SpADe: Multi-Stage Spam Account Detection for Online Social Networks

Federico Concone, Giuseppe Lo Re, *Senior Member, IEEE*, Marco Morana, and Sajal K. Das, *Fellow, IEEE*

Abstract—In recent years, Online Social Networks (OSNs) have radically changed the way people communicate. The most widely used platforms, such as Facebook, Youtube, and Instagram, claim more than one billion monthly active users each. Beyond these, news-oriented micro-blogging services, e.g., Twitter, are daily accessed by more than 120 million users sharing contents from all over the world. Unfortunately, legitimate users of the OSNs are mixed with malicious ones, which are interested in spreading unwanted, misleading, harmful, or discriminatory content. Spam detection in OSNs is generally approached by considering the characteristics of the account under analysis, its connection with the rest of the network, as well as data and metadata representing the content shared. However, obtaining all this information can be computationally expensive, or even unfeasible, on massive networks. Driven by these motivations, in this article we propose SpADe, a multi-stage Spam Account Detection algorithm with reject option, whose purpose is to exploit less costly features at the early stages, while progressively extracting more complex information only for those accounts that are difficult to classify. Experimental evaluation shows the effectiveness of the proposed algorithm compared to single-stage approaches, which are much more complex in terms of features processing and classification time.

Index Terms—Social network security, spam detection, artificial intelligence

1 Introduction

The widespread diffusion of Online Social Networks (OSNs) has enabled new forms of communication that allow people to regularly share almost any kind of information within a virtual community. Nowadays, a number of OSNs are available to address the needs of different types of users, providing them with a variety of services and objectives. Some aim to create networks of people who know each other (e.g., Facebook), or to connect people interested in news coming from all over the world (e.g., Twitter); others are oriented to professional networking (e.g., LinkedIn), some offer instant messaging services, such as WhatsApp, Telegram, or Viber, while some others are mainly intended for sharing multimedia contents, e.g., Instagram and Youtube.

Thanks to their ease of use, popular OSNs claim billions of active users, most of which are unfortunately not aware of the threats coming from the cyber space. This represents the main reason why malicious users are attracted to the social networks as much as, or even more than, legitimate ones.

Research on social network security covers a wide number of topics, from *account hijacking*, *fraud* and *impersonation attacks* to *malware distribution* [1]. Beside them, spam detection is a well-known, and still open, challenge which affects social networks as well as any other type of network-based application [2].

In general terms, spammers are entities (real users or automated software agents) that repeatedly send unsolicited messages for various purposes, e.g., supporting commercial, slandering, or proselytizing campaigns [3]. Even though several spam detection techniques have been proposed in the literature, the art of spamming continuously evolves and new intelligent approaches for identifying spammers are constantly needed. The behavior of early social bots, for instance, was quite simplistic as they were just intended to spread messages to as many users as possible. As soon as the spam detection algorithms became able to identify the typical characteristics of these bots, such as the presence of a biased following/followers ratio (FF) as compared to real users, the attackers quickly improved their strategy [4]. A trustworthy FF value, for instance, could be easily forged by relying on groups of social bots which cooperate to mimic the interactions among normal OSN users, thus avoiding the corresponding countermeasures [5].

As a consequence, spam analysis in online social networks is generally approached by considering different levels of information that describe the user as a whole. To this aim, a variety of features and classification algorithms exist. Whereas the latter are typically borrowed from those adopted in other machine learning contexts, the feature extraction process is strictly dependent on the set of information that the OSN makes available. This commonly includes the characteristics of the account, its connection with the rest of the social network, as well as data and metadata representing the content shared. However, what is never considered in the existing works is the effort required

- *Federico Concone, Giuseppe Lo Re, and Marco Morana are with the Department of Engineering, University of Palermo, 90128 Palermo, Italy. E-mail: {federico.concone, giuseppe.lore, marco.morana}@unipa.it.*
- *Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA. E-mail: sdas@mst.edu.*

Manuscript received 10 September 2021; revised 28 July 2022; accepted 10 August 2022. Date of publication 16 August 2022; date of current version 11 July 2023.
The work of Sajal K. Das was supported by the NSF under Grants SaTC-2030624 and OAC-2104078.
(Corresponding author: Federico Concone.)
Digital Object Identifier no. 10.1109/TDSC.2022.3198830

TABLE 1
Comparison Between SpADe and Other Relevant Spam Detection Approaches

	SpADe	[7]	[8]	[12]	[13]	[17]	[24]	[26]	[28]	[29]	[30]
Honey-profiles		✓	✓								
URLs blacklists				✓	✓						
Wide feature sets and ML	✓					✓	✓	✓			
Deep-Learning											✓
Other learning methods									✓	✓	
Multi-stage approach	✓										
Number of features	39	10	10	14	15	18	107	12	12	–	9
Effort in detection	Low	High	High	Medium	Medium	High	High	High	Medium	Low	High
Assessment on *private* dataset	✓	✓	✓	✓	✓	✓	✓		✓	✓	
Assessment on *public* dataset	✓							✓			✓
Assessment on multiple datasets	✓							✓		✓	✓

to extract each feature, which deeply impacts on the capability of the classifier to provide timely results.

Driven by these motivations, in this paper we propose SpADe, a multi-stage Spam Account Detection technique with reject option, whose purpose is to exploit less costly features at the early stages, while progressively extracting more complex information only for those accounts that are more challenging to classify.

SpADe consists of four stages of analysis that progressively combine information about (i) the general characteristics of the account, (ii) the URLs shared, (iii) the similarity of the published contents, and (iv) the relationship between the user and the rest of the social network. Bayes classifiers are adopted to implement the accept/reject mechanism in all the stages except the last one, in which decision trees are exploited to make a decision regardless of the uncertainty degree. The effectiveness of the approach was proven by considering as case study the most widely diffused microblogging platform, i.e., Twitter. Nevertheless, it is worth noting that the features we have chosen, as well as the classification algorithm, are reasonable for any other OSN.

The major contributions of this work are summarized as follows.

- The paper presents SpADe, a novel Spam Account Detection approach that takes into account the effectiveness of each feature as well as its observation (collection/processing) cost; at the best of our knowledge, this is the first work in which the two aspects are considered together.
- The most representative features presented in the literature were selected and organized into four consistent categories, each of which captures a different facet of spamming behaviors and is characterized by a homogeneous observation cost.
- A novel multi-stage classification algorithm with reject option that incrementally exploits set of features of increasing complexity was designed. This allows to classify an account as soon as the chosen confidence level is reached, without the need to capture the whole feature set for every account under analysis.
- SpADe is evaluated both on a dataset of about 40.000 users we retrieved from the Twitter stream during the last year, and on a popular reference public dataset of about 11.000 users collected in 2017. Comparing the results obtained on datasets of different sizes, and acquired in different epochs, made it possible to carry out a robust evaluation of the proposed method.

The remainder of the paper is organized as follows: related works are outlined in Section 2. The mathematical background of the proposed multi-stage classification algorithm is provided in Section 3. Section 4 presents SpADe and the features exploited at each stage by highlighting their role in spam detection. Experimental settings and results are discussed in Section 5. Conclusions follow in Section 6.

2 Related Work

Spam detection is a popular research topic that has been widely addressed in the last decades. More recently, the focus has shifted towards the detection of spam campaigns on Online Social Networks (OSNs), which represent one of the most fertile grounds for this type of cyber threat [6]. The main reason is that users of OSNs can share information in many different ways, and so do spammers, making their behavior difficult to predict. In this paper, we focus on Twitter analysis because *tweets* generally refer to popular events and are therefore characterized by a high information content.

This section presents a review of the state of the art by following the evolution of spam detection systems. Related works are arranged in categories, which reflect the most important characteristics of the studies discussed.

Honey-Profiles. Early spam detection was mainly based on statistical analysis of the account activities. This type of systems required a protected environment in which the spammer could act undisturbed, allowing the detection algorithm to monitor and learn its behavior. In the Social Honeypot Project [7], for instance, an automated bot is assigned to every account to be analyzed in order to capture meaningful features that may reveal a malicious activity. The authors of [8] exploited 60 Twitter bots as honeypots to attract a total of 36,000 accounts, which were analyzed by observing their activities and relationships with their neighborhood. In [9], an extensive study on how spammers operate to target Facebook, Twitter and MySpace is presented. In order to observe four categories of spammers, called *displayers*, *braggers*, *posters* and *whisperers*, large sets of honey profiles were created with the aim of capturing information about the accounts they are connected with, and the messages they received. Then, users were classified by Random Forest exploiting

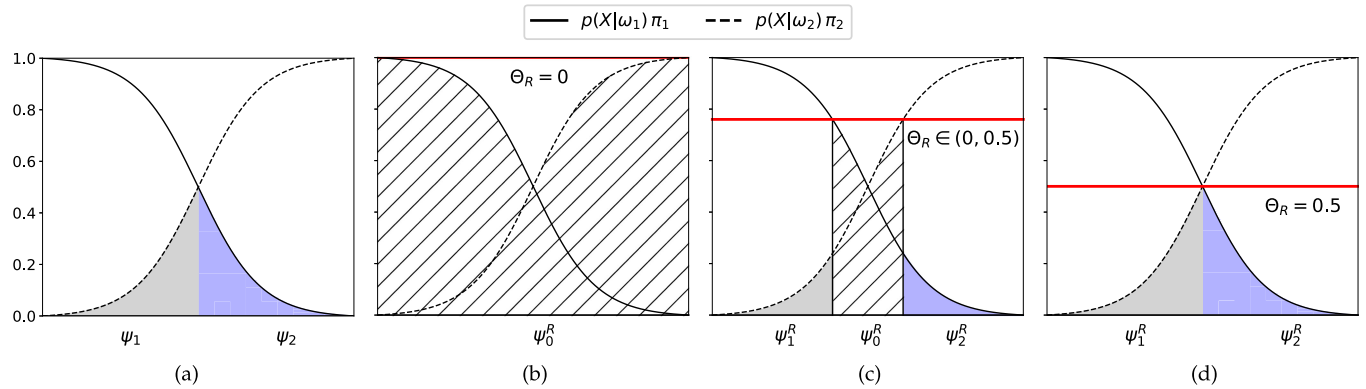

Fig. 1. Bayes decision rules and the corresponding errors for a two-class classification problem (a) without reject, (b) with reject option and threshold $\Theta_R = 0$, (c) with Θ_R in (0,0.5), and (d) with $\Theta_R = 0.5$.

conventional features, such as following/follower ratios, URLs, message similarity, account activity and quality of the neighborhood. Such an analysis also revealed the possibility of identifying spam campaigns [10] in which several bots cooperate with the same goal. The use of social honeypots is also discussed in [11], which also highlights how these traps can be effective in identifying previously unknown spamming patterns. The major limitation of these solutions is that several honeypots must be implemented in order to make the approaches effective. However, when dealing with large communities of spammers, this turns out to be computationally expensive, or even unfeasible.

