. Vis. Commun. Image Represent.*, vol. 51, pp. 155–161, Feb. 2018.

[23] O. Loyola-González, M. A. Medina-Pérez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, R. Monroy, and M. García-Borroto, "PBC4cip: A new contrast pattern-based classifier for class imbalance problems," *Knowl.-Based Syst.*, vol. 115, pp. 100–109, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705116304002

[24] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of twitter accounts: Are You a Human, Bot, or Cyborg?" *IEEE Trans. Dependable Secure Computing*, vol. 9, no. 6, pp. 811–824, Nov. 2012.

[25] L. M. Aiello, M. Deplano, R. Schifanella, and G. Ruffo, "People are strange when you're a stranger: Impact and influence of bots on social networks," *Links*, vol. 697, no. 483, pp. 1–566, 2012.

[26] F. B. Keller, D. Schoch, S. Stier, and J. Yang, "How to manipulate social media: Analyzing political astroturfing using ground truth data from South Korea," in *Proc. ICWSM*, 2017, pp. 564–567.
[27] P. Suárez-Serrato, M. E. Roberts, C. Davis, and F. Menczer, "On the influence of social bots in online protests," in *Proc. Int. Conf. Social Informat.* Cham, Switzerland: Springer, 2016, pp. 269–278.
[28] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "Botornot: A system to evaluate social bots," in *Proc. 25th Int. Conf. Companion World Wide Web*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 273–274.
[29] B. Wang, A. Zubiaga, M. Liakata, and R. Procter. (2015). "Making the most of tweet-inherent features for social spam detection on twitter." [Online]. Available: https://arxiv.org/abs/1503.07405
[30] O. Loyola-González, A. López-Cuevas, M. A. Medina-Pérez, B. Camiña, J. E. Ramírez-Márquez, and R. Monroy, "Fusing pattern discovery and visual analytics approaches in tweet propagation," *Inf. Fusion*, vol. 46, pp. 91–101, Mar. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253517307716
[31] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots+ machine learning," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2010, pp. 435–442.
[32] A. H. Wang, "Detecting spam bots in online social networking sites: A machine learning approach," in *IFIP Annu. Conf. Data Appl. Secur. Privacy*. Berlin, Germany: Springer, 2010, pp. 335–342.
[33] F. Ahmed and M. Abulaish, "A generic statistical approach for spam detection in online social networks," *Comput. Commun.*, vol. 36, nos. 10–11, pp. 1120–1129, 2013.
[34] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278
[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[36] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM Trans. Knowl. Discovery Data (TKDD)*, vol. 8, no. 1, p. 2, 2014.
[37] Z. Gilani, E. Kochmar, and J. Crowcroft, "Classification of twitter accounts into automated agents and human users," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 489–496.
[38] J. P. Dickerson, V. Kagan, and V. S. Subrahmanian, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?" in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2014, pp. 620–627.
[39] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Dna-inspired online behavioral modeling and its application to spambot detection," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 58–64, Sep./Oct. 2016.
[40] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Inf. Sci.*, vol. 260, pp. 64–73, Mar. 2014.
[41] S. Van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 1, pp. 121–141, 2008.
[42] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Trans. Knowl. Discovery Data (TKDD)*, vol. 3, no. 3, p. 14, 2009.
[43] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++: A clustering algorithm for data streams," *J. Exp. Algorithmics*, vol. 17, pp. 2–4, Jul. 2012.
[44] N. Chavoshi, H. Hamooni, and A. Mueen, "Identifying correlated bots in twitter," in *Proc. Int. Conf. Social Informat.* Cham, Switzerland: Springer, 2016, pp. 14–21.
[45] B. Cervantes, R. Monroy, M. A. Medina-Pérez, M. Gonzalez-Mendoza, and J. Ramirez-Marquez, "Some features speak loud, but together they all speak louder: A study on the correlation between classification error and feature usage in decision-tree classification ensembles," *Eng. Appl. Artif. Intell.*, vol. 67, pp. 270–282, Jan. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197617302488
[46] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 1999, pp. 43–52.
[47] A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz, "From Local Patterns to Global Models: The LeGo Approach to Data Mining," in *Proc. Int. Workshop Local Patterns Global Models (LeGo)*, 2008, pp. 1–16.
