Expert Systems With Applications 135 (2019) 129–152

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Unsupervised collective-based framework for dynamic retraining of supervised real-time spam tweets detection model

Mahdi Washha[a,*], Aziz Qaroush[b], Manel Mezghani[a], Florence Sedes[a]

[a] IRIT, University of Toulouse, CNRS, INPT, UPS, UT1, UT2J, France
[b] Department of Electrical and Computer Engineering, Birzeit University, Ramallah, Palestine

ARTICLE INFO

Article history:
Received 26 June 2018
Revised 4 April 2019
Accepted 29 May 2019
Available online 6 June 2019

Keywords:
Twitter
Real-time
Spam
Social spammers
Twitter stream

ABSTRACT

Twitter is one of the most popular social platforms. It has changed the way of communication and information dissemination through its real-time messaging mechanism. Recently, it has been used by researchers and industries as a new source of data for various intelligent systems, such as tweet sentiment analysis and recommendation systems, which require high data quality. However, due to its flexibility and popularity, Twitter has become the main target for spamming activities such as phishing legitimate users or spreading malicious software, which introduces new security issues and waste resources. Therefore, researchers have developed various machine-learning algorithms to reveal Twitter spam. However, as spammers have become smarter and more crafty, the characteristics of the spam tweets are varying over time making these methods inefficient to detect new spammers tricks and strategies. In addition, some of the employed methods (e.g. blacklisting) or spammer features (e.g. graph-based features) are extremely time-consuming, which hinders the ability to detect spammer activities in real-time. In this paper, we introduce a framework to deal with the volatility of the spam contents and new spamming patterns, called the spam drift. The framework combines the strength of unsupervised machine learning approach, which learns from unlabeled tweets, to retrain a real-time supervised tweet-level spam detection model in a batch mode. A set of experiments on a large-scale data set show the effectiveness of the proposed online unsupervised method in adaptively discovers and learns the patterns of new spam activities and achieve stable recall values reaching more than 95%. Although the average spam precision of our method is around 60%, the high spam recall values show the ability of our proposed method in reducing spam drift problems compared to traditional machine learning algorithms.

1. Introduction

The new great characteristics of Social Web that involve users as information producers have exposed different information quality (IQ) problems (Agarwal & Yiliyasi, 2010). For example, Twitter, which is the most popular microblogging sites, has a real-time messaging mechanism that makes it more popular and suitable for handling real-time public events and updates. In addition, due to its popularity, social-based researchers adopt it as a main source of information in performing their experiments on related research areas (Abascal-Mena, Lema, & Sèdes, 2015; Hoang & Mothe, 2016; Mezghani et al., 2015; Mezghani, Zayani, Amous, Péninou, & Sèdes, 2014; Zubiaga, Spina, Amigó, & Gonzalo, 2012; Zubiaga, Spina, Fresno, & Martínez-Unanue, 2011). However, the simplicity and flexibility of using these sites, and the absence of any effective restrictions on content posting action might be viewed as additional challenges for having IQ issues. Indeed, social spam content, which is published by a well-known kind of ill-intentioned users, so-called social spammers, is one of the most common noises appearing on online social media (OSM) sites and is categorized under IQ problems. Social spammers intensively post nonsensical contents such as advertisements, porn materials, viruses, malware, and phishing websites in different contexts (e.g., topics) and in an automated and systematic way (Benevenuto, Magno, Rodrigues, & Almeida, 2010; Washha, Qaroush, & Sèdes, 2016). Moreover, Social spammers exploit trending topics and available services or APIs to lunch their spammy content in short periods (e.g., one day) to maximize their monetary profits and speed up their spamming behavior. For example, on Twitter, social spammers leverage different set of provided services to launch their spam attacks through: *URL*, *Hashtag*, and *Mention* services. Besides these services, Twitter provides APIs for developers to be used in their third-party applica-

* Corresponding author.
E-mail addresses: mahdi.washha@irit.fr (M. Washha), aqaroush@birzeit.edu (A. Qaroush), florence.sedes@irit.fr (F. Sedes).

https://doi.org/10.1016/j.eswa.2019.05.052
0957-4174/

tions. Social spammers exploit this distinctive service as an opportunity to automate their spamming behavior.

Social spam might be defined as a piece of irrelevant information; however, this interpretation is quite not accurate. We justify this misinterpretation through the definition of information retrieval (IR) systems (Manning, Raghavan, & Schütze, 2008) in which the relevancy of the retrieved documents in the IR systems is dependent on the input search query. Thus, irrelevant documents with respect to an input query are "not" necessary to be a spam content. Hence, as an additional definition, social spam might be viewed as irrelevant information that doesn't have an interpretation in any context as long as the input query is not a spam content. Since social spam is a pure IQ problem, we project the problem on five IQ dimensions including accuracy, believability, reputation, value-added, and relevancy. Indeed, spam content does not represent real-world data, and thus it has a low degree in accuracy and believability dimensions. Also, the reputation of spam is also low because normal users tend to circulate accurate information in general. Finally, spam content doesn't deliver any benefit for the OSM users in terms of value-added and relevancy dimensions. Although projecting social spam problems on IQ world provides more insights regarding handling the problem efficiently; social spammers spend great efforts to increase the degree of IQ dimensions. Therefore, understanding and knowing facts about social spammers and their behaviors can contribute to providing effective solutions for the social spam problem. The work of spam detection is very important to both industries and academia because social spam is also very critical in other social media platforms.

Social media platforms, including Twitter, has recently been used as a new data source by researchers which potentially could have many applications within emergency management and crisis coordination, making the streaming of high-quality tweets a serious challenge with the continuous presence of ill-intentioned individuals (Imran, Castillo, Diaz, & Vieweg, 2015). Among these platforms, Twitter API is more open and accessible, which makes Twitter more favorable to developers to building tools to access data. In addition, Twitter data can contain valuable metadata including geospatial data. Research and applications on Twitter data ranges from sentiment analysis, time series analysis, and Network analysis which can be exploited in relating intelligent systems applied in industry, government, and universities (Giachanou & Crestani, 2016). For example, Twitter uses artificial intelligence techniques to determine what tweet recommendations to suggest on users timelines. Also, several companies use Twitter Sentiment Analysis to develop their business strategies, to find out customers feelings towards products or brand, and to predict the stock market movements (Mittal, 2011).

Several methods were presented in the recent research literature for detecting spamming activities on Twitter. However, these methods have several weaknesses that make them below the desired level of efficiency in detecting dynamic spammer activities in real-time. These weaknesses could be summarized as follows: (i) most of these methods are based on supervised machine learning approach which is trained on static, human-annotated datasets which is very human-labor cost, (ii) reliance on small and static annotated datasets to build a model to follow-up social spammers' patterns and their tricks, is not an efficient solution because of the lack of information in the tweet object itself, information fabrication problem, and the variation in social spammers' behaviors (Chen, Zhang, Xiang, Zhou, & Oliver, 2016; Grier, Thomas, Paxson, & Zhang, 2010). For example, Fig. 1 shows a sequence of streamed spam tweets that attacked the "KCA" event by three correlated spam accounts (social spammers). From this example, various patterns might be elicited: (1) same URL used in posting tweets; (2) the same prefix was exploited in filling screen-name attribute "voteddlovatu"; and (3) there was a high similarity in the user-name attribute among the three accounts, and (iii) previous studies didn't provide sufficient analysis and formulations regarding the features used by the spam detection model to work in real-time.

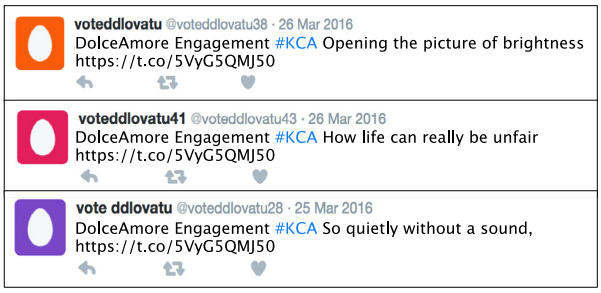


Fig. 1. An example of three correlated spam tweets posted in a consecutive way by three different spam accounts.

In spite of the volatility of the spam content and new spamming patterns, we can exploit the high correlation among spammers' behavior when they launch their spam campaigns for aggregating and analyzing continuously-streamed tweets using unsupervised methods to produce annotated datasets of tweets that can be used for repetitively retraining and updating spam classification models. In this paper, we introduce and experiment a framework that leverages an unsupervised method in providing, automatically and periodically, an annotated dataset for updating supervised real-time spam tweet detection model. More precisely, our framework is mainly composed of two main modules: (i) an online collective-based unsupervised spam tweet model; and (ii) a supervised real-time spam tweet detection model. The first module collects and stores streamed tweets. Then, it applies clustering methods on the stored tweets, followed by a rule-based method that labels each cluster of tweets to provide annotated tweets. The second module is responsible for classifying instantly every streamed tweet into spam or non-spam, with periodically leveraging the annotated datasets that come from the first module to retrain and update the current classification model. We demonstrate the effectiveness of the proposed framework through a series of intensive experiments conducted on a dataset streamed from 50 different Twitter trending topics consisting of more than 2 million tweets. Compared to two known methods designed for real-time spam tweets detection, the experimental evaluation shows the efficiency of the proposed framework in detecting spam tweets in terms of precision, recall, and *F*-measure performance measures. Also, it provides the ability to have control of the target quality of the tweets. In summary, the main contributions of the work introduced in this paper are: (i) provide an up-to-date survey and analysis regarding related studies organized in a hierarchy way, (ii) we have collected and labeled a large Twitter dataset of around 2 million Tweets from 50 different Twitter trending topics to be used in data analysis and experimental evaluation we will also make this dataset available for others researchers to use, (iii) providing a complete framework based on an online unsupervised learning method to deal with spam drift problems, by automatically and periodically preparing annotated training datasets to retrain a supervised tweet-level spam detection model, (iv) the proposed tweet-level classification model didn't require pre-training on the prepared annotated datasets. Additionally, no human intervention is required; which saves time, cost, and resources. (v) introducing an optimized set of discriminative, hardly manipulated, and lightweight features extracted only from the streamed tweets, without requiring any external information from Twitter's servers, (vi) compared to the state-of-the-art methods, the experimental evaluation shows the efficiency of the proposed framework in reducing the impact of spam drift problems.

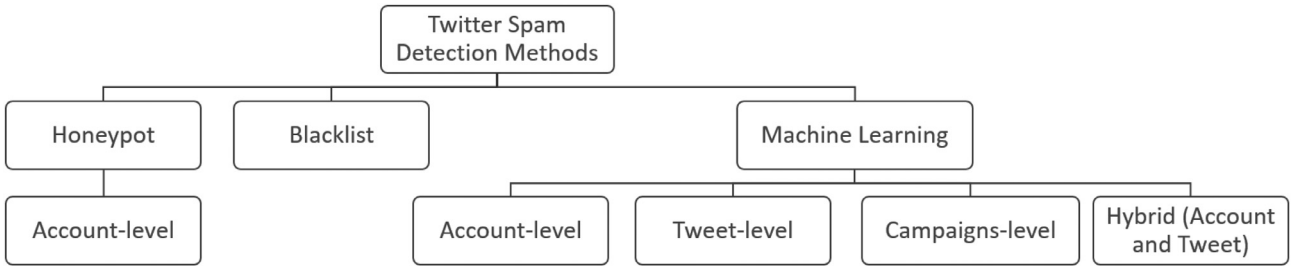


Fig. 2. A taxonomy for Social spam detection methods in Twitter (Kabakus & Kara, 2017; Washha, Shilleh et al., 2017; Wu, Wen et al., 2017).

2. Related work

Twitter gives its users the opportunity to report spam accounts through clicking "Report: they are posting spam" option that is available in all accounts. When an account is being reported, Twitter's administrators manually review and deeply analyze that account to make later suspension decision. However, such reporting mechanism is inefficient for fighting social spammers, because it needs significant efforts from both users and administrators when considering billions of users. Moreover, many users may provide fake reports and thus not all reports are necessary true. As an additional attempt to address the social spam problem, Twitter has a set of rules with permanently suspending the accounts that violate those rules (Twitter, 2016). Unfortunately, social spammers are smart enough to bypass Twitter's rules. For instance, social spammers may coordinate multiple accounts by distributing the desired workload among these accounts to mislead the detection. Consequently, Twitter's approaches are ineffective for reducing spam drift in real-time spam filtering.

The shortcomings in Twitter's anti-spam mechanism have motivated researchers to introduce more robust methods to increase data quality for the applications that use Twitter as a main source of information. After a deep look into a wide range of scientific research related to the spam detection methods on Twitter, we build a detailed taxonomy for these methods as shown in Fig. 2, which is based on different criteria, including: (i) type of the detection approach (Honeypot, Blacklist, and Machine Learning) and (ii) level of detection (Tweet, Account, and Campaign) exploited in the detection methods (Kabakus & Kara, 2017; Wu, Wen, Xiang, & Zhou, 2017).

2.1. Honeypot approach

Traditional supervised learning methods require an initial human labeled dataset which cannot work efficiently with spam drift issues. Alternatively, some spam discovery approaches rely on community reporting mechanism by using Social honeypot. A social honeypot is viewed as an information system resource that can monitor social spammers' behavior through logging their information such as the information of their accounts and any available content (Lee, Caverlee, & Webb, 2010). Social honeypots are valuable tool for gathering and understanding spamming activities, specifically community-based online activities. However, there is no significant difference between Twitter's anti-spam mechanism and the social honeypot approach. Both of them need administrative control to make a decision regarding the accounts that have been fallen into the honeypot trap to reduce the false positive rate, which in turns is time-consuming. Lee et al. (2010) deployed a set of honeypots for MySpace and Twitter and they used an updated trained classifier to identify spammers. Stringhini, Kruegel, and Vigna (2010) deployed 900 honeypots in Facebook, MySpace, and Twitter and they manually identified spammers from all requests.