URLs Blacklists. URLs are frequently injected by the spammers into trending topics and related messages. WarningBird [12] aimed at detecting spammers by following the URLs through all their redirections so as to obtain the target IP addresses; then, a set of features is computed and analyzed in order to assign the *suspicious* label to the corresponding URLs. Results show the good performance of the system; nevertheless, the effectiveness of WarningBird dramatically drops when a obfuscation mechanism is applied to the URLs, e.g., via the URL shortening services. The authors of [13] extended the analysis of URLs by considering also how the links are received by the community, i.e., counting the actual number of clicks. However, the analysis is limited to a few different shortening services and the correlation between URLs and other type features is not considered.

Wide Feature Sets and ML. In order to identify the distinctive characteristics of a broader set of spamming strategies, several works proposed a variety of features [3], [14] that can be exploited as the basis for Machine Learning (ML) models. The system presented in [15], for instance, leverages on characteristics that are able to capture the way tweets are written, as well as the user's posting frequency, social interactions, and influence on the Twitter network. These features are exploited to train a Support Vector Machine (SVM) classifier capable of correctly identifying 70% of spammers and 96% of non-spammers. Even though these results are notable, the method lacks of considering other relevant aspects that are typical of spammers. In [16], elements such as the behavioral and content entropy, bait-techniques, and profile vectors are considered. The corresponding features were used to train four different supervised learning algorithms, namely Decision Tree, Random Forest, Bayes Networks, and Decorate. Results indicate that such a feature set allows to achieve good performance with any of the four algorithms. A different kind of features aimed to model the interactions between users and their followers is exploited in [17]. The idea is that spammers can easily alter features regarding their own behavior, while those based on to their relationships with the community are more difficult to change. Nevertheless, complex attacks based on *sybil* networks [18] might seriously reduce the effectiveness of this kind of features. Sybil account detection is addressed in [19], where a method called Ianus is proposed to discover fake accounts according to registration information. The study moves from the observation that sybil accounts are characterized by different registration patterns than legitimate ones. Then, sybil detection is solved as a graph inference problem in which registrations are modeled as nodes, and strongly connected nodes are more likely to represent sybils. Another approach to deal with compromised accounts is discussed in [20]. Malicious changes are distinguished from legitimate ones through statistical analysis and anomaly detection techniques. The system, called COMPA, exploits features capable of capturing recurring temporal patterns in the account usage, information about the messages (e.g., language, topic, the application used to share them), as well as the presence of URLs/mentions and the connections of the user with the social graph. Social graphs for spammer detection are also examined in [21], where Graph Convolutional Networks (GCNs) and Markov Random Fields (MRFs) are combined to detect neighbor message-passing and capture human insights in user following relations. The analysis of communities, and in particular of the topics that spread through them, can make it possible to identify groups of accounts with abnormal behaviors. POISED [22] is a system that models the different propagation paths of benign and malicious messages in order to distinguish between legitimate and spam accounts. Experimental evaluation performed on Twitter data shows the effectiveness of this approach, even against *poisoning* and *evasion* adversarial attacks. The detection of anomalous topics is addressed in [23], where a topology-based method to detect cooperative and organized spammer groups in micro-blogging communities is proposed. An anomaly detection problem is also formulated in [24], where spammers are described by means of 107 features. This system combines two data stream clustering [25] algorithms, namely StreamKM++ and DenStream, that allow to correctly identify the most of the spammers, with low

TABLE 2
The Set of Features Aimed to Describe the General Characteristics of the Account, the URLs and the Contents Shared by the User, as Well as the Properties of its Neighborhood

	ID	Meaning	Reference
Account	f_1	Follower Count	[15], [24], [39]
	f_2	Friend Count	[15], [24], [39]
	f_3	Tweet Count	[24]
	f_4	Account Age	[14], [15], [24]
	f_5	Favorites Count	[24]
	f_6	List Count	[39]
	f_7	Hashtag Count in description	[24], [39]
URL	f_8	URL Ratio	[15], [17]
	f_9	Unique URL Ratio	[17]
	f_{10}	Average Tweets with URLs	[15], [39]
	f_{11}	Duplicate URLs Ratio	[14], [16]
	f_{12}	Spam URL Ratio	[40]
	f_{13}	Same Redirection URL Ratio	[16], [40]
	f_{14}	Blacklisted URL Ratio	[40]
Content	f_{15}	Hashtag Ratio	[15], [17]
	f_{16}	Mention Ratio	[14], [17]
	f_{17}	Unique Mention Ratio	[16], [17]
	f_{18}	Retweet Ratio	[14], [17], [39]
	f_{19}	Reply Statuses Ratio	[15]
	f_{20}	Spam Tweet Ratio	[14]
	f_{21}	Automated tweet ratio/Tweet sources	[16]
	f_{22}	Average of length of tweets	[14], [39]
	f_{23}	Number of unique mention	[16]
	f_{24}	Variance in Tweet Intervals (VaTi)	[16], [17]
	f_{25}	Variance in n. of tweets per time unit (VaTw)	[16], [17]
	f_{26}	Ratio of VaTi and VaTw (TiTw)	[16]
	f_{27}	Number of tweets per day	[14]
	f_{28}	Max time between consecutive tweets	[14]
	f_{29}	Mean time between consecutive tweets	[14]
	f_{30}	Near-duplicates	[16]
Neighborhood	f_{31}	Following to Followers Ratio	[16]
	f_{32}	Follower to Following Ratio	[14], [39]
	f_{33}	Follower ratio	[14], [17], [24]
	f_{34}	Mean Followers Following to Follower Ratio	[17]
	f_{35}	Reputation	[17]
	f_{36}	Clustering Coefficient	[17]
	f_{37}	Follower based Reputation	[17]
	f_{38}	Community-based Reputation	[17]
	f_{39}	Community-based Clustering Coefficient	[17]

percentage of false positives. Data stream clustering is also discussed in [26], where a modified version of DenStream based on a set of incremental Bayes classifiers is presented. In this case, the feature set is designed so as to capture relevant characteristics of both the user's behavior and the tweet content. A different approach is presented in [27], where the acceptance of a user from other members of the community is considered as an indicator of its reliability. In particular, the authors propose an unsupervised spam detection algorithm in which high *peer acceptability* values are assigned to users that have common interests, e.g., users discussing the same topics and/or sharing the same contents.

Deep Learning. Other works proposed the use of Deep Learning (DL) because of the lower effort required for feature extraction [31]. In [30], a novel DL technique is showed to outperform two machine-learning classifiers. The DL approach considers only users' tweets and needs Google's Word2vec algorithm to learn tweet syntax, while ML algorithms exploit 9 easy to extract user/account's meta-data and text-based features. Word2vec is also used in [32] to convert tweets into dense vectors, which are analyzed by means of Recurrent Neural Networks (RNNs). The limitations of this approach are common to all deep learning based techniques [33], i.e., the reasons for a certain output are difficult to understand and there is no standard theory to guide in the selection of the right DL strategy.

TABLE 3
Classifiers Employed to Analyze the Features Listed in Table 2, Namely, Random Forest (RF), Decision Tree (DT), Support-Vector Machine (SVM), Bayesian Network (BN), k-Nearest Neighbors (k-NN), and Other

Reference	RF	DT	SVM	BN	k-NN	Other
[17]	✓	✓		✓		
[24]						✓
[14]	✓	✓	✓	✓	✓	✓
[15]			✓			
[16]	✓	✓		✓		✓
[39]	✓		✓	✓	✓	✓
[40]						✓

Other Learning Methods. Approaches based on pure ML suffer from the constant evolution of the spammers, which continuously worsens the performance of existing methods. Incremental learning aims to keep the models up to date in order to deal with new attack strategies. This aspect is deeply analyzed in [28], where a model, called Lfun (Learning from unlabeled tweets), is proposed to include new unlabeled spam tweets into the classifier training process. On the same principle operates AdaGraph [34], an unsupervised graph-based technique that dynamically builds and updates a graph of behaviors to detect spam in OSNs. Although the performance of this approach are quite relevant, it cannot be adopted for massive graphs analysis because of the cost of collecting community-based features. Social FingerPrinting [29] combines supervised and unsupervised techniques in order to identify two different types of automated spammers, namely, those interested in advertising products on e-commerce platforms and promoting a political candidate during the electoral campaign. The behavior of each account is encoded as a sequence of characters that represents a sort of *digital DNA*; then, a similarity measure between DNA sequences is used to detect genuine or spamming accounts.

Analysis of the literature reveals that spammers' behavior can be modeled through a variety of feature sets, capable of capturing the essence of a tweet, the characteristics of the account, as well as the interaction between users in the network. However, all the works described so far do not explicitly consider the cost of obtaining the features, which in many cases is prohibitive and can significantly affect the classifier's ability to provide timely results. For this reason, our approach aims to progressively exploit features of increasing complexity, depending on the the peculiarities of each spammer. The idea is somehow similar to the process of diagnosing a clinical condition through a series of investigations of increasing cost and complexity [35].

The characteristics of SpADe are summarized in Table 1, which also provides a comparison with some of the works discussed in this section.

3 Decision Under Uncertainty

Bayesian decision theory assumes that decisions are taken according to the probability of a possible outcome. Given

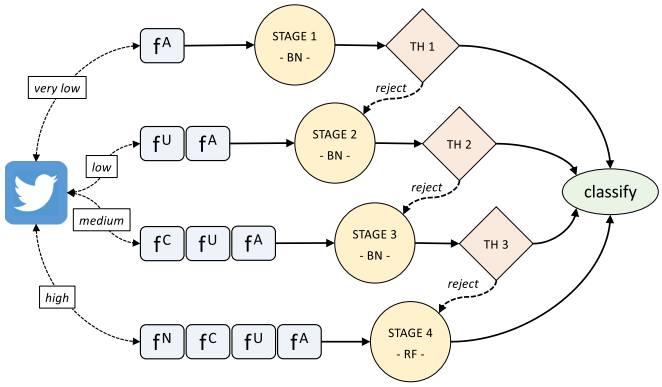

Fig. 2. Overview of the features and algorithms used in the four stages of analysis that characterize SpADe.

the problem of associating an observation X with a class from the finite set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_Y\}$, a generic decision rule would suggest to choose the class that minimizes the classification error, i.e., the one whose *posterior* probability given X is the greatest

$$p(\omega_y|X) > p(\omega_c|X), \quad c = 1, \ldots, Y \quad c \neq y, \tag{1}$$

or equivalently, in terms of *class-conditional* and *prior* probabilities

$$p(X|\omega_y)\pi_y > p(X|\omega_c)\pi_c, \quad c = 1, \ldots, Y \quad c \neq y. \tag{2}$$

Such a decision process can also be seen as splitting the observation space into a set of regions, $\Psi = \{\psi_1, \ldots, \psi_Y\}$, such that if $X \in \psi_y$ then X is associated with the class ω_y.