[48] M. García-Borroto, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, "Finding the best diversity generation procedures for mining contrast patterns," *Expert Syst. Appl.*, vol. 42, no. 11, pp. 4859–4866, 2015.
[49] M. García-Borroto, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, M. A. Medina-Pérez, and J. Ruiz-Shulcloper, "LCMine: An efficient algorithm for mining discriminative regularities and its application in supervised classification," *Pattern Recognit.*, vol. 43, no. 9, pp. 3025–3034, 2010.
[50] M. A. Medina-Pérez, R. Monroy, J. B. Camiña, and M. García-Borroto, "Bagging-tpminer: A classifier ensemble for masquerader detection based on typical objects," *Soft Comput.*, vol. 21, no. 3, pp. 557–569, Feb. 2017.
[51] H. Fan and R. Kotagiri, "An efficient single-scan algorithm for mining essential jumping emerging patterns for classification," in *Proc. Adv. Knowl. Discovery Data Mining, 6th Pacific–Asia Conf. (PAKDD)*, M.-S. Chen, P. S. Yu, and B. Liu, Eds. Berlin, Germany: Springer, 2002, pp. 456–462.
[52] H. Fan and K. Ramamohanarao, "A Bayesian approach to use emerging patterns for classification," in *Proc. 14th Australas. Database Conf. (ADC)*, Darlinghurst, Australia: Australian Computer Society, 2003, pp. 39–48.
[53] L. Wang, H. Zhao, G. Dong, and J. Li, "On the complexity of finding emerging patterns," in *Proc. 28th Annu. Int. Comput. Softw. Appl. Conf.*, vol. 2, Sep. 2004, pp. 126–129.
[54] X. Zhang, G. Dong, and K. Ramamohanarao, "Information-based classification by aggregating emerging patterns," in *Intelligent Data Engineering and Automated Learning—IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents* (Lecture Notes in Computer Science), K. Leung, L. W. Chan, and H. Meng, Eds. Berlin, Germany: Springer, vol. 1983, 2000, pp. 48–53.
[55] G. Dong, X. Zhang, L. Wong, and J. Li, "Caep: Classification by aggregating emerging patterns," in *Proc. 2nd Int. Conf. Discovery Sci. (DS)*, S. Arikawa and K. Furukawa, Eds. Berlin, Germany: Springer, 1999, pp. 30–42.
[56] D. Yuxin, Y. Xuebing, Z. Di, D. Li, and A. Zhanchao, "Feature representation and selection in malicious code detection methods based on static system calls," *Comput. Secur.*, vol. 30, no. 6, pp. 514–524, 2011.
[57] D. Cieslak, T. Hoens, N. Chawla, and W. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," *Data Mining Knowl. Discovery*, vol. 24, no. 1, pp. 136–158, 2012.
[58] F. J. Aherne, N. A. Thacker, and P. I. Rockett, "The bhattacharyya metric as an absolute similarity measure for frequency coded data," *Kybernetika*, vol. 34, no. 4, pp. 363–368, 1998.
[59] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Burlington, MA, USA: Morgan Kaufmann, 1993.
[60] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth, 1984.
[61] S. Rodda and S. Mogalla, "A normalized measure for estimating classification rules for multi-class imbalanced datasets," *Int. J. Eng. Sci. Technol.*, vol. 3, no. 4, pp. 3216–3220, 2011.
[62] H. Fan and K. Ramamohanarao, "Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 6, pp. 721–737, Jun. 2006.
[63] W. Buntine, "Theory refinement on Bayesian networks," in *Proc. 7th Conf. Uncertainty Artif. Intell. (UAI)*, San Francisco, CA, USA: Morgan Kaufmann, 1991, pp. 52–60.
[64] F. Pernkopf, "Bayesian network classifiers versus selective k-NN classifier," *Pattern Recognit.*, vol. 38, no. 1, pp. 1–10, 2005.
[65] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.
[66] J. G. Moreno-Torres, J. A. Saez, and F. Herrera, "Study on the impact of partition-induced dataset shift on k-fold cross-validation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1304–1312, Aug. 2012.
[67] Y. Li and X. Zhang, "Improving k Nearest Neighbor with Exemplar Generalization for Imbalanced Classification," in *Proc. 15th Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining (PAKD)*, J. Z. Huang, L. Cao, and J. Srivastava, Eds. Berlin, Germany: Springer, 2011, pp. 321–332.
[68] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell. (UAI)*, San Francisco, CA, USA, 1995, pp. 338–345. [Online]. Available: http://dl.acm.org/citation.cfm?id=2074158.2074196
[69] Z.-Q. Zeng, H.-B. Yu, H.-R. Xu, Y.-Q. Xie, and J. Gao, "Fast training support vector machines using parallel sequential minimal optimization," in *Proc. 3rd Int. Conf. Intell. Syst. Knowl. Eng.*, Nov. 2008, pp. 997–1001. doi: 10.1109/ISKE.2008.4731075.