2.2. Blacklist approach

Most spammers promote their services/products by embedding URL links in the spam tweet. Therefore, an effective way of spam detection is to detect tweets containing spam links which rely on the third party blacklisting techniques. For example, Twitter employed Googles Safe Browsing API to prevent malicious links (Grier et al., 2010). In fact, blacklist methods detect spam by searching the list and it can be applied for domain level rather than specific URL. Blacklisting techniques are commonly deployed in web filtering services such as Twitter spam detection or for dataset labeling. In addition, it provides a lightweight approach with a lower cost than existing classifiers (Ma, Saul, Savage, & Voelker, 2011). However, it cannot deal with the dynamic behavior of spamming activities and thus it's not appropriate for real-time detection because in average it takes 4 days for the blacklist to include the new spam URLs. In addition, many spammers try to embedded shortened URLs, which disable the performance of blacklisting techniques (Wu, Wen et al., 2017). Moreover, some URLs detection techniques are based on the correlations between the extracted URLs from several tweets in which require more time to retrieve tweets from Twitter servers (Lee & Kim, 2013).

2.3. Machine learning approach

To automate the task of spam detection, most of the Twitter spamming detection methods are based on machine learning techniques. However, the main difference between these methods is in the selected features along with their formulations. In fact, almost every paper in these methods introduce a group of distinct features and apply a set of well-known machine learning methods to detect spamming activities. Features used by machine learning methods for Twitter spam detection are varying and differ in terms of their level (e.g account, tweet, and campaign), formulations, powerfulness, ease of manipulation, and their suitability for real-time detection. Authors in Sedhai and Sun (2017a) and Yang, Harkreader, and Gu (2013) provide a comprehensive study and analysis regarding hashtag, tweet, account, graph, and timing features with respect to their performance in Twitter spam detection.

Selection of the machine learning method (e.g. supervised, unsupervised, or semi-supervised) is mainly based on the availability of annotated dataset. In fact, most of the spam detection methods have employed supervised machine learning algorithms, which are trained on one or more types of spam features, that are distributed between tweet-based features, account-based features, and campaign-based features. Account-based methods are based on the features extracted from a Twitter account such as user name, creation date, location, number of followings, number of tweets, number of mentions, number of moments, number of likes, and number of retweets. The works introduced in Benevenuto et al. (2010), Washha et al. (2016), Wu, Liu, Zhang, and Xiang (2017), Wang (2010), McCord and Chuah (2011), Stringhini et al. (2010), Meda et al. (2016), Bara, Fung, and Dinh (2015), Hu, Tang, and Liu

(2014) and Hu, Tang, Zhang, and Liu (2013) have been focused on extracting features (e.g., the number of friends, number of followers, similarity between tweets, and ratio of URLs in tweets) from users' accounts. In more dedicated studies, the works presented in Cao and Caverlee (2015) and Wang and Pu (2015) have identified the spam URLs through analyzing the behavior of shortening URLs such as the number of clicks and the length of the redirection chain. Features extracted from a single Twitter account are simple and lightweight. However, they are easily manipulated by social spammers using a group of bots. In addition, account-level detection is less effective for spammers who may act as legitimate users by posting nonspam content regularly. This behavior motivated researchers to leverage graph theory to extract more complex features from a set of Twitter accounts. For instance, the studies presented in Yang, Harkreader, and Gu (2011), Yang, Harkreader, Zhang, Shin, and Gu (2012) and Almaatouq et al. (2016) have examined the relation among users through using some graph theories and metrics to measure three features, including node betweenness, local clustering, and bi-directional relation ratio. Leveraging such complex features gives high spam accounts detection rate; however, they are not suitable for real-time Twitter-based applications, because of the huge volume of data that must be retrieved from Twitter's servers as well as graph operations, that require exponential time.

Tweet-level spam detection is a lightweight method that requires instant analysis and they are based on the features extracted from tweets such as tweet contents (e.g. text and links), sender, mentions, hashtags, links, number of retweets, number of replays, send dates, location (Kabakus & Kara, 2017; Wu, Wen et al., 2017). Tweet-level spam detection is essential to fight against spamming activities at a more fine-grained level. Most of these methods are based on using language models (e.g TF-IDF and bag-of-words) to compute the similarity between a tweet and other tweets in the same trending topic, or based on detecting malicious URLs embedded in the tweets (Lee & Kim, 2012; Martinez-Romo & Araujo, 2013; Thomas, Grier, Ma, Paxson, & Song, 2011). However, these methods may not be suitable for real-time filtering because it needs the tweets that have been posted on the same topic from Twitter's servers. Moreover, the traditional ways to filter URLs are based on blacklisting and HTML parsing which cannot handle shortened URLs, and take significant time to update blacklist. Other tweet features like sender, mentions, hashtags, links, number of retweets, number of replays, send dates, and location are simple and lightweight.

In campaign-level, researchers treated spam problem from a collective perspective view. Therefore, instead of detecting spam tweets one by one, they clustered spam into a set of groups according to their similarity on tweet contents or URLs (Wu, Wen et al., 2017). Chu, Widjaja, and Wang (2012) clustered a set of desired accounts according to the URLs available in the posted tweets. Then, a defined set of features from the accounts clustered is designed to build a binary classification model using machine learning algorithms to identify spam campaign. Chu, Gianvecchio, Wang, and Jajodia (2012) have proposed a classification model to capture the difference between bot, human, and cyborg with taking into consideration the content of tweets and spamming behavior. Campaign-based methods are not appropriate for real-time filtering due to the high volume of data required from Twitter's servers. In addition, some campaigns are classified manually which is extremely time-consuming (Thomas, Grier, Song, & Paxson, 2011). Finally, in hybrid-based methods, researchers use a combination of features extracted from Twitter accounts and tweets contents in order to provide more robust spam detection method (Chen, Zhang, Chen, Xiang, & Zhou, 2015; Chu, Gianvecchio, Wang, & Jajodia, 2010; Inuwa-Dutse, Liptrott, & Korkontzelos, 2018; Wang, Zubiaga, Liakata, & Procter, 2015). However, using hybrid features, requires careful features selection to make a trade-off between detection rate and detection time, in addition to the needed information from Twitter servers to compute features.

In general, supervised machine learning methods that have been used to detect spam tweets in real-time require a set of discriminative, lightweight, and not easy manipulated features, in addition to the existence of an annotated datasets for the training phase (Benevenuto et al., 2010; Chen, Zhang, Xie et al., 2015). Since tweet object has a limited amount of content or information, a few numbers of the features described in Table 1 are adopted in real-time spam tweet detection. Moreover, the major drawback of this approach is in the process of training classifiers, which is based on static datasets that reflect current spammer strategies without paying attention to the spam drift issues. To overcome this problem, we need to retrain classifiers periodically based on up-to-date datasets that catch the new spamming strategies (Chen et al., 2017). Authors in Chen, Zhang, Xiang, and Zhou (2015) proposed asymmetric self-learning method to update classifier periodically. They added the incoming classified tweets, which were classified using an initial trained model to the training dataset. Then, after a defined period (e.g., 1 day or 2 days), the classification model is retrained using the old training tweets along with the recent classified tweets. However, this approach is completely dependent on the initial trained model, and there is no guarantee about its performance in effectively detecting new social spammers' patterns. Similar approach are proposed in Lee et al. (2010). The authors employed honeypots as an information resource to monitor and collect spammers behaviors and log their information. They pass detected candidate spam profiles to the trained classifier and then return back profiles that are classified as a spammer to periodically update classifiers. However, the initial dataset used for training classifier is small. In addition, the approach includes human inspectors for validating the quality of the extracted spam candidates. Surendra and Aixin (Sedhai & Sun, 2017b) proposed a semi-supervised spam detection method consisting of two main modules: four-level spam tweet-based detection module which operates in real-time mode and an updating module which operates in batch mode. After finding the confidently labeled tweets, the detection module including blacklisted, near-duplicate, ham tweet detector, and tweet classification models are updated accordingly. The proposed detector uses tweet-based features which are suitable for real-time detection. However, some discriminative features that can be derived from the user account and historical tweets of the users are missing. In addition, among four detectors, the classification model contributes to 87% of tweets labeling, for which the performance is dependent on the initial data set. Chao et. al. (Chen et al., 2017) proposed a scheme called Lfun that can automatically detect changed spam tweets from new unlabeled tweets and incorporate them into classifiers retraining process. The proposed method employed 12 lightweight features and uses two components to extract changed spam tweets including learn from detected spam tweets and learn from human labeling. However, the performance of the scheme is mainly dependent on the first component which trained on an initialized labeled dataset. In addition, human labeling is time consuming.

To sum up, the above studies have the following weaknesses: (i) they assume that all pre-information (e.g., a blacklist of spamming domains and annotated dataset or trained classification models) to label tweets exist, (ii) most of them are based on updating dataset using the output of the classifier which can't guarantee to learn new spamming activities, since the classifier is trained on an initial dataset, and (iii) some of them are missing important lightweight account-level features, and (iv) some of them require human inspectors for validating the quality of the detector. Unlike these studies, our proposed framework have the following strengthens: (i) it didn't require pre-information like e.g., a blacklist of spam-

Table 1
A description of content and user features exploited in spam tweets detection, with classifying them based on their suitability for real-time filtering.

Feature Name	Description	Real-Time Suitability
	Content Features	
Number of Hashtags	The number of words that begin by "#" symbol (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of URLs	The number of links, including shorten links (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of Words	The number of words written in tweet where white-space is used a separator among words (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of Characters	The number of characters used in creating the tweet, including numbers and symbols (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of Mentions	The number of accounts mentioned in the tweet through looking for words starting by "@" (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of Retweets	The number of retweets that the tweet has gained (Benevenuto et al., 2010).	√
Number of Spam Words	The number of spam words that exist in the tweet according to a define list of spam words (Benevenuto et al., 2010; Martinez-Romo & Araujo, 2013).	√
Number of Trending Topics	The number of words that represent trending topics circulated in Twitter (Benevenuto et al., 2010; Martinez-Romo & Araujo, 2013).	√
Number of hashtags per words	The ratio of the number of hashtags to the number of words in the tweet (Benevenuto et al., 2010).	√
Number of URLs per Words	The ratio of the number of URLs to the number of words in the tweet (Benevenuto et al., 2010).	√
Number of Numeric Characters	The number of numeric digits in the tweet (Benevenuto et al., 2010; Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015; Martinez-Romo & Araujo, 2013).	√
Number of Replies	The number of times that the tweet has been replied by other users (Benevenuto et al., 2010; Martinez-Romo & Araujo, 2013).	√
Number of Favourites	The number of accounts/users that have favorited the tweet (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√
Tweet and URL Page Title Divergence	The Kullback Leibler Divergence value computed between the text of the tweet and the title of the URL website, if any (Martinez-Romo & Araujo, 2013).	✗
Tweet and Topic Content Divergence	The average value of the Kullback Leibler Divergence values computed between each tweet of the considered tweet topic and the text of the considered tweet, if any (Martinez-Romo & Araujo, 2013).	✗
Tweet and User's Tweets Divergence	The average value of the Kullback Leibler Divergence values computed between each tweet posted by the tweet user and the considered tweet content (Martinez-Romo & Araujo, 2013).	✗
User Features		
Number of Followings	The number of accounts/users that the user of the tweet follows (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√
Number of Lists	The number of accounts/users that has listed the user of the tweet (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√
Number of Followers	The number of accounts/users that follow the user of the tweet (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√
Account Age	The number of milliseconds spent since the creation date of the account of the tweet (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√
Number of Tweets	The number of tweets that the user of the tweet has tweeted (Chen, Zhang, Xiang et al., 2015; Chen, Zhang, Xie et al., 2015).	√

ming domains and trained classifier, (ii) our proposed framework is capable of continuously updating itself by using unsupervised learning method, which is based on a set of discriminative account and tweet-level features without any human intervention, and (iii) introducing an optimized set of discriminative, and lightweight account and tweet-level features extracted from only the streamed tweets, without requiring any external information from Twitter's servers.

3. Problem definition and formalization

Any Twitter stream can be represented as a finite set of chronologically sorted tweets, defined as $S_t = \{T_1, T_2, \ldots, T_{t-1}, T_t\}$, where $t \in \mathbb{N}^+$ represents the number of seconds since starting the streaming process, and T_1 and T_t are the first and latest tweets that has been streamed. Indeed, the Tweet object contains different attributes related to the tweet content and its user. Therefore, we represent the tweet element $T_\bullet$ by a 6-tuple of attributes, $T_\bullet =$ (*User*,*#Retweets*, *#Replies*, *#Favourites, Text, Time*), where *#Retweets* represents the number of retweets that the tweet has gained, *#Replies* is the number of comments performed as a reply on the tweet, *#Favourites* is the number of likes that the tweet has, $Time \in \mathbb{Z}_{\geq 0}$ is the posting date of the tweet in *seconds* time unit computed since January 1, 1970, 00:00:00 GMT, while the *Text* and *User* attributes are defined as follows:

- **Text:** The textual content of the tweet is represented as a finite set of ordered words, $Text = \{w_1, w_2, \ldots\}$. This set of words is extracted by segmenting the tweet content using the whitespace separator. The word element $w_\bullet$ might be a hashtag, URL, user's account mentioned, and more.
- **User:** Twitter provides simple meta-data about the user who posted a tweet. Hence, we further represent the user object by 7-tuple of attributes defined as, $User = (SN, UN, UA, \#Tweets, \#Followers, \#Followees, \#Lists)$, where *#Tweets* is the number of tweets that the user has posted on his account, *#Followers* is the number of accounts that follow the user, *#Followees* is the number of accounts that the user follows, *#Lists* is the number of accounts that list the user. The rest of the attributes, *SN, UN, UA*, are further defined as follows:
  - **Username (UN):** Twitter allows users to name their accounts with a maximum length of 20 characters. Users can use whitespace, symbols, special characters, and numeric numbers in filling their username attribute. This field is not necessary for being unique and thus the users can name their accounts by already used names. We represent this

attribute as a set of ordered characters, defined as $UN = \{d_1, \ldots, d_i\}$, where $d_\bullet \in \{Printable\,Characters\}$[1] is the character and $i \in \mathbb{Z}_{\geq 0}$ is the position inside the username string.