Because of their probabilistic nature, Bayes decision rules are not free from errors; hence, assuming that some classification errors are more costly than others, it is reasonable to associate to each decision d^i, $i \in \{1, \ldots, Y\}$, a *loss function* $\lambda_{i,y}$ that quantifies the penalty of classifying as ω_i when the actual class is ω_y. The *zero-one loss function*, for instance, assigns no loss to a correct decision, and a unit loss to any error

$$\lambda_{i,y} = \begin{cases} 0 & \text{if } i = y, \\ 1 & \text{if } i \neq y. \end{cases} \tag{3}$$

Given the loss function $\lambda_{i,y}$, the *conditional risk* associated with the ith decision is defined as

$$R(d^i) = \sum_{y=1}^{Y} \lambda_{i,y}\, p(\omega_y|X). \tag{4}$$

Considering a simple scenario in which only two classes exist, i.e., $\Omega = \{\omega_1, \omega_2\}$, and $D = \{d^1, d^2\}$, the decision rule should suggest to classify X as ω_1 if $R(d^1) < R(d^2)$, as ω_2 if $R(d^2) < R(d^1)$, whereas the choice would be arbitrary if the two risks are equal. Choosing the lowest conditional risk allows to minimize the *overall risk* R_T, which is defined as

$$R_T = \int_{\psi_i} R(d^i)\, p(X)\, dX = \sum_{y=1}^{Y} \lambda_{i,y} \int_{\psi_i} p(X|\omega_y)\pi_y\, dX. \tag{5}$$

TABLE 4
Datasets Used in the Experimental Assessment

	1KS-10KN [48]	our-dataset
Collection period	April to July 2010	June to December 2020
Number of accounts	11.000	40.000
Number of tweets	1.000.000	8.000.000
Features on account	✓	✓
Features on URL	✓	✓
Features on content	✓	✓
Features on neighbors	✓	✓

Based on Eq. (3), such a risk can be rewritten as

$$\begin{aligned} R_T &= \sum_{y=1, y\neq i}^{Y} \lambda_{i,y} \int_{\psi_i} p(X|\omega_y)\pi_y\, dX \\ &= 1 - \int_{\psi_i} p(X|\omega_i)\pi_i\, dX = p(error|X), \end{aligned} \tag{6}$$

that corresponds to the *average probability error*.

3.1 Classification With Reject

Unfortunately, the decision rules from Eqs. (1) and (2) do not directly consider the conditional risk of a wrong decision, then they always allow to classify an input, whatever the classification error is. This can sometimes lead to an excessive misclassification rate; for this reason, if the risk is too high, the possibility of refusing to decide is introduced. In this case, given Eq. (6), the decision to *classify* or *reject* can be made according to a threshold $\Theta_R \in [0, 1]$ on the overall risk R_T [36]:

$$d^i = \begin{cases} classify & \text{if } \max\{p(X|\omega_y)\pi_y\} \geq 1 - \Theta_R, \\ reject & \text{if } \max\{p(X|\omega_y)\pi_y\} < 1 - \Theta_R. \end{cases} \tag{7}$$

Rejection, denoted by d^0, provides an extra choice within the decision space $D = \{d^0, d^1, \ldots, d^Y\}$, and corresponds to define a new region ψ_0^R within the observation space, i.e., $\Psi^R = \{\psi_0^R, \psi_1^R, \ldots, \psi_Y^R\}$. When the reject option is considered, the loss function is also redefined as

$$\lambda_{i,y} = \begin{cases} 0 & \text{if } i = y, \\ 1 & \text{if } i \neq y, i \neq 0, \\ l_c & \text{if } i = 0 \text{ (reject)}, \end{cases} \tag{8}$$

where $l_c \in (0, 1)$ is the value of the loss cost, i.e., the penalty occurring when the decision d^0 is made [35].

The advantages gained from rejecting are demonstrated by the following theorem.

Lemma 1. *Given a classification problem with Y classes, if the reject threshold $\Theta_R \geq 1 - \frac{1}{Y}$, then the decision to reject is never made.*

Proof. Since the probabilities of all the Y outcomes sum to 1

$$\sum_{y=1}^{Y} p(X|\omega_y)\pi_y = 1, \tag{9}$$

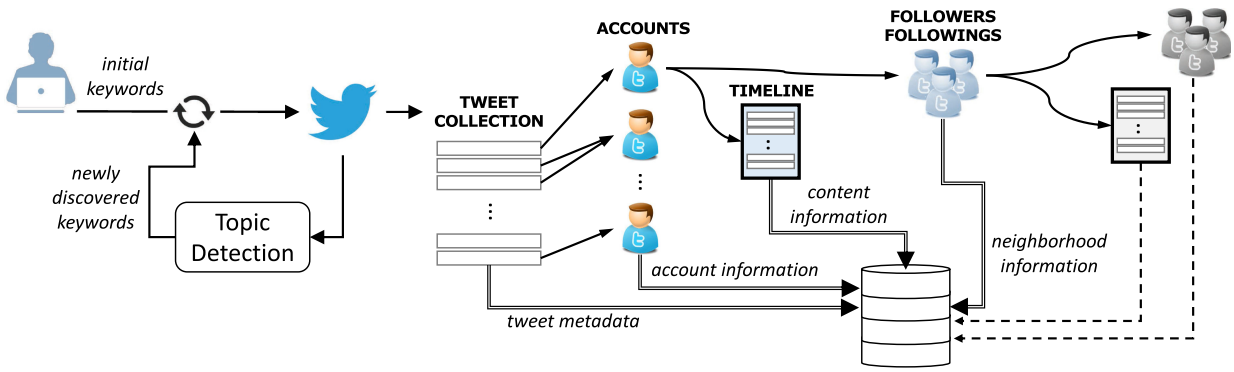

Fig. 3. Data collection procedure. A initial set of static keywords is used to query the Twitter stream; then, topic detection is performed to find out new terms emerging from the topics and to update the queries. Tweets are analyzed to retrieve the corresponding metadata and authors. For each author, the timeline and the neighbors are explored. Then, the procedure is repeated for follower and following accounts extracted in the previous step.

the highest joint probability cannot be lower than $1/Y$

$$\max\{p(X|\omega_y)\pi_y\} \geq \frac{1}{Y}. \tag{10}$$

As a consequence, the reject condition defined in Eq. (7)

$$\max\{p(X|\omega_y)\pi_y\} < 1 - \Theta_R, \tag{11}$$

is verified for any $\Theta_R < 1 - \frac{1}{Y}$, i.e., $\Theta_R \in [0, 1 - \frac{1}{Y})$. □

Theorem 2. *Given $0 \leq \Theta_R < 1 - \frac{1}{Y}$, the average probability error of a classifier that includes the reject option, $p(error^R|X)$, is not higher than the error $p(error|X)$ made by a classifier in which the reject option is not available*

$$p(error^R|X) \leq p(error|X).$$

Proof. The proof is given for a binary classification problem and can be easily extended to a multi-class scenario. Each region within the observation space can be defined as

$$\psi_i = \{X : p(X|\omega_i)\pi_i > p(X|\omega_j)\pi_j \ \forall i \neq j\}. \tag{12}$$

According to Eq. (7), the reject option impacts on the observation space by redefining the regions as

$$\begin{aligned} \psi_i^R &= \psi_i \cap \{X : p(X|\omega_i)\pi_i \geq 1 - \Theta_R\}, \\ \psi_0^R &= \psi_i \cap \{X : p(X|\omega_i)\pi_i < 1 - \Theta_R\}. \end{aligned} \tag{13}$$

Following Eq. (6), the average probability error in a system with reject depends on these regions and can be expressed as

$$p(error^R|X) = \int_{\psi_i^R} p(X|\omega_j)\pi_j \, dX. \tag{14}$$

Then, the difference between $p(error|X)$ and $p(error^R|X)$ is

$$\begin{aligned} \Delta &= \int_{\psi_i} p(X|\omega_j)\pi_j \, dX - \int_{\psi_i^R} p(X|\omega_j)\pi_j \, dX \\ &= \int_{\psi_i - \psi_i^R} p(X|\omega_j)\pi_j \, dX. \end{aligned} \tag{15}$$

Now, let us evaluate the relationship between Δ and Θ_R.

Case $\Theta_R = 0$: according to Eq. (7), a zero threshold causes any observation X to be rejected; thus, $\psi_i^R = \emptyset \ \forall i > 0$, and the integral over ψ_i results in $\Delta = p(error|X)$.

Case $\Theta_R = 0.5$: according to Lemma 1, if $\Theta_R \geq 0.5$ the rejection is never performed. Then, $\psi_i = \psi_i^R \ \forall i > 0$, and $\Delta = 0$.

Case $0 < \Theta_R < 0.5$: rejection can be chosen and $\psi_0^R = \psi_i - \psi_i^R$. As a consequence, Δ depends on ψ_0^R, that is the reduction of the error is proportional to the size of the reject region. □

The outcomes of the properties demonstrated so far can be also observed by comparing the plots in Fig. 1, in which the easiest case of a two-class problem is illustrated for the sake of clarity.

The first plot (Fig. 1a) refers to the classifier without reject; here, two decision regions exist, namely ψ_1 and ψ_2, and the classification errors for the classes ω_1 and ω_2 correspond to the violet and grey areas respectively, as defined by Eq. (6). The other three plots show the effect of introducing the reject option. For instance, if a threshold $\Theta_R = 0$ is considered (Fig. 1b), every observation is rejected; as a consequence, since the classification is not performed, all the errors are null and only the reject region ψ_0^R exists. As the threshold value increases (Fig. 1c), the errors are reduced by an amount that depends on the size of the reject region. However, when $\Theta_R = 0.5$, the region ψ_0^R is null (Fig. 1d) and the errors, due to the wrong classification of ω_1 and ω_2, are the same as Fig. 1a. The same holds for any value $\Theta_R > 0.5$, as proved in Lemma 1.

3.2 Multi-Stage Classification With Reject

Given that proper threshold values are demonstrated to reduce the classification error, a new problem has to be faced: how to deal with the observations that are rejected.

A multi-stage classifier can be designed to address this issue by introducing s stages, $s \in \{1, \ldots, S\}$, each of which applies the Bayes decision rule to a partial observation vector $x_s \subseteq X$. As one would expect, the decision at the stage s must take into account the reject decisions made at the previous stages. Such a sequence of decisions can be seen as a first-order Markov chain [37], where the decision at the stage s is dependent only on the stage $s-1$. Thus, starting from Eq. (4), the conditional risk for the multi-stage classifier is defined as

$$R(d_s^i) = \sum_{y=1}^{Y} \lambda_{i,y} \, p(\omega_y|x_s) \, R(d_{s-1}^0), \tag{16}$$

where $R(d^0_{s-1})$ is the conditional risk of the previous stage, and $\lambda_{i,y}$ is the loss defined in Eq. (8).