[70] S. Le Cessie and J. C. Van Houwelingen, "Ridge estimators in logistic regression," *Appl. Statist.*, vol. 41, no. 1, pp. 191–201, 1992. [Online]. Available: http://www.jstor.org/stable/2347628
[71] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
[72] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[73] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Beijing, China: Tsinghua Univ. Press, 2001.
[74] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.
[75] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[76] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[77] J. T. Roscoe, *Fundamental Research Statistics for the Behavioral Sciences* (International Series in decision processes), 2nd ed. New York, NY, USA: Holt, Rinehart & Winston, 1975.
[78] T. C. Krehbiel, "Correlation coefficient rule of thumb," *Decision Sciences J. Innovative Edu.*, vol. 2, no. 1, pp. 97–100, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.0011-7315.2004.00025.x
[79] S. García and F. Herrera, "An Extension on 'Statistical Comparisons of Classifiers over Multiple Data Sets' for all Pairwise Comparisons," *J. Mach. Learn. Res.*, vol. 9, nos. 2677–2694, p. 66, 2008.

OCTAVIO LOYOLA-GONZÁLEZ received the B.Eng. degree in informatics engineering and the M.Sc. degree in applied informatics from the University of Ciego de Ávila, Cuba, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from the National Institute of Astrophysics, Optics and Electronics, Mexico, in 2017. He is currently a Research Professor with the Tecnologico de Monterrey, Puebla Campus, where he is also a member of the GIEE-ML (Machine Learning) Research Group. He is also a member of the Mexican Researchers System (Rank 1). He has been involved in many research projects about pattern recognition, which have been applied on biotechnology and dactyloscopy problems. His research interests include contrast pattern-based classification, data mining, one-class classification, masquerader detection, fingerprint recognition, and palmprint recognition.

RAÚL MONROY received the Ph.D. degree in artificial intelligence from Edinburgh University, in 1998, under the supervision of Prof. A. Bundy. He is currently a Full Professor in computing with the Tecnologico de Monterrey, and also leads GIEE-ML, a group of computer scientists interested in machine learning models. Since 1998, he has been a member of the CONACyT's National Research System, currently rank 3, a Fellow of the National Academy of Sciences, and a Founding Fellow of the Mexican Academy of Computing. His research interests include the discovery and application of machine learning techniques for anomaly detection, robot motion planning, and the uncovering and then correction of errors in either a system or its specification.

JORGE RODRÍGUEZ received the M.Sc. degree in computer science and the Ph.D. degree in engineering sciences from the Tecnologico de Monterrey, in 2013 and 2017, respectively. He was with HP Labs, Palo Alto, CA, USA, on Machine Learning and Cloud Computing Projects. He currently holds a postdoctoral position with GIEE-ML, Tecnologico de Monterrey, working on latent palmprint and fingerprint identification, and a member of the Mexican National Researchers System (SNI) level C. His research interests include the application of machine learning, and artificial intelligence for physical and information security.

ARMANDO LÓPEZ-CUEVAS graduated in electronic engineering from the Autonomous University of Nayarit, in 2005, received the M.Sc. degree in electrical engineering from the Center for Research and Advanced Studies, CINVESTAV in collaboration with the National Institute for Nuclear Research, in 2009, and the Ph.D. degree from CINVESTAV Guadalajara, in 2014. He has conducted research stays at different institutes and currently holds a position at ITESM. His current research interests include natural language processing and machine learning.

JAVIER ISRAEL MATA-SÁNCHEZ received the B.S. degree in music production and engineering from the Tecnologico de Monterrey, Monterrey Campus, in 2017, where he is currently pursuing the master's degree in computer science. His thesis is on regards bot detection and one class classifiers. His research interests include supervised classification, one-class classifiers, bot detection, and data science.