- **Screen Name (SN):** This attribute is a mandatory field and it must be filled at the creation time of the account. Users must choose a unique name that hasn't been used previously by other users, with a maximum length of 16 characters. Twitter also restricts the space of allowed characters to include only the alphabetical letters, numbers, and "_" character. Similar to the username attribute, we represent this field as an ordered set of characters, defined as $UN = \{d_1, \ldots, d_i\}$, where $d_\_ \in \{Printable\,Characters\}$ is the character and $i \in \mathbb{Z}_{\geq 0}$ is the position inside the username string.
- **User Age (UA):** When a user creates an account on Twitter, the creation date of the account is registered on Twitter's servers without providing any permissions to modify it in the future. We exploit the creation date, as an accessible and available property in the user's object, to compute the age of the account. Formally, we calculate the age in *days* time unit through subtracting the current time from the creation date of the account, define as $UA = \frac{Time_{now} - Time_{creation}}{864 * 10^5}$, where $Time_{now}, Time_{creation} \in \mathbb{Z}_{\geq 0}$ are number of milliseconds computed since January 1, 1970, 00:00:00 GMT.

According to this representation, the problem of real-time tweet-level spam detection can be defined as follows; Given a tweet streamed at time *t*, T_t, and a set of already streamed tweets, S_{t-1}, our problem is to predict whether the tweet T_t is a spam or non-spam, with leveraging only the available information in both T_t and the set of already streamed tweets S_{t-1}. More formally, we aim at designing a model, $F : \mathbf{x} \rightarrow \{spam, non - Spam\}$, which takes a feature vector of the streamed tweet $T_\bullet$ as an input and predicts its class label as an output.

4. Dataset description and ground truth

A dataset with ground-truth is required to train and evaluate a supervised machine learning spam detection methods. In fact, most of the researchers use their own dataset and some of them didn't make it publicly available (Benevenuto et al., 2010; Martinez-Romo & Araujo, 2013). In addition, for privacy reasons, when social-network-based researchers publish a dataset, they only provide the target object IDs (e.g., tweets and accounts) to retrieve them from servers of the desired social network. However, providing the IDs of the spam tweets or accounts is not enough because Twitter might already have suspended the corresponding accounts and thus nothing to retrieve from the servers.

The most challenging task in creating a large dataset is the annotation process. Currently, researchers are using four ways to generate ground truth, including: manual inspection, blacklists, suspended accounts, and clustering (Chen, Zhang, Chen et al., 2015; Hu et al., 2014; Hu et al., 2013; Sedhai & Sun, 2017a; Thomas, Grier, Song et al., 2011; Wu, Liu et al., 2017). Manual inspection is costly, time-consuming, and some times subjective. Blacklists (e.g. google safebrowsing) are an effective automated method. However, not all spam tweets contain URLs and also some spam tweets contain URLs that may direct to legitimate content. Therefore, Blacklist can be applied only for tweets containing URLs. On the other hand, suspended accounts are also automated method and work by labeling all of the suspended accounts tweets as spam. In fact, Twitter decides to suspend an account if it engaging spamming activities including posting misleading, deceptive, or malicious links.[2] However, sometimes, suspended accounts may contain non-spam tweets. Finally, in clustering methods (e.g. near-duplicate and expectation maximization), all tweets in the same cluster will be annotated with the same label. Up to our knowledge, there are two publicly available datasets suitable for tweet-level spam detection (Chen, Zhang, Chen et al., 2015; Sedhai & Sun, 2015). In Chen, Zhang, Chen et al. (2015) the authors used the blacklist method to annotate collected tweets, and thus they are concerned only on tweets containing URLs. On the other hand, authors in Sedhai and Sun (2015) used four main stages in the annotation process. However, due to the way the tweets were collected, the collection does not contain full user profiles, which limits extracting account-level features. In addition, the data set was collected based on popular hashtags, not on user basis, which does not guarantee to contain all tweets of any user. Therefore, since our methodology: (i) uses hybrid features (tweet-level and account-level) to train a real-time spam detection model, (ii) deals with all tweets not only the tweets having URLs, and (iii) uses tweets writing style similarity and tweets posting behavior similarity features to label new streamed tweet which require retrieving user tweets, both datasets are not suitable for our proposed approach. As a result, we decided to collect our own dataset and generate ground-truth.

Building large tweet dataset consists of two main stages, the collection stage, and the annotation stage. For the collection stage, we have developed a crawler that uses the Twitter Streaming APIs. Actually, real-time spam detection methods are applied on a stream of tweets related to one or more entities (hashtag, username, and URL). Therefore, to simulate and investigate such cases, we have chosen the hashtag as a target entity since most of the researches and applications stream the tweets from a particular hashtag or topic (Chellal, Boughanem, & Dousset, 2016; Hoang & Mothe, 2016; Sedhai & Sun, 2015; Zubiaga et al., 2012). We have launched our crawler for four months, started since 1/Jan/2015, where 2.1 million of relevant tweets from 50 trending hashtags have been collected and also stored based on their posting time. In the annotation stage, since manual labeling is expensive and blacklists are used only for tweets containing URLs, we have leveraged the suspended accounts method which is widely used in social spam detection to annotate our collected tweets (Hu et al., 2014; Hu et al., 2013; Thomas, Grier, Song et al., 2011; Washha, Qaroush, Mezghani, & Sèdes, 2017a; Wu, Liu et al., 2017). The process checks whether the user of each tweet was suspended by Twitter. In case of suspension, both the user and his tweets are labeled as spam. We have performed this process for one year after crawling the tweets in order to have a large number of spam users and their tweets.

In total, as reported in Table 2, we have found about 78,000 users (accounts) labeled as social spammer, and 881,000 legitimate users. Also, the number of spam tweets existing in our dataset is more than 208,000 tweets, forming about 10% of 2.1 million tweets. The number of tweets posted is obviously greater than the number of tweets retrieved since the former number represents the tweets that have been streamed into the hashtags selected, while the latter number corresponds to the ultimate tweets that have been posted since the creation of the accounts. As the dataset is not balanced at the class level, we compute the normalized version of the statistics per 100 users to have a more fair comparison between social spammers and legitimate users. The normalized version of the number of URLs shows an obvious misusing of URLs in spreading social spammers' content, compared to the legitimate users. It is expected that the number of verified users is zero since having a verified account requires to contact Twitter's administrators; thus the spam accounts are too difficult to be verified. Another interesting possible conclusion is that the distribution of spam tweets is not necessary to be uniform, meaning that social spammers may have some hidden preferences for selecting hash-

[1] http://web.itu.edu.tr/sgunduz/courses/mikroisl/ascii.html.
[2] https://help.twitter.com/en/rules-and-policies/twitter-rules.

Table 2
Distribution of different statistics for social spammers (spam accounts) and legitimate users (non-spam accounts) existing in our dataset.

	Social spammers				Legitimate users			
Statistic Name	Number	Percentage	Number (per 100 users)	Percentage	Number	Percentage	Number (per 100 users)	Percentage
Number of users	78,074	8.1%	–	–	881,207	91.9%	–	–
Number of geo-enabled users	5986	1.8%	8	18.2%	316,617	98.2%	36	81.8%
Number of verified users	0	0.0%	0	0.0%	2978	100%	1	100%
Number of users' followers	78,143,567	2.9%	100,089	25.8%	2,526,736,521	97.1%	286,735	74.2%
Number of users' followees	50,839,084	3.6%	651,165	81.5%	1,302,269,081	96.4%	147,782	18.5%
Number of tweets posted	944,566,070	5.4%	1,209,834	39.1%	16,604,525,699	94.6%	1,884,293	60.9%
Number of tweets retrieved	208,546	10.1%	267	55.8%	1,857,479	89.9%	211	44.1%
Number of retweeted tweets	80,773	9.1%	104	53.3%	808,263	90.9%	92	46.6%
Number of replied tweets	855	2.4%	1	33.3%	24,895	97.6%	3	66.6%
Number of URLs	127,655	1.9%	163	55.1%	1,166,666	98.1%	133	44.9%

Table 3
Distribution of 50,000 spam and non-spam tweets streamed into the top 20 hashtags existing in our dataset, showing an obvious variation in the number of spam tweets of 20 hashtags.

	Non-spam tweets		Spam tweets			Non-spam tweets		Spam tweets	
Topic name	Number	Percentage	Number	Percentage	Topic name	Number	Percentage	Number	Percentage
#iHeartAwards	39,478	78.9%	10,522	21.1%	#Harmonizers	38,844	77.7%	11,156	22.3%
#KCA	38,992	77.9%	11,008	22.1%	#quote	46,076	92.2%	3924	7.8%
#BestFanArmy	40,982	81.2%	9018	18.8%	#NowPlaying	47,841	95.7%	2159	4.3%
#TreCru	49,541	99.1%	459	0.9%	#BTS	48,798	97.6%	1202	2.4%
#Periscope	48,831	97.7%	1169	2.3%	#VoteMaineFPP	47,756	95.5%	2244	4.5%
#SoundCloud	46,709	93.4%	3291	6.6%	#gameinsight	45,492	90.9%	4508	9.1%
#np	47,150	94.3%	2850	6.7%	#VoteKathrynFPP	47,427	94.8%	2573	5.2%
#RT	36,922	73.8%	13,078	26.2%	#android	46,162	92.3%	3838	7.7%
#5SOSFam	41,476	82.9%	8524	17.2%	#love	46,802	93.6%	3198	6.4%
#Directioners	44,704	89.4%	5296	10.6%	#giveaway	47,602	95.2%	2398	4.8%

tags. More precisely, Table 3 reports the distribution of the spam and non-spam tweets streamed into top 20 hashtags and shows a clear variation in the number of spam tweets. The stream of some hashtags such as *#RT* has been intensively polluted with an estimated ratio of 1 spam tweet to 3 non-spam tweets, while there are hashtags that haven't been polluted too much like *#TreCru*. Indeed, there is no clear interpretation behind this high variation in the distribution of spam tweets; however, the importance of the hashtag, and how long time the hashtag has been trending are the most possible reasons.

5. Unsupervised collective-based and real-time spam filtering model

5.1. Model design: an overview

Supervised learning methods are the classical approach adopted in literature for building spam tweets detection models. As commonly known in the machine learning field, applying these methods need an annotated dataset. Unfortunately, having such data is often very expensive in terms of annotation time, and/or human resources. In addition, social spam classification models require continuous adaption using new training datasets to follow-up new social spammers' patterns and behaviors. Thus, obtaining a static training dataset to train a classification model is *not* an efficient solution at all.

Therefore, we propose a design of an online collective-based spam tweets classification framework that utilizes the great benefits of unsupervised machine learning methods, to periodically and automatically provide an annotated dataset by which updated supervised classification models can be produced. The model employs the correlation between social spammers' tweets in a short period to predict spamming behavior. As described in Fig. 3, our framework consists of two main modules: (i) real-time tweet filtering model; and (ii) periodic classification model learning. The first module prepares a feature vector for a streamed tweet through extracting a set of predefined light features and then passes the vector to an already learned classification model to predict the class label of the streamed tweet. The second module, which is the core of the framework, stores incrementally the streamed tweets in a storage component (e.g., database) and then frequently creates a newly labeled training dataset using unsupervised methods once a certain number of new tweets is stored in the storage component. Upon satisfying the condition of streamed tweets, a new feature space is prepared using all annotated tweets in the storage component. Finally, a classical supervised learning method (e.g., Random Forest, SVM, J48) is applied to the new labeled feature space to build a binary classification model to replace the current classifier model.

5.2. Collective-based unsupervised predictive model

Leveraging Twitter REST APIs to retrieve more information about users of the streamed tweets is the best solution to precisely label each tweet as spam or not. However, the impractically of this approach in terms of time brings serious challenges to design an efficient method suitable for processing large scale (sometimes endless) of streamed tweets. Therefore, instead of inspecting each tweet individually, we overcome this shortcoming by proposing an automatic approach that inspects the correlation between spam accounts and their tweets at different levels using unsupervised clustering methods. The design of the proposed approach comes in five-stages as illustrated through an example in Fig. 4. For a given set of streamed tweets, the first stage extracts the users who posted the tweets streamed. The second stage clusters the users set based on the age of each user's account. In the next stage, for each generated cluster, a defined number of communities is detected through an optimization process. In the fourth stage, we extract hand-designed features for each community using only users' tweets and accounts information. The last stage makes a decision about each community using a simple discriminative feature-based

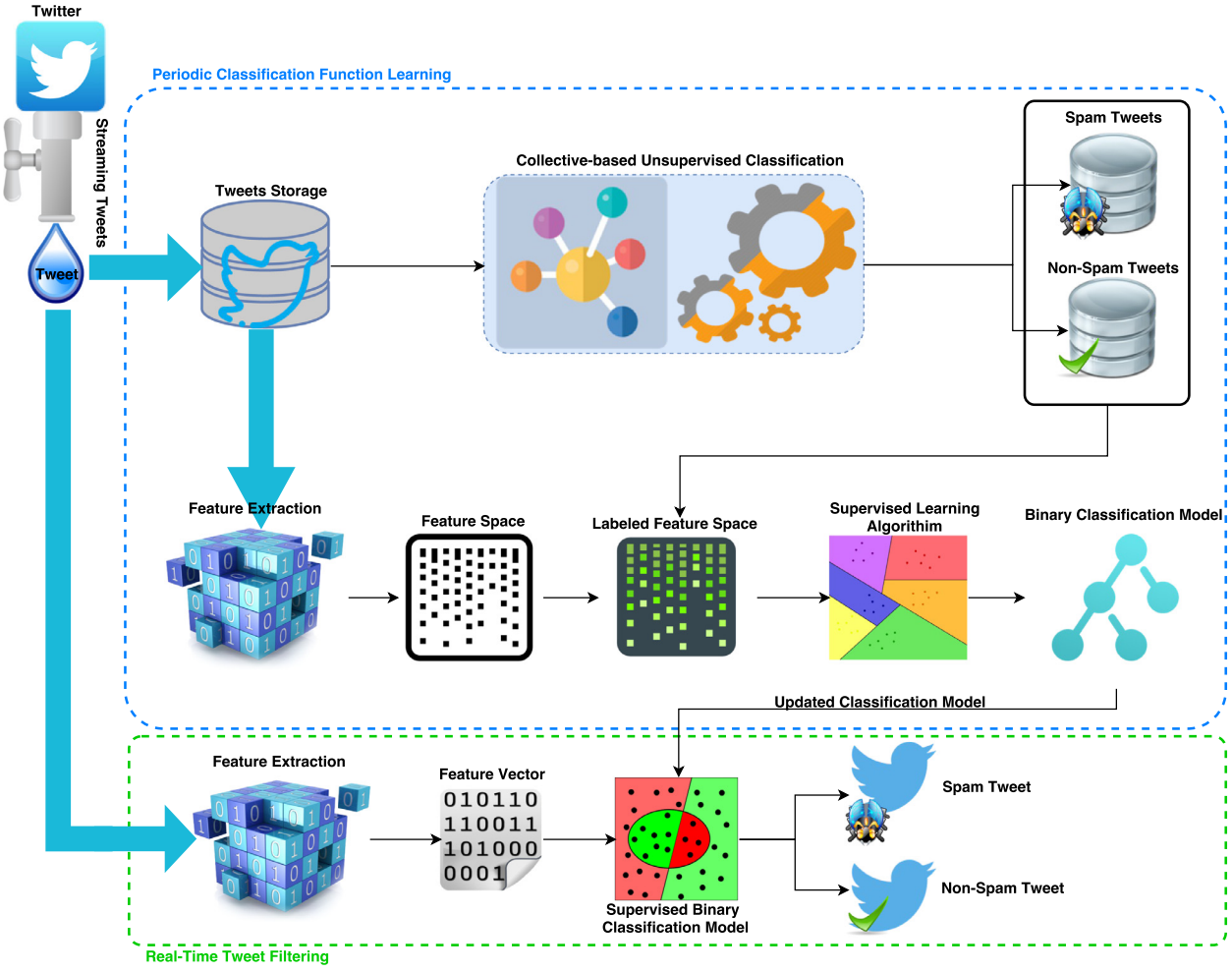


Fig. 3. A diagram showing the flow and the steps of the two main components in our framework: (i) periodic classification function learning; (ii) and real-time tweet detection or filtering.