In the scenario addressed here, such a *loss* strictly depends on the cost ϕ of making an observation x_s. As a consequence, the multi-stage loss function can be obtained from Eq. (8) as

$$\lambda^s_{i,y} = \begin{cases} 0 & \text{if } i = y, \\ 1 & \text{if } i \neq y, i \neq 0, \\ \phi(x_{s+1}) & \text{if } i = 0 \text{ (reject)}. \end{cases} \quad (17)$$

Such a multi-stage classification process brings advantages in terms of recognition performance because a decision is made only when the inputs are certain enough, and moving to the next stage is too costly. Moreover, an optimal multi-stage classifier will exhibit the following property [38]: $\phi(x_{s-1}) \leq \phi(x_s) \leq \phi(x_{s+1})$.

4 SpADe Overview

A common assumption of machine learning models is that a collection D of heterogeneous data can be described by a finite set of features $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$. The cost of *Observing* each feature value f_n depends on two quantities, namely, the time required to *Collect* the subset of data $d_n \subseteq D$ from which f_n can be computed, and the complexity of the algorithms that actually *Process* d_n in order to produce the feature value f_n

$$T_O(f_n) = T_C(f_n) + T_P(f_n), \quad \forall\, n \in [1, N]. \quad (18)$$

However, it is frequent that some groups of features $f^G \subseteq \mathbf{f}$ may be computed from the same subset of data, while also exploiting algorithms that have similar complexities. Therefore, groups of *homogeneous* features can be selected by imposing some constraints on the values of T_C and T_P

$$f^G = \{f_n : \quad \tau_1 \leq T_C(f_n) \leq \tau_2 \wedge \epsilon_1 \leq T_P(f_n) \leq \epsilon_2\}, \quad (19)$$

where τ and ϵ define a range of collection and processing times, respectively. In SpADe, $\mathbf{f}$ consists of 39 features (Table 2) that have been deeply analyzed in the literature and are demonstrated to be effective in capturing the characteristics of different spam behaviors. The criteria in Eq. (19) were applied in order to split $\mathbf{f}$ in homogeneous groups; as a result, four groups were identified. In particular, the properties of the account are observed at the first stage ($x_1 = f^A$), then URLs information is included ($x_2 = f^U$), content is evaluated at the third stage ($x_3 = f^C$), and finally neighborhood is visited ($x_4 = f^N$).

An analysis of the works in which the features adopted were first described (see Table 3) revealed that Bayesian Network (BN) and Random Forest (RF) are the most frequently chosen algorithms for their classification. The former is particularly suitable to evaluate the rejection due to its probabilistic nature, while the latter is proved to be one of the most proper classifiers when dealing with large feature sets [28], [41]. These considerations led us to make SpADe exploit Bayes classifiers to implement the accept/reject mechanism of the first three stages, while the last-stage relies on Random Forest.

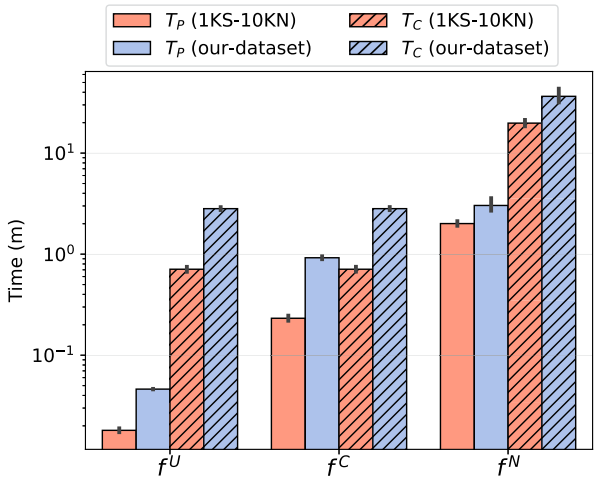

Fig. 4. Time to collect and process URL-, Content-, and Neighborhood-based features. Processing of f^U involves the tasks *TL* and *BL*, while *TL*, *ND* and *TC* are performed to compute f^C. The features in f^N are obtained through the tasks *CD*, *CB* and *NB*.

Given the methodological framework presented in the previous Section, classification performed at every stage s is based on a cumulative feature vector F_s that includes all the observation made so far, i.e., $F_s = \bigcup x_{1,\ldots,s}$. An overview of the multi-stage classification process is provided in Fig. 2. As long as the classification confidence does not reach the desired acceptance threshold, the process is repeated by choosing the rejecting option; however, at the last stage, when no further examinations are possible, a decision is made regardless of the achieved confidence. Multi-stage classification also results in a higher processing speed since the average number of features per stage is substantially lower than that required in a single-stage [42].

4.1 Feature Extraction

The following subsections describe the feature extraction processes that characterize each stage. A quantitative evaluation of the observation costs of the four feature sets is presented in Section 5.2, while other possible configurations are discussed in Section 5.3.

4.1.1 Stage 1: Account Analysis

Account analysis is the easiest to perform because the related features (top block in Table 2) can be easily extracted from public Twitter profiles. The features f_1 and f_2 capture the tendency of spammers to have a low number of followers and friend; f_3 and f_4, together, allow to detect accounts that, despite having been recently created, have produced a large number of tweets, which could indicate an automated spamming behavior. Finally, features f_5, f_6, and f_7 contain important statistics about the presence of *predatory elements*, such as favorites and hashtags. Although these features are extremely easy to compute, they can be altered just as easily, e.g., by buying followers in the so-called social media's black market. Thus, it is reasonable to exploit the set f^A to perform an early classification (at the first stage) only if the risk associated with a given observation is low; otherwise, it would be more convenient to extract more complex features at the next stages.

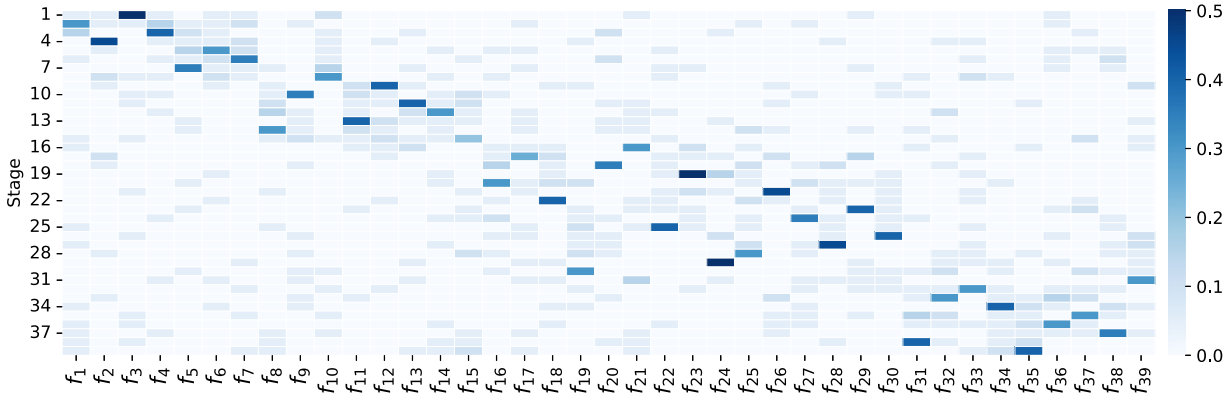

Fig. 5. Heatmap representing how many times (%) the features adopted in SpADe were chosen by CASCARO at the generic stage s_i, with $i \in [1, 39]$.

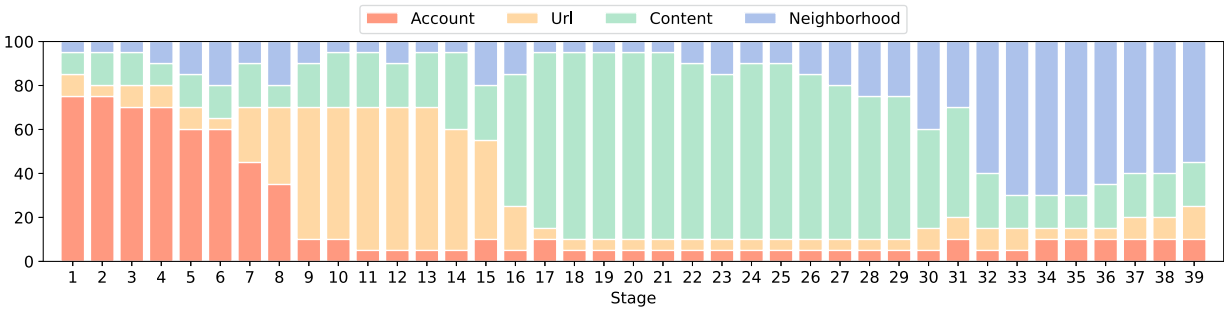

Fig. 6. Stacked bars representing the usage (%) of the groups of features defined in SpADe at the generic stage s_i of CASCARO, with $i \in [1, 39]$.

4.1.2 Stage 2: URLs Analysis

Tweets containing links to external websites are more likely to be re-tweeted, which is the primary goal of most spammers, i.e., to quickly reach as many people as possible. In order to evaluate the quality and the quantity of URLs shared by a user, we choose to rely on the 7 features reported in the second block of Table 2. The most intuitive element to observe in the user's timeline is the amount of URLs shared; in the simplest case, if every tweet contains an URL, then the probability the user is a spammer is very high. This aspect is captured by means of the features f_8, f_9, f_{10}, f_{11}. Aside from sharing a large number of URLs, some types of spamming activities are aimed to promote specific URLs, such as those related to commercial products or untrusted/malicious websites. The features f_{12}, f_{13}, f_{14} have been introduced to examine this kind of behavior; while f_{12} traces all URLs that contain spam-related keywords, such as those regarding money gain and adult contents [43], the feature f_{13} counts URLs that point to the same IP/domain, and f_{14} tests URLs for malicious contents by relying on third-party services, such as *Google'Safe Browsing*.

Since the computation of this set of features requires to retrieve all the user's tweets, the observation cost is clearly higher than the one measured at the first stage. Moreover, f_{14} requires external safe-browsing services, whose analysis-response time is not predictable.

4.1.3 Stage 3: Content Analysis

A straightforward characteristic of spammers is their tendency to repeatedly share same information, or *similar* information that implies the same content. In order to capture different aspects of content-based spamming, we selected the 16 features listed in the third block of Table 2.

The subset from f_{15} to f_{23} is intended to point out the differences between real tweets and those forged by automated spammers. The latter, for instance, are inclined to abuse popular hashtags so that their tweets can be more easily found; thus, the ratio of hashtags used in the tweets to the total number of tweets (f_{15}) is higher for spammers than trusted users. The same considerations apply to mentions, retweets, replies, and so on.

The analysis of timing is also important to evaluate if contents are shared by following regular (artificial) patterns. We address this aspect by means of the features f_{24} to f_{29}. For instance, the variance in the time taken by an account to post tweets (f_{24}), as well as the variance in the number of tweets (f_{25}) are two useful parameters to distinguish between bots and legitimate users, which are expected to tweet stochastically [16].

A further analysis is aimed to process the user's timeline in order to detect tweets that are not exact copies of each other, but differ in a few characters. The feature f_{30}, that highlights the presence of *near-duplicates* contents, is computed through an effective clustering approach based on the combination of two algorithms, namely MinHash and Locality-Sensitive Hashing (LSH) [44], [45]. The output of the *near-duplicates* detection is a collection of clusters containing similar tweets. Thus, the feature f_{30} is actually computed on this output and consists of a set of values representing the size and the number of clusters obtained from each timeline.

4.1.4 Stage 4: Neighborhood Analysis

The last, most computationally expensive, stage of analysis concerns the evaluation of the user as a member of a

community. As observed in [46], [47], it is quite difficult for a spammer to alter, or even influence, the behavior of its neighborhood, especially when composed of genuine users.

The most relevant characteristics to look at for performing neighborhood analyses are reported in the last block of Table 2. Some of them, i.e., $f_{31} - f_{34}$, describe the degree of interaction between users and their followers, or friends. The features f_{35} and f_{36} measure, respectively, the probability that two users become followers of each others (this value is expected to be high for genuine users that usually send requests to accounts they actually know), and the level of trust existing between two nodes, which, in the OSNs domain, depends on how close the connected nodes are, e.g., based on common friendships. The remaining two features, f_{38} and f_{39}, provide an evaluation of the account according to the community it belongs to. The former calculates the average reputation of the community in terms of reciprocity rate, i.e., the fraction of the users who follow back in response to followings. The latter studies the degree to which accounts tend to cluster together within the community. For both features the lower the value they provide, the more likely it is that the account is a spammer.

In order to obtain community-based features it is necessary to explore both the target user and all the accounts in its neighborhood. Hence, these features should be computed for a small number of users only, if previous classification stages have not led to a decision.

5 Experimental Results

After describing the data collection process, the following sections present a set of experiments aimed at tuning the system parameters and evaluating the classification performance. Then, comparative analysis are provided in order to assess the performance of SpADe with respect to some relevant related works.

5.1 Data Collection

Experiments have been carried out on two different datasets, whose characteristics are summarized in Table 4. The former is a reference public dataset, named 1KS-10KN [48], which consists of 11 k accounts and more than 1 million tweets crawled by means of the Twitter APIs in the period April-July 2010. The dataset contains also a set of features regarding the accounts and their timelines, as well as information about the URLs contained in the tweets. As its name suggests, 1KS-10KN is characterized by a ratio of spammers to genuine of 1:10, i.e., 1,000 accounts are labeled as *spammer* and the remaining 10.000 as *genuine*. More details about the dataset and the features provided can be found in [46].

The second dataset was collected by carrying out the procedure summarized in Fig. 3. Data collection starts by querying the Twitter stream through a set of static keywords, which will be successively refined. Initial keywords include elements that are generally used by spammers to reach as many users as possible. In particular, we chose a set of common spammy words [43] (such as "earn money", "free money", "no credit check", "viagra", "enlargement pill", "legal bud", etc.), and a list of trending topics/hashtags that were obtained from a preliminary API request. Tweets matching the queries are analysed by a topic detection algorithm [49], [50] with the aim of putting together similar tweets and finding out important terms emerging from them, such as keywords that characterize newly discovered topics or recently popular hashtags and mentions. Hence, the initial set of keywords is progressively updated by including these terms or deleting those unused. Such a strategy allows to acquire a large volume of data while keeping the focus on relevant topics. As a results, between June and December 2020 we collected 8 million tweets and 40 thousands accounts, which were processed to compute all the features required by SpADe. For each tweet we retrieved both the associated metadata (e.g., tweet ID, date of creation, and so on) and the author's ID, which is essential for obtaining account-related information. Then, for each account, the tweets in the timeline and the set of followers and followings were acquired. Finally, timeline and neighbors extractions were performed even on followers and followings accounts so as to capture information needed for computing the neighborhood-based features.

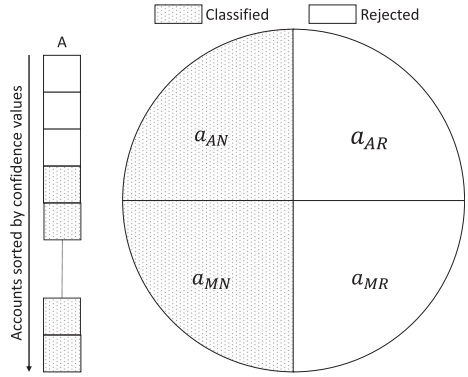

Fig. 7. According to the confidence value and the the reject ratio r, the **a**ccounts analysed at each stage can be **A**ccurately classified, **M**isclassified, **R**ejected, or **N**ot rejected. As a results, four sets can be identified: a_{AN}, a_{MN}, a_{AR}, and a_{MR}.

A semi-automatic labeling procedure [44] was adopted to assign ground-truth to collected data. The scheme consists of three phases. The first two automatically ascribe labels to "easy" users based on the analysis of the URLs and the similarity of the content shared: if URLs are malicious (e.g., blacklisted) or the timelines contain many repeated elements, users are labelled as spammers. Otherwise, manual annotation is required. In this case, the third phase aims to minimize label assignment effort by creating groups of similar users, perform manual annotation of just a few samples per group, and then extend the label to the whole set. The entire process is detailed in [44].

5.2 Observation Cost Evaluation

Quantifying the effort needed to observe the four categories of features outlined in Table 2 is crucial to implement the reject mechanism. According to Eqs. (18) and (19), the observation time of every group f^G is determined by the time required to *collect* and to *processes* the relative raw data.

The former amount depends on the capability of the observer to query the data source, e.g., by means of the available APIs. For instance, if the analyses are performed by the social media company itself, the collection time is negligible since all the data needed to calculate the features are available immediately. However, in the more general case, potential beneficiaries of SpADe include organizations

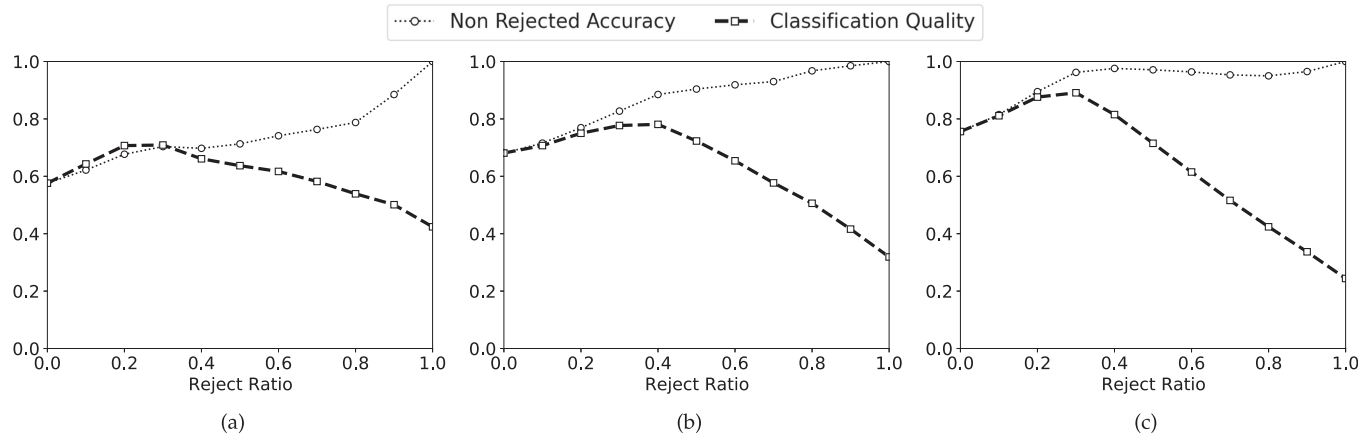

Fig. 8. Non-rejected accuracy and classification quality achieved at the first (a), second (b), and third (c) classification stage. Stage 4 is omitted as no reject option is allowed. Results shown in each column refer to the subset of samples rejected at the previous stage.

involved in countering cybercrimes (e.g., cyberbullying or other phenomena conveyed by OSNs), marketing and advertising companies interested in distinguishing real and fake accounts, government agencies working in the field of cybersecurity (e.g., for discovering data flows that could trigger devious political campaigns and misinformation), and even academic researchers, from both the fields of computer and social sciences, that being external to the OSN would need a certain amount of time to collect data to be processed. In general terms, this time for a group of homogeneous features, i.e., $T_C(f^G)$, can be defined as a function of four parameters, namely the number of data to collect (d), the maximum number of data that can be retrieved from a single API call (max_d), the maximum number of API calls allowed within a certain time window (max_c), and the duration of the time window itself (Δ_t).

The second component of the observation time, i.e., the processing time T_P, depends on the complexity of the algorithms chosen to actually calculate the feature values. For instance, the core processing tasks performed in SpADe involve *timeline browsing* (TL), *verification of blacklisted* URLs (BL), *near duplicate clustering* (ND), analysis of *tweets' contents* (TC), *community detection* (CD) and *browsing* (CB), and *neighborhood browsing* (NB).

The observation times of the URL-, Content-, and Neighborhood-based features discussed so far were assessed through an experimental analysis conducted by means of the two datasets presented in Section 5.1. Tests were run on a multi-core server equipped with 4 Intel Xeon at 2.00 GHz and results are summarized in Fig. 4. Please note that the evaluation of $T_O(f^A)$ is omitted since f^A is obtained directly from raw account information. The T_C and T_P bars exhibit similar trends on both datasets; in particular, it can be observed that neighborhood-based features (f^N) are the most expensive to obtain, both in terms of collection (striped) and processing (solid) times. The groups f^U and f^C are characterized by the same collection time since URLs are embedded in the tweets; however, processing content-based features takes more because of the computational complexity of near-duplicates analysis. Overall, results confirm an increasing trend in costs when moving from URL- to neighborhood-based features. It is worth noting that even in the specific case of the OSN data owner, though the collection time is negligible, the processing time is progressively higher.

These outcomes can be exploited to quantify the effort required to move from one stage to another. According to Eq. (17), the *loss* $\phi(x_s)$ of the multi-stage classifier at the stage s is strictly related to the cost of making the observation x_s. For instance, at the second stage of SpADe, x_2 corresponds to the set $F_2 = \{f^A, f^U\}$. Thus, by averaging and normalizing the results shown in Fig. 4, the following loss values are chosen: $\phi(F_s) \approx \{0, 10^{-2}, 10^{-1}, 1\}$. It is worth noting that the loss of the last stage is about 1, since the maximum cost is reached when all the features are considered together.

5.3 Stages Order Selection

The performance of a multi-stage classifier depends heavily on the order in which the features are employed. Using the most effective features in the early stages could in fact reduce the error, but may in some cases increase the overall complexity of the system.