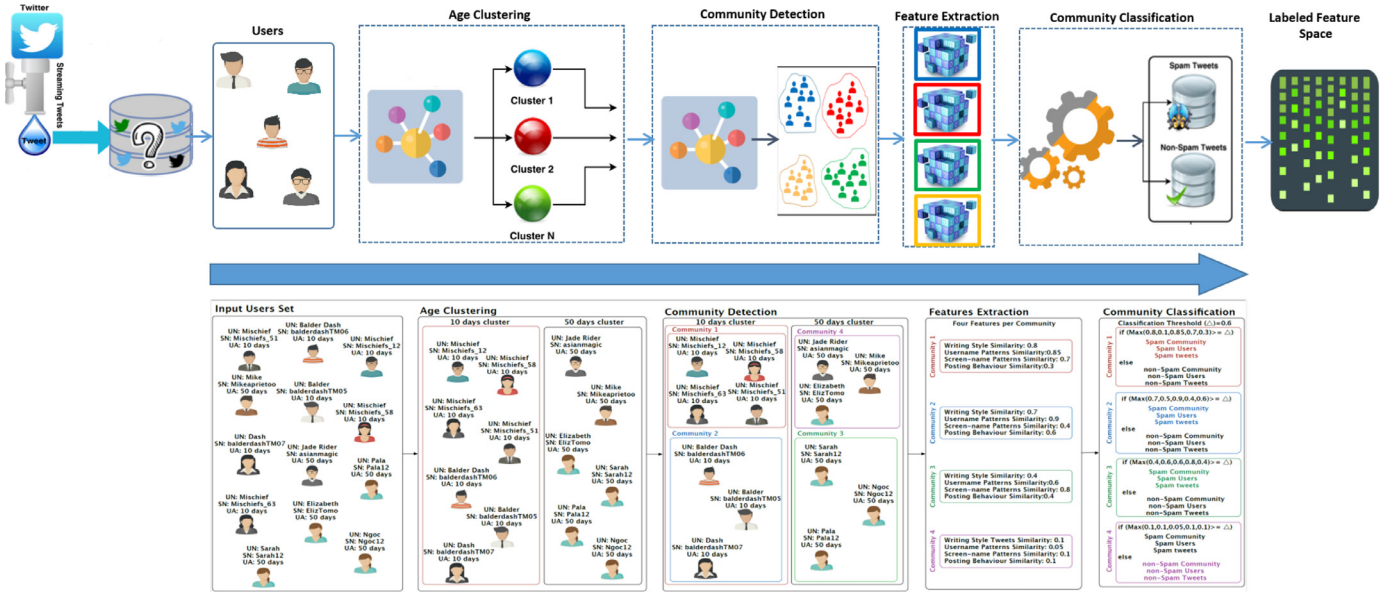


Fig. 4. An example describing the functionality of the 5-stage unsupervised classification: (i) user set extraction; (ii) account age clustering; (iii) community detection; (iv) community-based features extraction; (v) and community-based classification.

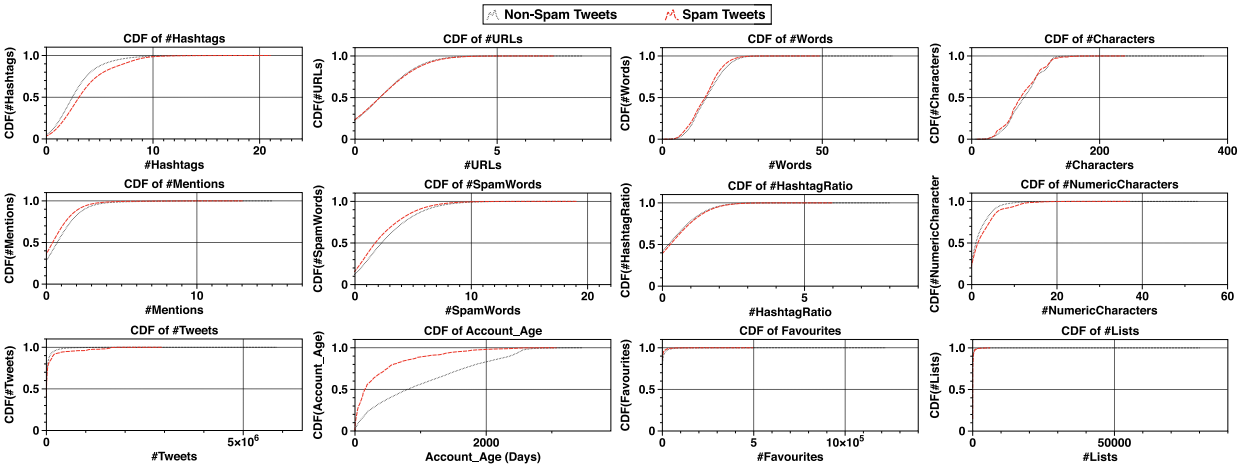


Fig. 5. CDF of 12 tweet features drawn for a randomly selected 10 days of streaming spam and non-spam tweets.

classification model that labels each tweet of spam communities as spam tweets.

Stage 1: Users Set Extraction. We design the clustering and the community detection stages based on leveraging the users' information that is available in the streamed tweets. Formally, for the latest streamed tweets set, S_t, the unique set of users is defined as $Users = \{T.User | T \in S_t\}$ where $|Users| \leq |S_t|$.

Stage 2: User Age-based Clustering. Social spammers have the ability to create hundreds or thousands of the Twitter accounts in a short period not exceeding few days, for launching their spam campaigns (Benevenuto et al., 2010; Washha et al., 2016). Also, the impact of the creation time has shown its effectiveness in detecting spam accounts so that when a set of accounts have a close and recent creation date, the probability of being a spam accounts increased. Fig. 5 show the Cumulative Distribution Function (CDF) of 12 tweet-level features which extracted for spam and non-spam tweets. These CDFs are computed for a randomly selected 10 days of tweet streaming to study the strength of these features. The CDF of the features shows that social spammers tend to behave exactly as legitimate users to avoid detection. An interesting point worth to mention is that the CDF of account age feature is the most robust one compared to the rest of the features since the creation date of accounts is non-editable by users.

Thus, the creation date of Twitter accounts can be an effective means for grouping the spam accounts that might have a correlation between them. According to that, the *Users* set is clustered based on the user age (UA) attribute. In a formal way, let $C_a^{Age} = \{u | u \in Users, u.UA = a\}$ be a day-cluster containing the users who have an account age equaling $a \in Ages$, where $Ages = \{u.UA | u \in Users\}$ is a set of distinct users ages. Obviously, the number of day clusters is dynamically determined, which exactly equals to the size of the *Ages* set (i.e., $|Ages|$).

Stage 3: Community Detection. Social spammers might create uncorrelated spam campaigns at the same time (Stringhini et al., 2010; Wang & Pu, 2015; Washha, Shilleh, Ghawadrah, Jazi, & Sèdes, 2017). In other words, we might have an age cluster containing spam accounts belonging to different spam campaigns. Also, many non-spam users join Twitter daily, which increase the probability of having non-spam users created on the same day as the spam ones. Therefore, to distinguish between different uncorrelated spam campaigns and non-spam accounts, a community detection stage is performed on each cluster resulted by age-based clustering stage. We define each spam campaign as a community having a high correlation between its users, where the correlation in a given community can be measured over naming accounts level, duplicated tweets content, or similar posting behavior.

In this paper, we adopt the Non-negative Matrix Factorization (NMF) as an unsupervised method, to infer communities' structure because of its outstanding performance in clustering problems (Yang & Leskovec, 2013). NMF has turned into one of the preferable tools for decomposing data into low-rank factorizing matrices to yield a parts-based representation. It has distinct features of preserving the structure of the original input data and keeping the non-negativity in both weight and basis. The latent semantic space of the NMF method has a very intuitive explanation in some clustering problems. For instance, in NMF based document clustering, each axis of the latent semantic space stands for the basic topic of a particular cluster where each document is represented by the additive combination of the basic topics.

For the given community detection problem, NMF works through partitioning or factorizing one or more information matrices into hidden factor matrices for an aging cluster, C_a, $a \in Ages$, of users. Formally, the factorization of information matrices is mathematically defined as an optimization minimization problem as:

$$\min_{H \geq 0} ||\mathbf{X} - \mathbf{H}\mathbf{H}^T||_F^2 = \min_{H \geq 0} \left(\sqrt{\sum_{i=1}^{|C_a^{Age}|} \sum_{j=1}^{|C_a^{Age}|} |x_{ij} - \mathbf{h}_i \mathbf{h}_j^T|^2} \right)^2 = \min_{H \geq 0} \sum_{i=1}^{|C_a^{Age}|} \sum_{j=1}^{|C_a^{Age}|} |x_{ij} - \mathbf{h}_i \mathbf{h}_j^T|^2 \quad (1)$$

where $||\bullet||_F$ is the Frobenius norm of the considered matrix, $\mathbf{X} \in R^{|C_a^{Age}| \times |C_a^{Age}|}$ is an information matrix representing the strength of the social connections (i.e., similarity among a pair of users) between users, $\mathbf{H} = [\mathbf{h}_1 \dots \mathbf{h}_K] \in R^{|C_a^{Age}| \times K}$ is the community structure hidden factor matrix of K communities, and the jth row vector $\mathbf{h}_j = [h_{j1}, \dots, h_{jK}] \in R^{1 \times K}$. The entry x_{ij} reflects the strength of the social connection between the $u_i \in C_a^{Age}$ user and $u_j \in C_a^{Age}$ user. The entry h_{ij} in the hidden factor matrix can be interpreted as the confidence degree of user $u_i \in C_a^{Age}$ belonging to the j^{th} community. It is important to mention that each user belongs to *one* community only, not more than one.

Obviously, inferring the hidden matrix $\mathbf{H}$ requires a formal definition of the information matrix $\mathbf{X}$. For example, $\mathbf{X}$ might be an adjacency matrix representing the social connections or the links among users of a given age cluster C_a^{Age}. However, obtaining the

adjacency matrix in our case is not possible since the available information about users is limited to simple meta-data that describe accounts, which did not give enough information about the followers and the followees. Hence, in this paper, we leverage the available and accessible information to estimate social connections between users through proposing three definitions of the information matrix $\mathbf{X}$ denoted as $\mathbf{X}^{SN}$, $\mathbf{X}^{UN}$, and $\mathbf{X}^{WS}$ where each of which is formally defined as follows:

- *Screen Name Similarity* (X^{SN}): Since the screen name field must be unique, social spammers tend to adopt a particular fixed pattern when creating multiple accounts acting as a spam campaign. For example, in Fig. 1, the spammer has adopted the name "voteddlovatu" as a fixed pattern and repeated it in filling the screen name field. Intuitively, the high matching in the screen name between users (or accounts) increases the probability of the users to belong to the same community. Therefore, we define the information matrix $\mathbf{X}^{SN}$ to measure the degree of matching between the screen name attribute. More precisely, for each two users $u_i, u_j \in C_a^{Age}$, $a \in Ages$, the degree of matching for a particular entry in the matrix $\mathbf{X}^{SN}$ is defined as:

$$x_{ij}^{SN} = \frac{max\{|m| : m \in N-gram(u_i.SN) \cap N-gram(u_j.SN), N \in \{1,\dots,min(|u_i.SN|,|u_j.SN|)\}\}}{min(|u_i.SN|,|u_j.SN|)} \tag{2}$$

  where $|\bullet|$ is the cardinality of the considered set, $N-gram(\bullet)$ is a function that returns a set of contiguous sequence of characters for a given name (set of ordered characters) based on the value of N. For better understanding, the 3-gram (or trigram (Cavnar & Trenkle, 1994)) of the screen name "vote" is {"*vot*", "*ote*"}. The above definition can detect the matched pattern wherever it appears in the screen name attribute. For instance, let "vote12" and "tovote" be screen names for two different spam users, the degree of matching according to Eq. (2) is around $(\frac{4}{6})66.6\%$, which resulted from the use of pattern "vote", regardless the position of the pattern.
- *User Name Similarity* (X^{UN}): Conversely the screen name attribute, spammers may duplicate username attribute as many as they wish. In fact, they aim to use a structured and representative (not random) names to attract legitimate users (Washha, Qaroush, Mezghani, & Sèdes, 2017b). Therefore, fully or partially matching among users based on such an attribute increases the probability of being in the same community. Thus, we define the information matrix $\mathbf{X}^{UN}$ to measure the degree of similarity among users based on the user name attribute. Formally, given two users $u_i, u_j \in C_{age}^{Age}$, the degree of similarity is defined as:

$$x_{ij}^{UN} = \frac{max\{|m| : m \in N-gram(u_i.UN) \cap N-gram(u_j.UN), N \in \{1,\dots,min(|u_i.UN|,|u_j.UN|)\}\}}{min(|u_i.UN|,|u_j.UN|)} \tag{3}$$