One way to find the optimal sequence is by testing all possible permutations and determine the best trade-off between cost and error. However, if the number of features is large enough, this process may be impractical. As introduced in Section 4, and further discussed in the previous Section 5.2, the 39 features adopted in SpADe are grouped into four categories according to their semantics and observation costs. This design choice actually allowed us to reduce the search space from $39!$ to just $4!$.

A different approach is suggested in [51], where the problem of establishing a *good order* of the features to adopt in a multi-stage system is addressed. In particular, CASCARO is a method based on a variant of the Monte Carlo Tree Search (MCTS), in which the problem of variable ordering is treated as a search problem in a tree of depth D +1, where D is the number of features. Each path in the tree is associated with a *reward* that depends on Λ, a parameter representing the penalty in case of misclassification. In order to demonstrate the effectiveness of the feature order used in SpADe, we applied the CASCARO procedure to a system with 39 stages (one for each feature).

Results of the tests performed with $\Lambda \in [1, 20]$ are illustrated in Figs. 5 and 6. The former shows the percentage the feature f_n was chosen by CASCARO at the generic stage s_i; thus, the darker the cell, the higher the usage of that feature. As it can be observed, simplest features (e.g., account-based ones, with $n = [1, 7]$) are usually chosen in the initial stages,

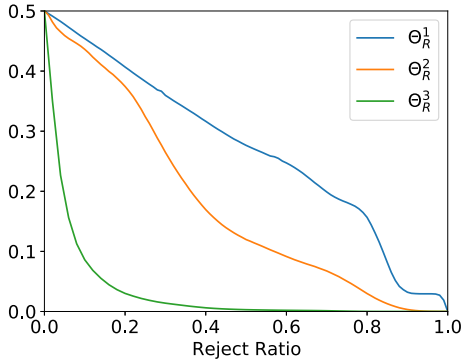

Fig. 9. Relationships between reject ratios and reject thresholds measured at the three classification stages.

while as the value of n increases, the choice of the corresponding features is postponed to later stages.

In order to further analyze this aspect, the individual features used in the CASCARO experiment were correlated with the groups employed in SpADe. Fig. 6 shows how many times the ith stage processed by CASCARO has used features belonging to one of the four groups in SpADe. The different colors highlight the existence of some patterns that, with a few minor exceptions, well match the distributions of the features we considered in SpADe. In particular, in the early stages of CASCARO *account*-related features are regularly selected; then, from stage 9 to 15 features ascribable to the *URL* group are chosen more frequently. Stages 16 to 29 extensively rely on *content*-based features, while at the remaining stages *neighborhood* information is preferred. These groups, regardless of the inner order in which the features are picked, correspond to those that are the core of our method. Thus, these results confirm the validity of the order adopted in the four stages.

5.4 Reject Threshold Evaluation

Choosing the proper threshold Θ_R^s at each stage is essential to balance the reject and classification rates, on which the performance of the spam detection system actually depends. On the one hand, low thresholds may increase the system accuracy as the decision to classify is made only when the outcome is almost certain; however, rigid thresholds could cause also inputs that would have been classified correctly at the current stage to be discarded. Conversely, high threshold values could lead the system to never reject, even when the outcome is uncertain. We present here a set of experiments aimed to find the proper threshold value for each classification stage.

In order to measure the performance of the classifier as a function of the fraction r of accounts rejected at each stage, we adopted the evaluation metrics proposed in [52]. Given a certain value of r, the classification of A accounts produces as output four distinct sets (see Fig. 7):

- a_{AN}: accounts **A**ccurately classified and **N**ot rejected;
- a_{MN}: accounts **M**isclassified and **N**ot rejected;
- a_{AR}: accounts **A**ccurately classified and **R**ejected;
- a_{MR}: accounts **M**isclassified and **R**ejected.

According to these quantities, two performance measures, namely *non-rejected accuracy* (NA) and *classification quality* (CQ), are defined

TABLE 5
Accuracy and F-Score Achieved by SpADe, Accounts (%) Rejected at the NTh Stage, and Total Percentage of Accounts Classified After the nth Stage)

	Accuracy	F-Score	Rejected (current %)	Classified (overall %)
Stage 1	0.90	0.91	91	9
Stage 2	0.92	0.93	69	37
Stage 3	0.91	0.95	27	83
Stage 4	0.95	0.96	-	100

$$NA = \frac{\|a_{AN}\|}{|A|}, \tag{20}$$

$$CQ = \frac{\|a_{AN} + a_{MR}\|}{|A|}, \tag{21}$$

The *non-rejected accuracy* measures the ability of the system to properly classify samples, accounts in our case, that are not rejected; the values of *non-rejected accuracy (NA)* can be used as rough indicators of the effectiveness of both the classifier and the features, evaluated on the "most evident" inputs, i.e., those that are not rejected. A more in-depth analysis of the reject region can be carried out through the *classification quality (CQ)* index, which assesses the performance of the classifier on the set of non-rejected accounts, and the reject policy on the set of misclassified accounts.

Fig. 8 shows the values of the two metrics computed on our dataset while varying the fraction of rejected inputs from $r = 0$ (no reject) to $r = 1$ (reject all). By observing the results of the first stage, Fig. 8a, we can notice that the more inputs are rejected the higher the *non-rejected accuracy*. The reason is that most of the inputs are uncertain because of the weakness of the features adopted. Hence, in order to achieve a proper level of accuracy it is necessary to impose a very high reject threshold. For instance, a threshold that guarantees a *non-rejected accuracy* of 90% would discard about 90% of the observed accounts (or accept only 10% of them). The same figure also provides a general measure of the effectiveness of the account-based feature set, which would lead to an accuracy of 60% without the reject option ($r = 0$).

Given a target reject ratio of 0.9, the *classification quality* observed at this first stage is quite low; in fact, striving for high accuracy in the non-reject region inevitably leads to reject also some "good" samples that could have been classified correctly. With respect to Eq. (21), this means that the number of a_{AN} and a_{MR} decreases as the reject ratio increases.

The performances observed at the second stage are more promising, as summarized in Fig. 8b. Here, in order to achieve the same target accuracy of 90%, only half of the analyzed accounts need to be rejected ($r = 0.5$). Even when no sample is rejected ($r = 0$), the features adopted at this stage are more accurate than the previous ones; this proves the usefulness of including URL information in the classification process. Moreover, also the classification quality increases, confirming that a proper threshold allows to obtain a satisfying number of correctly classified samples (a_{AN}), but also an adequate number of misclassified and correctly rejected samples (a_{MR}).

This trend is confirmed at the third stage, in which the reject ratio that maximizes the accuracy ($r = 0.3$) coincides

TABLE 6
Accuracy, F-Score, and Rejected Accounts (%) Measured While Varying the Ratio Between Spammers and Genuine Accounts

	1:2			1:5			1:10		
	Accuracy	F-Score	Rejected (%)	Accuracy	F-Score	Rejected (%)	Accuracy	F-Score	Rejected (%)
Stage 1	0.90	0.89	93	0.91	0.89	94	0.89	0.88	87
Stage 2	0.89	0.91	65	0.92	0.90	70	0.91	0.88	68
Stage 3	0.91	0.94	28	0.90	0.92	27	0.93	0.91	24
Stage 4	0.92	0.93	-	0.93	0.92	-	0.95	0.92	-

with the maximum classification quality, as depicted in Fig. 8c. It is worth noting that results shown in each column of the figure refer to the subset of samples rejected at the previous stage; thus, choosing a threshold that allows to achieve an accuracy of 90% at the third stage, for instance, would cause to reject only 30% of the accounts rejected at the second stage, which in turn are only half of those rejected at the second stage, which were 90% of those rejected at the first stage.

The reject ratios are strictly dependent on the reject thresholds chosen at each stage; thus, other tested were performed in order to highlight the relationship between the two values. We can observe from Fig. 8 that the lowest reject ratios for a target accuracy of 90% are 0.9, 0.5 and 0.3 at stage-1, stage-2, and stage-3 respectively. Then, Fig. 9 shows that these ratios can be obtained by choosing the thresholds $\Theta_R^1 = 0.03$, $\Theta_R^2 = 0.15$, and $\Theta_R^3 = 0.02$. Moreover, the same figure points out that as the system moves to the next stages, the reject thresholds decrease; this indicates a progressively reduced uncertainty because of an increasing feature significance.

5.5 Classification Performance

Once the observation costs were estimated and the classifier was tuned with the proper threshold values, a 10-fold cross validation was performed in order to assess the performances of SpADe in terms of overall accuracy, F-score, and percentage of classified accounts. Tests were repeated multiple times on a balanced subset of the dataset, randomly selecting the same number of spammers and genuine accounts.

Results, summarized in Table 5, indicate that the idea of progressively rejecting uncertain accounts allows the system to achieve an adequate (above 90%) classification rate at every stage. The F-Score values also highlights the ability of the system to drastically reduce the number of false positives and false negatives. Furthermore, it is possible to note that, stage by stage, the reject rate decreases as the feature sets become more and more significant. This last result suggests that even the effort required to process the features is progressively reduced; for instance, the neighborhood features are computed at stage four for only 17% of the initial set of accounts.

Since the performance evaluation might be biased by the ratio *n:m* between spammers (*n*) and genuine accounts (*m*), other tests were performed by considering different versions of the dataset with ratios 1:2, 1:5, and 1:10, which are increasingly more representative of the real social networks.

Table 6 shows that the system performances do not change significantly as different ratios are considered. Slightly better results are obtained when the proportion between spammers and genuine accounts is moderately unbalanced (e.g., 1:2); however, the average accuracy and f-score achieved in the 1:10 scenario are still above 90%. Also the percentages of inputs discarded at each stage are comparable, so confirming the quality of the rejection strategy.

The choice of the feature sets to adopt in each of the four stages was also supported by an experimental evaluation.

For each permutation of the four sets f^A, f^U, f^C, and f^N, a multi-stage system was defined and tuned by following the procedure described in Section 5.4. The performances of the 24 variations of the multi-stage system were measured in terms of their *classification cost*

$$\Phi = \sum_{s=1}^{S} \frac{a_c^s}{|A|}\ \phi(F_s), \tag{22}$$

where $a_c^s = ||a_{AN}|| + ||a_{MN}||$ is the number of account classified at each stage s by means of the cumulative feature vector F_s. It is worth noting that the value of Φ is maximum in a single-stage system, as all data are required at once in order to perform the classification.

Table 7 reports the percentages of accounts classified at each stage by imposing a target accuracy of 90%, where each row indicates a different sequence of feature sets and the overall observation cost of the resulting multi-stage classifier. The first sequence of features is the one we adopted in SpADe, which exhibits the lowest value of Φ. The last six rows show the highest percentage of accounts classified at the first stage, which demonstrates the effectiveness of the community-based features. However, these configurations also yield the highest observation costs, which reflects the effort required to compute the set f^N on a great number of accounts.

The costs of the other combinations we tested depend on how discriminative the feature sets are and the order in which they are used in the processing chain. In any case, it can be seen that none of the systems reaches the maximum complexity, stressing again the benefits of the proposed method over traditional approaches.

5.6 Comparison With State-of-the-Art ML Approaches

The last set of experiments aims to compare SpADe with three "single-stage" machine learning techniques without reject option, namely *Random Forest* (RF), *Decision Trees* (DT), and *Bayesian Networks* (BN). This choice is motivated by the results summarised in Table 3, which show that these algorithms are the most frequently employed in spam detection scenarios.

The main difference between our algorithm and those presented in the literature is undoubtedly the characterization of the observation cost. Hence, in addition to the

TABLE 7
Observation Costs of Different Configurations of the Multi-Stage System, Obtained by Varying the Order in Which the Four Feature Sets are Used

	%Classified@Accuracy0.9				
Feature sets order	Stage 1	Stage 2	Stage 3	Stage 4	Φ
f^A, f^U, f^C, f^N	9	28	46	17	**0.216**
f^A, f^U, f^N, f^C	9	28	47	16	0.486
f^A, f^C, f^U, f^N	9	14	53	24	0.255
f^A, f^C, f^N, f^U	9	14	77	0	0.786
f^A, f^N, f^U, f^C	9	80	8	2	0.807
f^A, f^N, f^C, f^U	9	80	10	1	0.815
f^U, f^A, f^C, f^N	18	1	60	21	0.272
f^U, f^A, f^N, f^C	18	1	81	1	0.809
f^U, f^C, f^A, f^N	18	20	24	38	0.399
f^U, f^C, f^N, f^A	18	20	62	0	0.638
f^U, f^N, f^A, f^C	18	72	7	3	0.724
f^U, f^N, f^C, f^A	18	72	7	3	0.729
f^C, f^A, f^U, f^N	73	1	3	24	0.312
f^C, f^A, f^N, f^U	73	0	27	0	0.343
f^C, f^U, f^A, f^N	73	2	3	22	0.292
f^C, f^U, f^N, f^A	73	2	25	0	0.325
f^C, f^N, f^A, f^U	73	24	0	3	0.316
f^C, f^N, f^U, f^A	73	24	2	0	0.316
f^N, f^A, f^U, f^C	89	1	2	8	0.898
f^N, f^A, f^C, f^U	89	1	7	3	0.897
f^N, f^U, f^A, f^C	89	2	0	9	0.899
f^N, f^U, f^C, f^A	89	2	1	7	0.891
f^N, f^C, f^A, f^U	89	8	0	3	0.898
f^N, f^C, f^U, f^A	89	8	3	0	0.898

accuracy and F-Score values, the comparative evaluations have to consider the complexity as defined in Eq. (22). Results of the comparison with *RF*, *DT*, and *BN* are reported in Table 8. Although these techniques are generally applied in balanced scenarios, we have selected a subset of our dataset in order to test their validity also with a spammer-genuine ratio of 1:10. Moreover, in order to make a fair comparison in terms of observation cost, we also considered a non-optimal configuration of SpADe based on the sequence of groups $\{f^N, f^U, f^C, f^A\}$. We refer to this system as SpADe*, which is characterized by an observation cost similar to that of the baselines.

All methods achieve fairly comparable performances in terms of accuracy and F-Score, but the classification cost of SpADe is extremely lower. The slight deterioration in the performance of SpADe* compared to the optimal configuration are mainly due to the fact that, although the SpADe reject option was tuned on a target accuracy of 90%, the last stage has no chance to reject and thus will be more prone to misclassification errors. Then, a higher probability of error subsists if poorly discriminative features are used at the last stage, such as f^A in the case considered.

The effectiveness and generality of SpADe were further assessed by setting up a new four-stage classification system that exploits a different set of features, namely the four categories of features described in [17]: *metadata*, *content*, *interaction*, and *network*. Also for this system, tested on the public 1KS-10KN dataset [48], we tuned the reject capability by computing the observation cost for the new feature sets, as well as the thresholds to be chosen at each stage in order to achieve 90% of accuracy.

Results, reported in Table 9, show that SpADe still outperforms the considered competitors. Moreover, it is possible to note that the performances of *RF*, *DT*, and *BN* are lower than those measured while using the proposed

TABLE 8
Comparison Between the Least (**SpADe**) and the Most Costly (**SpADe***) Versions of the System, Random Forest (**RF**), Decision Trees (**DT**), and Bayesian Networks (**BN**) Classifiers

	1:1			1:10		
	Accuracy	F-Score	Φ	Accuracy	F-Score	Φ
SpADe	0.92	**0.94**	**0.21**	0.91	**0.90**	**0.19**
RF	**0.93**	0.92	1	**0.92**	0.89	1
DT	0.90	0.90	1	0.91	0.87	1
BN	0.89	0.88	1	0.87	0.83	1
SpADe*	0.89	0.90	0.89	0.87	0.84	0.87

TABLE 9
Comparison Between the Least (**SpADe**) and the Most Costly (**SpADe***) Versions of the System, Random Forest (**RF**), Decision Trees (**DT**), and Bayesian Networks (**BN**) Classifiers Exploiting the Features From [17]

	1:1			1:10		
	Accuracy	F-Score	Φ	Accuracy	F-Score	Φ
SpADe	**0.90**	**0.91**	**0.20**	**0.89**	**0.87**	**0.18**
RF	0.85	0.86	1	0.84	0.81	1
DT	0.83	0.85	1	0.80	0.82	1
BN	0.77	0.79	1	0.75	0.71	1
SpADe*	0.84	0.85	0.88	0.81	0.80	0.85

feature set; this suggests that the features from [17] are probably less general in describing the spammers' behaviors. The costly version of the system, SpADe*, is characterized by a reduced detection accuracy that is very similar to *RF* and *DT*, and still superior to *BN*. These trends are almost the same regardless of the imbalance ratio, although the overall results are slightly worse in the 1:10 case.

In order to better understand the different performance of the methods considered as baselines, a final set of experiments was carried out to assess the contribution that different groups of features may have in such a "single-stage" classification process.

To this aim, the four groups f^A, f^U, f^C, and f^N were considered both individually and combined with each other, so making the cost of observing the feature subsets progressively higher. The leftmost plot in Fig. 10 shows that the accuracy of *RF*, *DT*, and *BN* increases as multiple groups are considered together, up to the best case - when the whole feature set is taken as input of the baselines. The curves indicate a very similar trend for the three classifiers, with RF providing slightly better performance regardless of the chosen feature group. The observation costs measured in the different cases provide further insights into the relationship between accuracy and efficiency of the classifiers.

The same procedure was repeated for the groups used in the experiments in Table 9, namely f^m, f^c, f^t, and f^n. Even in this case, we can observe a general trend indicating an improvement in performance as the observation cost increases, even though accuracy values are slightly lower than those observed in the private dataset.

These outcomes indicate that all features contribute to classification, i.e., they are not redundant, although the use of

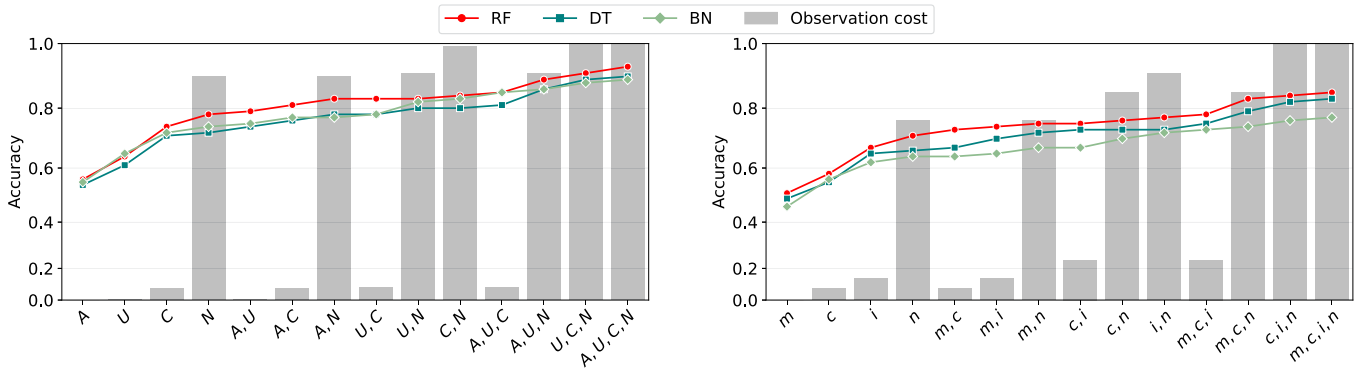

Fig. 10. Accuracy (coloured marks) and observation costs (grey bars) of *RF*, *DT*, and *BN* measured while considering different groups of features computed on the balanced versions of our dataset (left) and the public 1KS-10KN dataset [48] (right).

certain groups (e.g., f^C and f^N, or f^i and f^n) impacts on the performance more than others. This is also consistent with the other findings discussed in this Section, from which results the convenience of postponing the observation of expensive features to later stages, when fewer samples are evaluated.

6 Conclusion

In this paper we faced the problem of identifying spam accounts in social networks from a different perspective. Related works are generally oriented towards proposing new features capable of capturing the behavior of spammers, as well as new classifiers tuned on increasingly larger feature sets. However, feature acquisition and processing may be very costly in OSNs with millions of users. For this reason, we presented a multi-stage spam account detection technique with reject option, whose purpose is to initially exploit the features that are easier to compute, while progressively extracting more complex information only for those accounts that have not yet been classified.

The proposed system has been validated both on a dataset we retrieved from the Twitter stream, and on a reference public dataset. The performances have been also compared with single-stage state-of-the-art techniques that do not include the reject option, namely Random Forest, Decision Trees, and Bayesian Networks. The results highlighted the effectiveness of the multi-stage approach which achieves high accuracy in distinguishing between spammers and genuine accounts, while maintaining extremely low the overall complexity. These two characteristics are mainly due to the analysis, stage by stage, of increasingly significant features and to the ability of this system to classify the accounts only when it is quite confident of the outcome. Moreover, we observed that the accuracy of the multi-stage algorithm is comparable to that of a single-stage classifier that uses all the features at once; nevertheless, our approach allows to detect a spammer sooner, which also results in a lower complexity of the classification process.

The current approach equally weighs the misclassification of spammers and genuine accounts. However, while false positives could erroneously block honest users, undetected spammers could compromise the trustworthiness of the whole social network. As future work, this issue can be addressed by evaluating the effectiveness of a different loss function, which should be capable of assigning different penalties for an incorrect classification.