- *Names Writing Style Similarity* (X^{WS}): Based on our observations, social spammers may follow a particular style in writing the username and screen name attributes. For instance, in these two real spam accounts ($u_1 = \{SN = \text{"}vote5soss33\text{"}, US = \text{"}vote5sos\text{"}\}$, and $u_2 = \{SN = \text{"}saved028\text{"}, US = \text{"}saved\text{"}\}$), the social spammer has used the username attribute in filling the screen-name attribute with putting the username value in the beginning. Hence, as these two accounts belong to the same spam campaign, modeling such behavior can efficiently contribute to identifying spam communities. In order to model this similarity between a pair of users, we firstly define a function, $Pos(Str_1, Str_2)$, that takes two strings as an input and then it finds the location (**S**tart, **I**nside, **E**nd) of the string Str_2 in the string Str_1, written as:

$$Pos(Str_1, Str_2) = \begin{cases} SE & |Str_1 \cap Str_2| = |Str_2| = |Str_1| \\ S & |Str_1 \cap Str_2| = |Str_2| and \bullet_1 \in Str_1 \cap Str_2 \\ E & |Str_1 \cap Str_2| = |Str_2| and \bullet_{|Str_1|} \in Str_1 \cap Str_2 \\ I & |Str_1 \cap Str_2| = |Str_2| and \bullet_1 \notin Str_1 \cap Str_2 and \bullet_{|Str_1|} \notin Str_1 \cap Str_2 \end{cases} \tag{4}$$

where the two strings are represented as a finite set of ordered characters $Str_\bullet = \{d_1, d_2, \dots\}$, $d_\bullet \in \{Printable\,Characters\}$, the symbol $\bullet_1$ represents any character written at the beginning of a given string, while $\bullet_{|Str_1|}$ corresponds to the character written at the end of the string Str_1.

Therefore, for a pair of users $u_i, u_j \in C_a^{Age}, a \in Ages$, belonging to an age cluster, we define a writing style similarity matrix $\mathbf{X}^{WS}$ based on the equality of the pair in the *Pos* function value. For a particular entry in the matrix $\mathbf{X}^{WS}$, the similarity is defined as:

$$x_{ij}^{WS} = \begin{cases} 1 & Pos(u_i.SN, u_i.UN) = Pos(u_j.SN, u_j.UN) \\ 0 & otherwise \end{cases} \tag{5}$$

where here 1 means that the pair of users has same writing style, while 0 represents dissimilar writing style. For better understanding, when applying the *Pos* function on the given example of a pair of spam accounts (users), $(u_1 = \{SN = \text{"}vote5soss33\text{"}, US = \text{"}vote5sos\text{"}\}$, and $u_2 = \{SN = \text{"}saved028\text{"}, US = \text{"}saved\text{"}\})$, the "S" location is returned for the both users (i.e., $Pos(\text{"}vote5soss33\text{"}, \text{"}vote5sos\text{"}) = S, Pos(\text{"}saved028\text{"}, \text{"}saved\text{"}) = S$) since the username "vote5sos" appears in the beginning of screen name "vote5soss33" of the user u_1, and similar for the "saved" username of the user u_2. Thus, the writing style similarity equals to 1.

Non-negative matrix factorization method allows to integrate these three information matrices together in the same objective function. According to this, the objective function is defined as:

$$\min_{H \geq 0} ||\mathbf{X}^{SN} - \mathbf{H}\mathbf{H}^T||_F^2 + ||\mathbf{X}^{UN} - \mathbf{H}\mathbf{H}^T||_F^2 + ||\mathbf{X}^{WS} - \mathbf{H}\mathbf{H}^T||_F^2 \tag{6}$$

Obviously, the objective function in Eq. (6) infers the hidden factor matrix H to represent consistent community structure of related users. Indeed, this objective function is not jointly convex and has no closed form solution exists. Hence, we propose the use of a gradient descent approximation method as an alternative optimization approach. Since we have one matrix free variable ($\mathbf{H}$), the gradient descent method updates it iteratively until the variable converge. Formally, let $\mathcal{L}(\mathbf{H})$ denotes the objective function given in Eq. (6). So, at iteration τ, updating Eq. (6) is given by:

$$\mathbf{H}^{\tau} = \mathbf{H}^{\tau-1} - \eta . \frac{\partial \mathcal{L}(\mathbf{H}^{\tau-1})}{\partial(\mathbf{H})} = \mathbf{H}^{\tau-1} - 2\eta\Big(6\mathbf{H}^{\tau-1}(\mathbf{H}^{\tau-1})^T\mathbf{H}^{\tau-1} - (\mathbf{X}^{SN} + \mathbf{X}^{UN} + \mathbf{X}^{WS})\mathbf{H}^{\tau-1} - ((\mathbf{X}^{SN})^T + (\mathbf{X}^{UN})^T + (\mathbf{X}^{WS})^T)\mathbf{H}^{\tau-1}\Big) \tag{7}$$

where the parameter η denotes the gradient descent step in updating the matrix $\mathbf{H}$. We assign the value of η to a small con-

stant value (i.e. 0.05). Also, since the gradient descent method is an iterative process, a stop condition is required in such a case. For this, we used two stop conditions: (i) the number of iterations, denoted as M; and (ii) the absolute change in the H matrix for two consecutive iterations to be less than a threshold, i.e., $|(||H^{\tau}||_F - ||H^{\tau-1}||_F)| \leq \epsilon$.

One might view that the proposed information matrices and the age-clustering stage can be easily manipulated by social spammers to evade the detection. Indeed, this view could be correct when a social spammer creates very small spam bot consisting of no more than 5 spam accounts. Also, social spammers did not prefer to use a random function to generate screen names and usernames since the main objective of the social spammers is to lure legitimate users. Thus, social spammers have to use names suitable for the target that they want to achieve. For instance, if a social spammer wants to promote for a product "X" through devoting large spam bots, he must name the spam accounts using keywords related to the intended product. Social spammers have the option to create the accounts before the attack; however, they couldn't change the creation date attribute. Moreover, the purpose of using the age feature is to increase the difficulty in front of the social spammers to create thousands of accounts in short a period so that social spammers need to spend months to create a thousand of spam accounts. As the purpose of launching spambots is a monetary benefit, social spammers could not wait for this long time in creating their accounts as well as leaving them inactive may subject them for suspension from Twitter itself.

Stage 4: Community-Based Feature Extraction. In order to predict the class (spam or non-Spam) of each community, one or more features must be extracted from each community such that these features can effectively discriminate between spam and non-spam communities. Since social spammers may follow complex and different spamming strategies, no single feature can effectively discriminate between spam and non-spam communities. In addition, the design of such features must rely only on the available information in each community to avoid using REST APIs. Thus, we introduce a design of four community-based features that take into account the community's users along with their tweets. The four introduced features are distributed between account-based and tweet-based features. The username patterns similarity (**UNPS**), and the screen name patterns similarity (**SNPS**) are two features extracted using the username and screen name of the user attributes. On the other hand, Tweets writing style similarity (**TsWSS**), and Tweets posting behavior correlation (**TsPBC**), are tweet-based features which only leverage the available content in the tweets of a community. It is important to mention that there is a strong intuition behind the design of each feature, which will be illustrated statistically through different graphs of cumulative density function (CDF).

The total number of formed communities is dependent on the number of age clusters ($|Ages|$) beside the number of predefined communities K. Therefore, the ultimate number of communities is $|Ages| \times K$, where the community detection stage is applied to each age cluster. We represent the jth inferred community in the hidden matrix, H, by 7-tuple of attributes $C_j = (Users, Tweets, UNPS, SNPS, TsWSS, TsPBC, Label)$ where $Users$ is a finite set of the users belonging to the inferred community, $Tweets \subseteq S_t$ is all tweets that are posted by the users of the community, and $Lable \in \{spam, non-spam\}$ is the class label of the community. The remaining attributes are defined and formulated as follows:

- **Username Patterns Similarity (UNPS) and Screen Name Patterns Similarity (SNPS)**: Social spammers may adopt a particular pattern (e.g., "voteddlovatu") in creating their spam campaigns and therefore the probability of having spam communities biased toward a particular pattern used in creating accounts is relatively high. Since there is no obvious correlation among communities at the pattern level, nor a prior knowledge about the length and the name of the patterns, we must have a generic and independent way to determine whether the community has a spammy pattern. Thus, we rely on an intuitive and generalized fact which states that the probability distribution of the patterns in non-spam communities is close to the uniform distribution, while the spam communities have the opposite behavior. More precisely, we measure the degree of similarity between string patterns probability distribution extracted from users of a particular community with the uniform probability distribution of the patterns.

Formally, let PT^{UN} and PT^{SN} be two finite sets of string patterns extracted from the username and the screen name attributes for users of the j^{th} community, C_j. Also, let P_D^{UN} and P_D^{SN} be the corresponding probability distributions of the username and the screen name patterns, respectively. For the uniform distribution, let P_{uni}^{UN}, P_{uni}^{SN} be the corresponding uniform distributions of username and screen name patterns, respectively. For instance, for a particular community, let $PT^{SN} = \{\text{“}mischief\text{”}, \text{“}isch\text{”}, \text{“\_12”}, \text{“\_14”}\}$ and $P_D^{SN} = \{(\text{“}mischief\text{”}, 0.7), (\text{“\_15”}, 0.1), (\text{“\_14”}, 0.1), (\text{“\_12”}, 0.1)\}$ be a set of screen name patterns along with its probability distribution, and $\{(\text{“}mischief\text{”}, 0.25), (\text{“\_15”}, 0.25), (\text{“\_14”}, 0.25), (\text{“\_12”}, 0.25)\}$ be the uniform probability distribution of these patterns. To extract and catch all string patterns, the N-gram method is applied since social spammers may define patterns varying in their length and position. To perform the N-gram method, different values of N ranging from three to the length of the string are used, with ignoring low N values (one and two) because they provide meaningless patterns. For the jth community's users, represented as C_j, we extract the string patterns used in the username and screen attributes as follows:

$$PT^{UN} = \bigcup_{u \in C_j.Users} \bigcup_{N \in \{3,\ldots,|u.UN|\}} N-gram(u.UN)$$
$$PT^{SN} = \bigcup_{u \in C_j.Users} \bigcup_{N \in \{3,\ldots,|u.SN|\}} N-gram(u.SN) \quad (8)$$

The double unification ($\bigcup\bigcup$) can be viewed as a double "for" loops where the inner unification is responsible about returning all patterns, as a finite set of strings, that a single user has, while the outer unification unifies all sets of users' string patterns to have only one single set of the string patterns representing the community itself.

Since the pattern is a categorical random variable in which the string has not a meaningful order of magnitudes, we adopt the Kullback–Leibler divergence (Kullback & Leibler, 1951) (KL) method as a suitable and a fast way to measure the similarity between any two probability distributions of categorical random variables. However, the classical version of KL method cannot be directly exploited in computing similarity among (P_D^{UN} and PT_{uni}^{UN}) or (P_D^{SN} and PT_{uni}^{SN}) since the ∞ and 0 values correspond to dissimilar, and similar distributions, respectively. Hence, we perform a few modifications on the current version of KL method to inverse the semantic meaning of KL values (i.e., 0⇒ dissimilar and 1⇒ similar) and taking into account bounding its values. Thus, for the j^{th} community, the value of the $UNPS$ and $SNPS$ features are computed using the customized KL equation as follows:

$$C_j.UNPS = \frac{\log|PT^{UN}| - \sum_{w \in PT^{UN}} P_D^{UN}(w) * min(|\log \frac{P_D^{UN}(w)}{P_{uni}^{UN}(w)}|, \log|PT^{UN}|)}{\log|PT^{UN}|} \quad (9)$$

$$C_j.SNPS = \frac{\log|PT^{SN}| - \sum_{w \in PT^{SN}} P_D^{SN}(w) * min(|\log \frac{P_D^{SN}(w)}{P_{uni}^{SN}(w)}|, \log|PT^{SN}|)}{\log|PT^{SN}|} \quad (10)$$

where $|\bullet|$ is the cardinality (length) of the string patterns set, $P_D^{\bullet}(w)$ is the probability of occurring the pattern w based on the distribution of the considered patterns set, and $P_{uni}^{\bullet}(w)$ is the probability of occurring the pattern w according to the uniform distribution of the considered patterns set.

- **Tweets Writing Style Similarity (TsWSS)**: Single social spammer may create thousands of spam accounts for involving them in a spam campaign. Thus, the probability to have a correlation between the tweets of these accounts is quite high. According to our observations, the way or the style followed in writing tweets is mainly similar (e.g., one form) with possible correlations among them. For instance, the spam tweets of a campaign shown in Fig. 1 have a common style structure in writing tweets (word, word, hashtag, word, word, word, word, word, and the URL). It is obvious that the three tweets are too correlated, though their social spammer has been tricky in writing tweets through avoiding duplication in the content of the tweets. Computing the writing style similarity between tweets requires: (i) a new representation of each tweet through identifying the type (**W**ord, **H**ashtag, **U**rl, and **M**ention) of each whitespace separated string; (ii) and a metric that computes the degree of similarity among the new representation of the community's tweets. Therefore, we define a transformation function, $Type(ST) \in \{W, H, U, M\}$ that takes ST string as a parameter and returns the type of the input string (**W**ord, **H**ashtag, **U**rl, and **M**ention). Consequently, the new representation of a tweet $T_\bullet$ belonging to the jth community, C_j, is $Trans(T_\bullet) = \{(i, Type(w_i)) | w_i \in T_\bullet.Text\}$ where $i \in \mathbb{Z}^+$ is the position of the string in the tweet text and w_i is the string that requires a transformation. Unifying the new representation of all tweets provides a single unique set representing the writing style of the community's users. The cardinality of the unique set provides a meaningful indication about the writing style variation where the small cardinality means that the users have followed almost the same writing style. However, to precisely quantify how much the writing style is close among tweets, a reference value is required to compare the cardinality of the new set with it. The maximum value of the cardinality of the new set occurs when there is no intersection among the new representation of all tweets. Therefore, the cardinality of the new set will equal to the sum of $|Trans(T)|$ overall community's tweets. Formally, by the following equation, we measure the writing style similarity:

$$C_j.TsWSS = 1 - \frac{|\bigcup_{T \in C_j.Tweets} Trans(T)|}{\sum_{T \in C_j.Tweets} |Trans(T)|} \quad (11)$$

- **Tweets Posting Behavior Similarity (TPBS)**: Another possible form of correlation among spam accounts is the rate (e.g., every 5 min) of posting tweets. Intuitively, when the users (accounts) of a community have the same posting behavior, regardless of the posting period, the probability of the community being spam is high. The simplest way to compute the posting rate of a user is by examining the mean and the variance of the *Time* difference between every two consecutive tweets. However, social spammers can manipulate in these two statistics features through leaving a big gap between every two consecutive sets of tweets, leading to have a large variance and mean. We overcome this non-ignorable shortcoming through performing a quantitative user pairwise comparison at the posting time distribution of the user's tweets level. Then, a conclusion is drawn about the class label of the community based on the result of each pairwise comparison. Formally, for the j^{th} community, represented as C_j, let $P_{TS}^{u}[n]$ be the probability distribution of the tweet posting time of the user u, $u \in C_j.Users$ where $n \in \mathbb{Z}^+$ is a random variable representing the time in seconds. Since $P_{TS}^{u}[n]$ is a function of time and its random variable is quantitative in which its values with magnitudes have a meaningful order, we adopt the cross-correlation method which widely used in signal processing field for comparing two signals (Oppenheim, 1999), defined as follows:

$$PostSim(u_1, u_2) = \frac{\sum_{n=0}^{\infty}(P_{TS}^{u_1} \star P_{TS}^{u_2})[n]}{Min(\sum_{n=0}^{\infty}(P_{TS}^{u_1} \star P_{TS}^{u_1})[n], \sum_{n=0}^{\infty}(P_{TS}^{u_2} \star P_{TS}^{u_2})[n])}$$
$$= \frac{\sum_{n=0}^{\infty}\sum_{m=0}^{\infty} P_{TS}^{u_1}[m] P_{TS}^{u_2}[m+n]}{Min(\sum_{n=0}^{\infty}\sum_{m=0}^{\infty} P_{TS}^{u_1}[m] P_{TS}^{u_1}[m+n], \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} P_{TS}^{u_2}[m] P_{TS}^{u_2}[m+n])} \quad (12)$$

where u_1, u_2 are two different users belonging to the C_j community, "$\star$" is a symbol denoted to the correlation operation, and Min is a function that takes the minimum among two real number values. The correlation between two signals produces a new signal having different magnitudes where two highly correlated signals shall have large magnitudes. However, in order to quantify this correlation in a single real value, we compute the area under the new signal by adding the outer summation ($\sum_{n=0}^{\infty}$). As the area under the new signal (output signal) might be more than 1 and intuitively the maximum area is obtained when the two users' distributions are identical, we normalize it through computing the correlation between each user's distribution with itself, so-called auto-correlation, with taking into account the minimum among them as a normalization factor. For better understanding, Fig. 6 shows the posting time distribution (timely shifted and not normalized) of two different users having an obvious correlation in posting behavior. The cross-correlation between the two distributions has resulted in a new signal with an area of 18 (1+2+3+4+3+2+1). When applying the Eq. (12) on the given example, the value of the feature will be "1" since the area of the auto-correlation of each user's distribution equals to 18, meaning that the two users are completely correlated.

In computing the ultimate value of the *TPBS* feature, we compute first the probability distribution of *PostSim* over all possible user pairs existing in the $C^{j^{th}}$ community. Formally, let $P_{PostSim}$ (e.g., {(0.25, 0.4), (0.1, 0.6)}) be the probability distribution of the posting similarity and $P_{PostSim}^{Uniform}$ (e.g., {(0.25, 0.5), (0.1, 0.5)}) be the corresponding uniform distribution of *PostSim*. We quantify the difference between the distributions through performing cross-correlation between them, defined as:

$$C_j.TPBS = 1 - \frac{\sum_{n=0}^{\infty}(P_{PostSim} \star P_{PostSim}^{Uniform})[n]}{\sum_{n=0}^{\infty}(P_{PostSim}^{Uniform} \star P_{PostSim}^{Uniform})[n]}$$
$$= 1 - \frac{\sum_{n=0}^{\infty}\sum_{m=0}^{\infty} P_{PostSim}[m] P_{PostSim}^{Uniform}[m+n]}{\sum_{n=0}^{\infty}\sum_{m=0}^{\infty} P_{PostSim}^{Uniform}[m] P_{PostSim}^{Uniform}[m+n]} \quad (13)$$

where the high value (close to 1) of *TPBS* means that all users of the jth community have almost same posting behavior (i.e., almost same posting frequency) and thus that community has a high probability for being a spam campaign. On the other side, when the $P_{PostSim}$ be close to the uniform distribution, it means that almost no users have same posting behavior and thus that community has a low probability for being a spam campaign.

Stage 5: Community Classification Function. After computing the four community-based features for a community, the next step is determining whether that community is a spam or non-spam one. The main issue is what the best way to combine or weight the four features to form a community classification function. Handling robustly this issue requires to recall two key points: (i) the

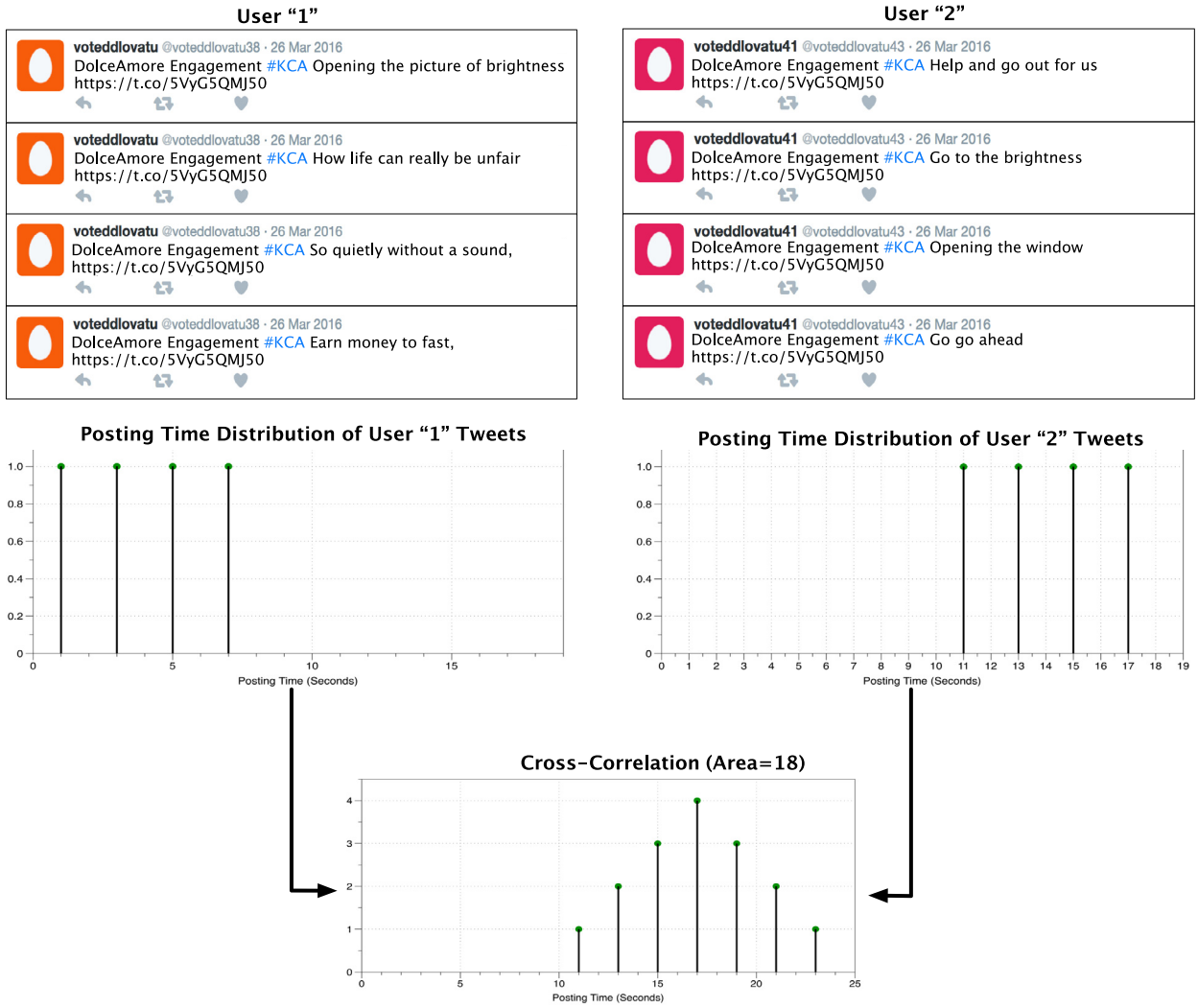


Fig. 6. An example describing the cross-correlation between posting time distribution of two different spam accounts (users).

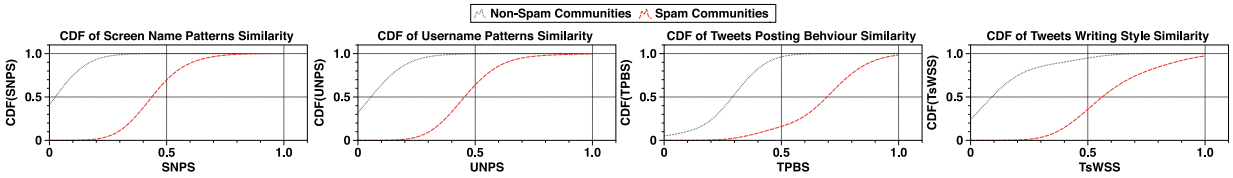
high values of the four features have a high degree of correlation with the probability of the considered community being a spam; (ii) and judging on a community as a spam needs at least one feature having a high value. The robustness and the strength of the four community-based features are easily captured through examining their CDF at different streaming periods. Thus, in Fig. 7, we show the CDF statistic of the four features drawn at three different 10-day streaming periods using the annotated dataset exploited in this paper. For each community feature, it is obvious that the area between the spam CDF and non-spam CDF is quite large, meaning that there is no too much overlapping between the value of the feature of the spam and non-spam communities. Also, the features of the non-spam communities have low values because of the early increasing in their CDF curves, while the features of the spam communities have the opposite behavior. Based on the two key points mentioned and the provided CDF statistics, we design a simple community classification through classifying the input community as a spam if one of the features has value more than a certain threshold Δ, formally defined for the j^{th} community C_j as follows:

$$C_j.Label = \begin{cases} spam & C_j.TsWSS \geq \Delta \;||\; C_j.TPBS \geq \Delta \;||\; C_j.SNPS \geq \Delta \;||\; C_j.UNPS \geq \Delta \\ non-spam & C_j.TsWSS < \Delta \;\&\; C_j.TPBS < \Delta \;\&\; C_j.SNPS < \Delta \;\&\; C_j.UNPS < \Delta \end{cases} \quad (14)$$

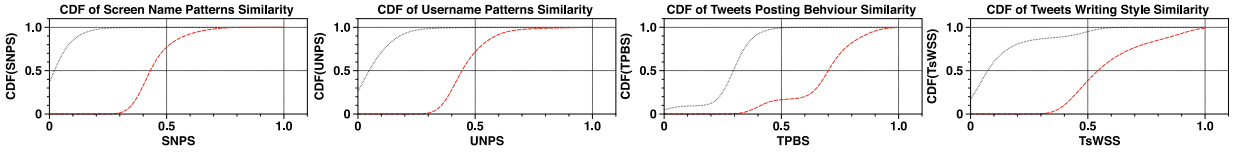
where "||" and "&" are "OR" and "AND" operations, respectively. The high value of $\Delta \in [0, 1]$ increases the difficulty of the conditions to be satisfied for labeling communities as a spam. Conversely, the low value of Δ leads to label too many communities as a spam. As the main purpose of the unsupervised classification is to provide an annotated dataset of spam and non-spam tweets, we leverage the label assigned for each community inferred through inheriting the label of each community to its users and their tweets available in the storage component. Formally, the annotated version of all tweets streamed, S_t, is extracted as follows:

$$Spam\_Tweets = \bigcup_{\substack{j \in \{1,\ldots,|Ages| \times K\} \\ C_j.Label = spam}} C_j.Tweets$$

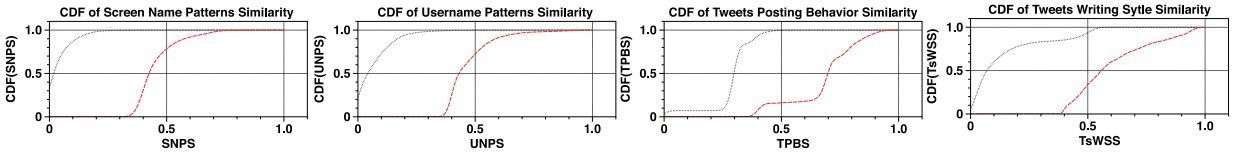
$$Non-Spam\_Tweets = \bigcup_{\substack{j \in \{1,\ldots,|Ages| \times K\} \\ j.Label = non-spam}} C_j.Tweets \quad (15)$$



(a) CDF of 4 community-based features drawn for the first period of spam and non-spam tweets.