The solution proposed could be integrated in a more complex system in which the last classification stage is performed by entities that have knowledge of the problem, namely *experts* [53]. In fact, it happens more and more frequently that, especially in critical systems, machine learning algorithms are assisted by human experts that are able to better untangle uncertain situations. This kind of approaches must face a number of relevant open challenges, the most crucial of which is finding the right balance between classification accuracy and human overwork. These aspects will also be studied in future research.

References

[1] S. Rathore, P. K. Sharma, V. Loia, Y.-S. Jeong, and J. H. Park, "Social network security: Issues, challenges, threats, and solutions," *Inf. Sci.*, vol. 421, pp. 43–69, 2017.

[2] R. P. Barnwal, N. Ghosh, S. K. Ghosh, and S. K. Das, "Publish or drop traffic event alerts? Quality-aware decision making in participatory sensing-based vehicular CPS," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 1, Nov. 2019, Art. no. 9.

[3] R. Kaur, S. Singh, and H. Kumar, "Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches," *J. Netw. Comput. Appl.*, vol. 112, pp. 53–88, 2018.

[4] G. Lingam, R. R. Rout, D. Somayajulu, and S. K. Das, "Social botnet community detection: A novel approach based on behavioral similarity in twitter network using deep learning," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, 2020, pp. 708–718.

[5] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The rise of social botnets: Attacks and countermeasures," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1068–1082, Nov./Dec. 2018.

[6] F. Concone, A. De Paola, G. L. Re, and M. Morana, "Twitter analysis for real-time malware discovery," in *Proc. AEIT Int. Annu. Conf.*, 2017, pp. 1–6.

[7] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: Protecting online communities from spammers," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 1139–1140.

[8] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: A long-term study of content polluters on Twitter," in *Proc. 5th Int. AAAI Conf. Weblogs Soc. Media*, 2011, pp. 185–192.

[9] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, 2010, pp. 1–9.

[10] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," in *Proc. 4th USENIX Conf. Large-Scale Exploits Emergent Threats*, 2011, Art. no. 4.

[11] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots machine learning," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 435–442.
[12] S. Lee and J. Kim, "WarningBird: A near real-time detection system for suspicious URLs in Twitter stream," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 3, pp. 183–195, May/Jun. 2013.
[13] C. Cao and J. Caverlee, "Behavioral detection of spam URL sharing: Posting patterns versus click patterns," in *Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Mining*, 2014, pp. 138–141.
[14] W. Herzallah, H. Faris, and O. Adwan, "Feature engineering for detecting spammers on Twitter: Modelling and analysis," *J. Inf. Sci.*, vol. 44, no. 2, pp. 230–247, 2018.
[15] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on Twitter," in *Proc. Collaboration, Electron. Messaging, Anti-Abuse Spam Conf.*, 2010, pp. 75–83.
[16] A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu, and C. Yang, "CATS: Characterizing automation of Twitter spammers," in *Proc. 5th Int. Conf. Commun. Syst. Netw.*, 2013, pp. 1–10.
[17] M. Fazil and M. Abulaish, "A hybrid approach for detecting automated spammers in Twitter," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2707–2719, Nov. 2018.
[18] F. Concone, F. De Vita, A. Pratap, D. Bruneo, G. L. Re, and S. K. Das, "A fog-assisted system to defend against sybils in vehicular crowdsourcing," *Pervasive Mobile Comput.*, vol. 83, 2022, Art. no. 101612.
[19] D. Yuan et al., "Detecting fake accounts in online social networks at the time of registrations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1423–1438.
[20] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "COMPA: Detecting compromised accounts on social networks," in *Proc. 20th Symp. Netw. Distrib. Syst. Secur.*, 2013, pp. 1–17.
[21] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with Markov random field reasoning for social spammer detection," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 1054–1061.
[22] S. Nilizadeh et al., "POISED: Spotting Twitter spam off the beaten paths," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1159–1174.
[23] Q. Dang, Y. Zhou, F. Gao, and Q. Sun, "Detecting cooperative and organized spammer groups in micro-blogging community," *Data Min. Knowl. Discov.*, vol. 31, no. 3, pp. 573–605, May 2017.
[24] 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, 2014.
[25] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. A. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, Jul. 2013, Art. no. 13.
[26] H. Tajalizadeh and R. Boostani, "A novel stream clustering framework for spam detection in Twitter," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 3, pp. 525–534, Jun. 2019.
[27] D. N. Koggalahewa, Y. Xu, and E. Foo, "Spam detection in social networks based on peer acceptance," in *Proc. Australas. Comput. Sci. Week Multiconference*, 2020, Art. no. 8.
[28] C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, "Statistical features-based real-time detection of drifted Twitter spam," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 914–925, Apr. 2017.
[29] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 561–576, Jul./Aug. 2018.
[30] Z. Alom, B. Carminati, and E. Ferrari, "A deep learning model for twitter spam detection," *Online Soc. Netw. Media*, vol. 18, 2020, Art. no. 100079.
[31] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in *Proc. Australas. Comput. Sci. Week Multiconference*, 2017, Art. no. 3.
[32] X. Ban, C. Chen, S. Liu, Y. Wang, and J. Zhang, "Deep-learnt features for Twitter spam detection," in *Proc. Int. Symp. Secur. Privacy Social Netw. Big Data*, 2018, pp. 208–212.
[33] N. O'Mahony et al., "Deep learning vs. traditional computer vision," in *Proc. Sci. Inf. Conf.*, 2019, pp. 128–144.
[34] A. Soliman and S. Girdzijauskas, "AdaGraph: Adaptive graph-based algorithms for spam detection in social networks," in *Proc. Int. Conf. Netw. Syst.*, 2017, pp. 338–354.
[35] P. Pudil, J. Novovicova, S. Blaha, and J. Kittler, "Multistage pattern recognition with reject option," in *Proc. 11th IAPR Int. Conf. Pattern Recognit.*, 1992, pp. 92–95.
[36] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Trans. Inf. Theory*, vol. 16, no. 1, pp. 41–46, Jan. 1970.
[37] M. W. Kurzyundefinedski, "On the multistage bayes classifier," *Pattern Recognit.*, vol. 21, no. 4, pp. 355–365, Jul. 1988.
[38] K. Trapeznikov, V. Saligrama, and D. Castañón, "Multi-stage classifier design," *Mach. Learn.*, vol. 92, no. 2/3, pp. 479–502, Sep. 2013.
[39] N. Y. Hassan, W. H. Gomaa, G. A. Khoriba, and M. H. Haggag, "Supervised learning approach for twitter credibility detection," in *Proc. 13th Int. Conf. Comput. Eng. Syst.*, 2018, pp. 196–201.
[40] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 35–47.
[41] T. Wu, S. Wen, Y. Xiang, and W. Zhou, "Twitter spam detection: Survey of new approaches and comparative study," *Comput. Secur.*, vol. 76, pp. 265–284, 2018.
[42] A. K. Jain, "Advances in statistical pattern recognition," in *Proc. Pattern Recognit. Theory Appl.*, 1987, pp. 1–19.
[43] S. Sedhai and A. Sun, "HSpam14: A collection of 14 million tweets for hashtag-oriented spam research," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 223–232.
[44] F. Concone, G. L. Re, M. Morana, and C. Ruocco, "Assisted labeling for spam account detection on twitter," in *Proc. IEEE Int. Conf. Smart Comput.*, 2019, pp. 359–366.
[45] F. Concone, G. L. Re, M. Morana, and C. Ruocco, "Twitter spam account detection by effective labeling," in *Proc. 3rd Italian Conf. Cyber Secur.*, 2019, pp. 1–12.
[46] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, 2011, pp. 318–337.
[47] G. Wang, T. Wang, H. Zhang, and B. Y. Zhao, "Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers," in *Proc. 23rd USENIX Conf. Secur. Symp.*, 2014, pp. 239–254.
[48] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on Twitter," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 71–80.
[49] S. Gaglio, G. L. Re, and M. Morana, "A framework for real-time Twitter data analysis," *Comput. Commun.*, vol. 73, pp. 236–242, 2016.
[50] S. Gaglio, G. L. Re, and M. Morana, "Real-time detection of twitter social events from the user's perspective," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 1207–1212.
[51] B. Hanczar and A. Bar-Hen, "CASCARO: Cascade of classifiers for minimizing the cost of prediction," *Pattern Recognit. Lett.*, vol. 149, pp. 37–43, 2021.
[52] F. Condessa, J. Bioucas-Dias, and J. Kovačević, "Performance measures for classification systems with rejection," *Pattern Recognit.*, vol. 63, pp. 437–450, 2017.
[53] M. U. S. Khan, M. Ali, A. Abbas, S. U. Khan, and A. Y. Zomaya, "Segregating spammers and unsolicited bloggers from genuine experts on Twitter," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 551–560, Jul./Aug. 2018.

Federico Concone received the master's and PhD degrees in computer engineering from the University of Palermo, Italy, 2017, and 2021, respectively. He is currently a postdoctoral research fellow with the University of Palermo. During his PhD studies, his research mainly focused on social sensing and online social networks. His current research interests are in the area of cyber security, parallel and distributed computing, Adversarial Machine Learning, and Machine Learning techniques with applications in smart environments.

Giuseppe Lo Re (Senior Member, IEEE) received the laurea degree in computer science from the University of Pisa, in 1990, and the PhD degree in computer engineering from the University of Palermo, in 1999. He is a full professor of computer engineering with the University of Palermo, since 2019. In the 1991, he joined the Italian National Research Council (CNR), where he achieved the senior researcher position. His current research interests are in the area of computer networks and distributed systems, focusing on reputation and security systems. He is senior member of the Communication Society, and of the Association for Computer Machinery.

Marco Morana received the laurea and the PhD degrees in computer engineering from the University of Palermo, Italy, in 2007 and 2011, respectively. He is an assistant professor of computer engineering with the University of Palermo since 2016. During his PhD studies, his research mainly focused on computer vision and pattern recognition. His current research interests include parallel and distributed computing, social network analysis, cyber security, intelligent data analysis for user profiling, data fusion and reasoning in smart environments.

Sajal K. Das (Fellow, IEEE) received the PhD degree in computer science from the University of Central Florida, Orlando, Florida, in 1988. He is a professor with the Department of Computer Science and a Daniel St. Clair Endowed chair of the Missouri University of Science and Technology, Rolla, Missouri. His research interests include cyber-physical systems, security and privacy, smart environments (smart city, smart grid, smart healthcare), IoTs, wireless and sensor networks, mobile and pervasive computing, Big Data analytics, parallel, distributed, and cloud computing, social networks, systems biology, applied graph theory and game theory. He is the founding editor-in-chief of Elsevier's *Pervasive and Mobile Computing Journal* and an associate editor for several journals including *IEEE Transactions of Mobile Computing, IEEE Transactions on Dependable and Secure Computing*, and *ACM Transactions on Sensor Networks*.

Open Access funding provided by 'Università degli Studi di Palermo' within the CRUI CARE Agreement