(b) CDF of 4 community-based features drawn for the second period of spam and non-spam tweets.



(c) CDF of 4 community-based features drawn for the third period of spam and non-spam tweets.

Fig. 7. Cumulative distribution function (CDF) of non-spam and spam communities drawn for our four collective-based features at three streaming periods where each period is 10 days, showing the effectiveness of these features in discriminating among spam and non-spam communities.

where $Spam\_Tweets \cup Non - Spam\_Tweets = S_t$. As these two annotated sets of tweets will be exploited in learning a binary classification function, the value of Δ also plays an important role in the size of these sets. For instance, setting Δ to 0.9 will likely produce a small size of spam tweets set since this condition might be satisfied on few communities.

5.3. Tweet classification model and real-time detection

We follow the classical approach for producing a binary classification model, $F(\mathbf{x})$, and that through applying the classical widely used machine learning algorithms trained on the annotated set of tweets. As a prerequisite for performing the training phase is defining the feature vector that will represent the tweet entity. Thus, we adopt 17 lightweight features described in Table 1 for building the feature vector. The steps for learning the classification model start by preparing the feature space consisting of the feature vector of the training tweets. Then, the feature space is labeled using the output of the unsupervised classification stage performed on the training tweets. The feature space can be easily viewed as a two-dimensional matrix with a size of $|S_t| \times 18$, where 18 is the sum of the size of the feature vector (17) and the class label (1). At last, classical supervised learning methods such as Random Forest, K-NN, and J48 could be applied on the labeled feature space to obtain a binary classification model, $F_{S_t}(\mathbf{x})$. Once the learning phase gets finished, the old classification function is replaced by the new one. It is important to mention that the training or learning phase is taken place in a background process whenever a new number of tweets (e.g., 500 tweets) is streamed, so-called "Updating Model Frequency". As an initial classification model, we classify incoming tweets as non-spam, $F_{Initial}(\mathbf{x}) = "non - spam"$, until the updating model frequency condition is satisfied to have a binary classification model.

At the operational real-time filtering phase, the current adopted classification model is used to predict the class label of every incoming tweet. Predicting the incoming tweet type requires first to extract the feature vector of the tweet, using the same 17 features leveraged in the training phase. Then, the class label of the considered tweet is predicted using the current classification model ($F_{Initial}(\mathbf{x})$ or $F_{S_t}(\mathbf{x})$).

6. Experimental setup and results

6.1. Experimental setup

Performance Metrics. Since the ground-truth of 2.1 million tweets which belonging to 50 different hashtags is available, we adopt *three* commonly used measures in classification problems to evaluate our proposed framework besides two states of the art methods. These measures include precision, recall, and f-measure, computed according to the confusion matrix of the Weka tool (Hall et al., 2009). These measures computed based on True Positive, False Positive, True Negative and False Negative parameters, where True Positive refers to detecting a spam tweet which is actually a spam tweet. We didn't report accuracy because our model is a binary classifier and our main task is to detect spam tweets, not non-spam tweets. In addition, the collected dataset is not balanced and thus the accuracy measure will not be informative.

Methods and Parameters. We compared our framework with two real-time spam detection methods presented in the literature, denoted as "Classical (traditional ML)" and "Asymmetric Self-learning (Chen, Zhang, Xiang et al., 2015)" methods. The classical one works through performing training on an annotated dataset of tweets for a once to build a classification model. After that, the trained model will be used at the operational detection phase all the time without retraining the model again. The Asymmetric Self-learning (Chen, Zhang, Xiang et al., 2015) method also uses a trained classifier model to detect spam tweet. However, the initial dataset was updated by adding each new streamed tweet and using the output of the classifier as the ground-truth label for that tweet. After streaming a certain number of tweets, the classification model is updated using the updated dataset. To evaluate these methods, a set of experiments were conducted on (i) various machine learning methods, including Random Forest, Decision Tree (J48), and K-Nearest Neighbor (K-NN) where WEKA tool is used as an implementation for these algorithms, (ii) various number of the training tweets (e.g. 500, 1000, and 5000), and (iii) various up-

Table 4
Parameters and learning algorithms setting for the classical, asymmetric self-learning (Chen, Zhang, Xiang et al., 2015), and our collective-based methods.

Approach	Learning Algorithms	Updating Model Frequency (Tweets)	Training Tweets	Unsupervised Classification Parameters
Classical (Traditional ML) Method	Random Forest (RF) – Number of Trees:10, 100, 500 Decision Tree (J48) – Confidence Factor (CF):0.5, 1.0, 3.0 K-nearest neighbour (K-NN) –K:2, 5, 10	–	50,010,005,000	–
Asymmetric Method		5,001,000	50,010,005,000	–
Collective-based Method		5,001,000	–	– Number of Communities (K): 5,10 – Classification Threshold(Δ): 0.2,0.5,0.8 – Number of Iterations (M): 5,000 – Stop Condition (ϵ): 0.00001 – Learning Rate (η): 0.001

dating frequency (e.g., 500 and 1000 tweets). For our method, we experimented the unsupervised classification module under various parameters including the number of communities (K), classification threshold (Δ), number of iterations (M), stop condition (ϵ), and learning rate (η). In fact, the number of communities and the classification threshold are the most important parameters in this stage. Therefore, we studied their impact through setting $K \in \{5, 10\}$, and $\Delta \in \{0.2, 0.5, 0.8\}$, while the number of iterations, stop condition, and learning rate are fixed to 5,000, 0.00001, and 0.001, respectively. Table 4 summarizes the setup parameters of the three methods.

Training and Testing Tweets. The classical and the Asymmetric Self-learning (Chen, Zhang, Xiang et al., 2015) spam tweet detection methods require a pre-training before putting them in the operation mode. We have performed our experiments using 50 different hashtags tweets and for each hashtag, we devote an independent classification model, resulting in 50 classification models. When using the classical method, each classifier is trained for the first streamed tweets (e.g., 500, 1000, or 5000) into the intended hashtag, while the rest streamed tweets are used for the testing. The Asymmetric Self-learning method is pre-trained as in classical method, but periodic retraining is performed when a defined number of new streamed tweets (e.g., 500, or 1000) is satisfied. Thus, in some points, the testing tweets are leveraged later as training tweets when the periodic retraining condition is satisfied. Our method differs from classical and Asymmetric Self-learning methods where there is no need for a pre-training since we assume that the user of the system doesn't have time to build an annotated dataset. Hence, with excluding the initial training phase, our method has been experimented using the same circumstances of the Asymmetric Self-learning method.

6.2. Experimental results

The main purposes of our set of experiments are to study three main aspects, summarized in: (i) getting insight into the performance of our unsupervised collective-based method in automatically providing labeled datasets for the purpose of updating classification models, (ii) study the effect of changing number of training tweets, updating frequency, number of communities, and the classification threshold on having a generalized model with high detection rate, and (iii) examining how much the use of different learning algorithms with manipulating their main parameters in improving the performance. The experiments have been conducted on 50 different hashtags illustrated in Section 4. Since each hashtag represents a stream of tweets, we reported the performance results at different values of streamed tweets (e.g., every 500 tweets) where the ultimate value of the performance metrics is computed using the results of 50 hashtags. More precisely, the confusion matrix after a particular number of streamed tweets is summed over 50 hashtags, leading to having a single confusion matrix by which the metrics are computed.

At the evaluation level, we reported the experimental results of the classical (traditional ML which based on training the classifier on static dataset without retraining it later), asymmetric self-learning (Chen, Zhang, Xiang et al., 2015), and our collective-based method in *three* main figures (Figs. 8–10), which form a summary resulted from a set of experiments reported in Appendix A. Each figure aims at: (i) showing the effect of increasing dataset size on the performance of the three approaches in terms of recall, precision and F-measure, (ii) highlighting the effect of spam drift in the performance of tweet spam detectors, and (ii) comparing the performance of different ML methods. In addition, Each summarized figure shows the empirical results for the best machine learning algorithm drawn from all possible spam detection configurations. For illustrate, the asymmetric self-learning method has 9 possible configurations (3 updating frequency configurations $\times$ 3 different training tweet size configurations) where we reported the performance result of the best learning method tested under these configurations. For example, the curve labeled "asymmetric self-learning_J48(CF=5)_Freq=300_Training=500" (solid blue line) shown in Fig. 8 reports the performance in terms of recall for the asymmetric self-learning method when using the J48 learning method, 300 updating tweets frequency, and 500 tweets for training an initial version of the classification model. Through deeply analyzing the four main figures, some interesting thoughts and remarks can be inferred.

Classical (Traditional ML) method results. The accuracy results, which is not reported in this paper, have almost a stable performance with more than 90% when running the classical method at different learning algorithms and parameters. The three learning algorithms (Random Forest, J48, K-NN) have almost the same accuracy, without noticing any important effects when changing their parameters (#Trees, CF, and K) on improving the results. According to the distribution of the class labels in our dataset, it is important to mention that the accuracy metric is not too indicative one since the class label distribution is imbalanced. In other words, having a high accuracy near 90% doesn't mean that all spam tweets have been detected because in our dataset the distribution of spam class is about 11% compared to 89% for non-spam class. At the first glance, we can conclude that the classical method is effective for detecting spam tweets in real-time; however, the spam recall values which reported in Fig. 8 are too low with an average value of 35%. This means that classical-based classification models almost predict incoming tweets as a non-spam. Fig. 8 also shows the impact of increasing training size on improving spam recall. This behavior is expected since large enough training tweets may contain a diversity of social spammers' patterns, which help classification

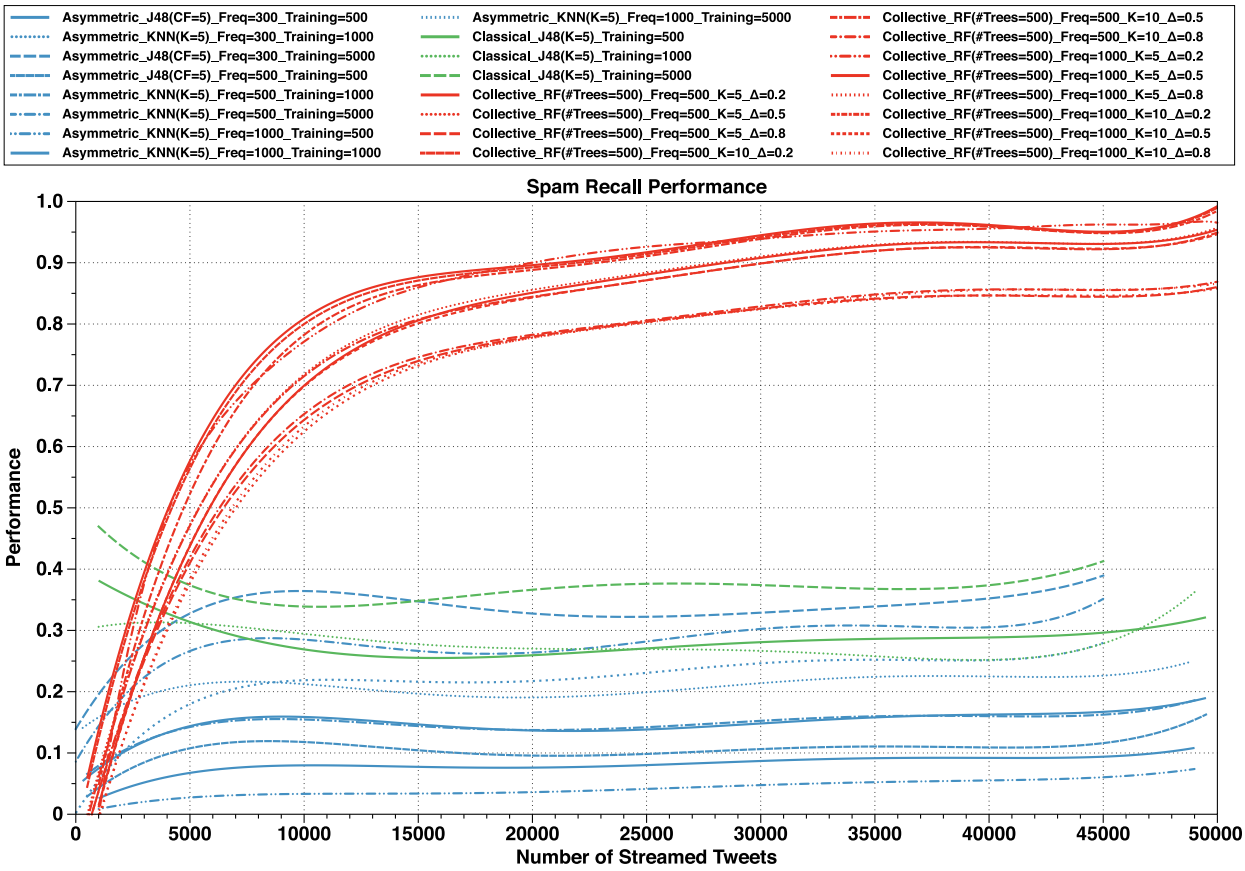


Fig. 8. Spam recall performance results of two baseline spam detection methods (Classical and asymmetric self-learning (Chen, Zhang, Xiang et al., 2015)), and our collective-based method, drawn for different method configuration parameter values and the best learning algorithm that performs well at those configurations.

models to detect them. Regarding learning algorithms, there is a clear diverse among them in which the decision tree (J48) learning method has a dominant performance, compared to K-NN and Random Forest learning methods. On the other side, Fig. 9, shows that the spam precision results of the classical method are better than spam recall ones, and showing also the direct correlation between the size of the training set and the spam precision performance metric. The high spam precision values ensure that the classical classification models classify the incoming tweets as spam when they have high confidence in that tweet being true spam. The decision tree (J48) has the lowest spam precision values, compared to the other learning algorithms. On the other hand, Random Forest has the best performance in terms of spam precision. The low spam precisions and high recall values obtained by J48 mean that the classification model (a sequence of if-else conditions) resulted by J48 have been designed through setting up the conditions that require any small clue to classify incoming tweet as a spam, while the Random Forest has established a group of decision trees made the conditions for classifying tweet as a spam too difficult. As the spam class F-measure metric is a combination of spam class precision and spam class recall metrics, as reported in Fig. 10, the value of the F-measure metric gets increased when increasing the number of the training tweets since the size of the training has a direct correlation with both the spam class recall and precision metrics.

Asymmetric self-learning (Chen, Zhang, Xiang et al., 2015). The impact of model update frequency and the training size parameters are obvious on increasing the spam recall values. Therefore, updating every streaming of 300 tweets achieves better spam recall values than doing that every 500 and 1,000. This behavior is reasonable and consistent because updating classification models as soon as possible makes them up-to-date to recent social spammers' patterns. Interestingly, although 300 tweets frequency is small enough to have an effective model, the increasing rate of spam class recall values is too small and near to zero. More precisely, the best and maximum spam class recall value has not exceeded 40%. This behavior can be explained by recalling the design of the method. The asymmetric self-learning method enriches the initial training set by adding incrementally every incoming tweet with labeling it based on the output of the current classification model. Therefore, the newly added tweets don't provide too much information about the patterns or behaviors related to social spammers since the built classification model predicts it as a spam tweet when the model has already learned over similar patterns or feature values. Consequently, with small improvement rates, the asymmetric self-learning method may need millions of tweets to obtain high spam recall values. On the other hand, the spam class precision results which presented in Fig. 9 have completely opposite behavior compared to the spam class recall values, through maintaining stable performance along the number of streamed tweets. Compared to the classical method, there are significant improvements in terms of precision occurred when retraining is carried out every either 500 or 1000 tweets. Thus, this proves the necessity of updating classification models to adopt social spammers' patterns and tricks. Consistently with the results of the asymmetric self-learning method, the Random Forest learning algorithm is dominant in the spam class precision, while the decision tree (J48) is the best in producing spam class recall values. For the spam class F-measure results reported in Fig. 10, they reflect the ineffective

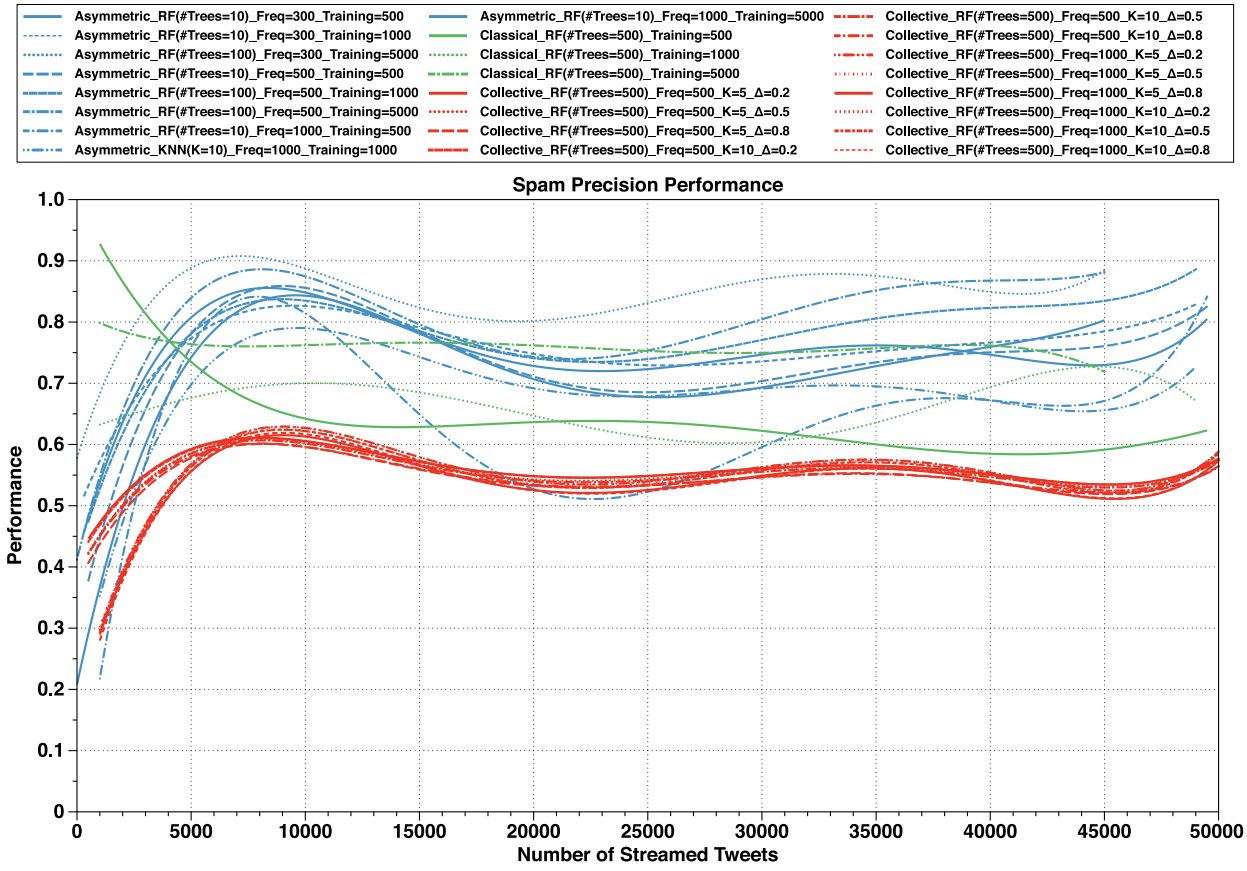


Fig. 9. Spam precision performance results of two baseline spam detection methods (Classical and asymmetric self-learning (Chen, Zhang, Xiang et al., 2015)), and our collective-based method, drawn for different method configuration parameter values and the best learning algorithm that performs well at those configurations.

of such a method for detecting tweets since the best maximum value that can be obtained is not exceeding 40%, while the classical method reaches around 50% at some configurations.

Collective-based Method Results. As reported in Fig. 8, our proposed method has high and superior spam class recall values with an average exceeding 80%, compared to the classical and asymmetric self-learning methods. This proves the effectiveness of the online unsupervised labeling method in producing automatically updated training datasets which can handle the dynamicity of social spammers on Twitter. With doing a pair-wise comparison along possible values of Δ, we find that the classification threshold Δ has a high direct correlation with the spam recall performance metrics. In other words, using small Δ values lead to having many spam tweets in the new training sets and thus learning over a diverse of social spammers' patterns. A large number of communities such as (K=10) does not provide too much contribution in detecting spam tweets. This behavior is because the number of uncorrelated spam campaigns that have attacked every hashtag is not more than 5. However, it is recommended to use a large number because the uncorrelated spam campaigns in the streamed tweets might increase with time. The tweets frequency has no significant impact on improving the spam recall values because, at each retraining phase, the training tweets have enough spam examples that make the classification models robust until the next retraining phase. It is important to mention that our collective-based method is not pre-trained such that before the first training phase, all streamed tweets are classified as non-spam ones. Indeed, this explains the behavior of having zero spam recall values at the beginning. The spam class precision values are quite low with an average performance of 55%. The main reason for having such spam class precision values is because of classifying non-spam communities as spam ones and thus the training sets will contain spam tweets which are truly non-spam ones. The results of the F-measure metric reported in Fig. 10 are quite stable along the number of streamed tweets and have almost similar behavior to the spam recall values. Furthermore, the impact of the number of communities and the number of tweets frequency is not clear in both the spam class precision and F-measure, leaving the full control to the classification threshold (Δ). Table 5 summaries the obtained results (average, max and the best achieved classifier) of the three approaches in terms of recall, precision and F-measure.

Time Performance Analysis. As the main purpose of our proposed system is to detect spam tweets in real-time, it is important to discuss in-depth the system performance in the detection time required, the CPU cost, and the needed resources. The recent statistics about the number of tweets show that every second, on average, around 6000 tweets are tweeted on Twitter, which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year.[3] In the streaming tweet mode, Twitter performs sampling and then pushes to the endpoint users only 1% of the instant tweets, having a maximum frequency of 60 tweets per second. This frequency imposes a constraint on our system to make a decision about every streamed tweet in margin no more than ($\frac{1}{60} \approx 16.7$ ms). Recalling that the component responsible about learning and updating new classifi-

[3] http://www.internetlivestats.com/twitter-statistics/.

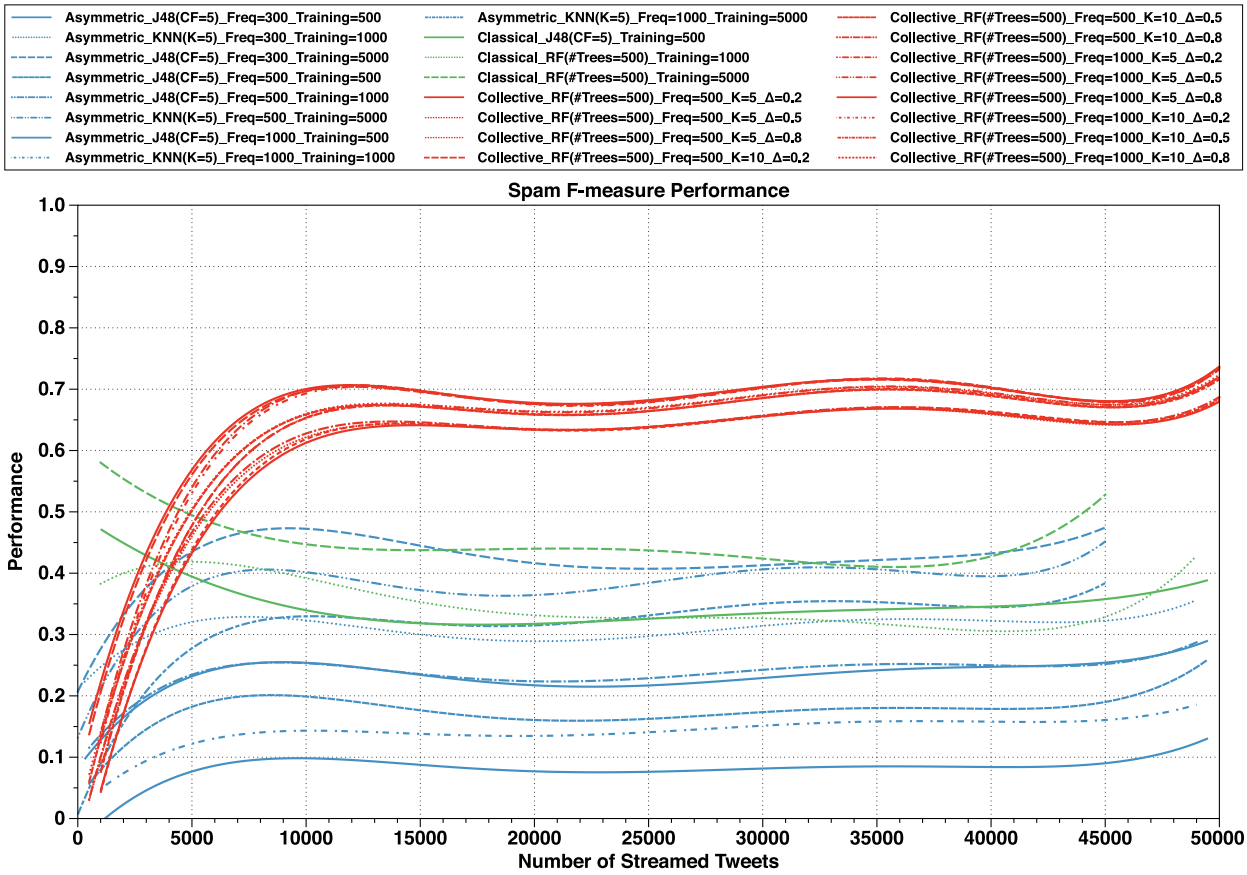


Fig. 10. Spam f-measure performance results of two baseline spam detection methods (Classical and asymmetric self-learning (Chen, Zhang, Xiang et al., 2015)), and our collective-based method, drawn for different method configuration parameter values and the best learning algorithm that performs well at those configurations.

Table 5
Summary of the obtained results in terms of recall, precision and *F*-measure.

	Recall			Precision			F-Measure		
Approach	Average	Max	ML	Average	Max	ML	Average	Max	ML
Classical (Traditional ML)	33%	41%	J48	71%	77%	RF	44%	52%	RF
Asymmetric Self-learning (Chen, Zhang, Xiang et al., 2015)	18%	38%	J48	78%	90%	RF	28%	47%	J48
Our Framework	91%	98%	RF	60%	63%	RF	70%	73%	RF

cation function is not required to work in real-time mode at all. In other words, that competent can be run in a background process which trains new classification model when the number of new streamed threshold condition is satisfied. The component of real-time spam tweet detection must treat every incoming tweet within a time less than 16.7 ms in order to prevent an overflow problem in buffering tweets and the synchronization problem. Hence, in our system, the adopted real-time tweet features, which are described in Table 1, require 3 ms for extraction and 2 ms in the prediction operation using the learned classification model, requiring 5 ms to process an incoming tweet. These values have been computed using a computer with i5 processor, a memory of 4GB, and storage of 1TB.

Discussion, implications and future work. The reported results in Figs. 8–10, shows that our proposed method has superior improvement specially in the recall measure with an average of 80% compared to 35% and 40% for classical and asymmetric self-learning method respectively. The low precision values obtained by our proposed system are not significant compared to the importance of obtaining high recall values. In fact, spam filtering in online social networks is conceptually different from the email spam filtering field. Therefore, classifying a truly non-spam email as spam one is a serious problem in email spam filtering, known as a false positive problem, since that email might be too important for the receiver. This situation is quite different in online social networks since in such a context the information circulated inside the networks are accessible by all registered users. In addition, due to a large number of tweets especially for trending topics, the users or applications (e.g. sentiment analysis and tweet summarization) are obviously interested in high quality and relevant information (non-spam tweets) more than low-quality ones (spam tweets). Moreover, classifying mistakenly a tweet as spam while it is truly non-spam is not a serious problem since another tweet can compensate the information within misclassified ones. Hence, maintaining high spam class recall values during the streaming is more important than having high precision values with low recalls.

Results (especially recall measure) corresponding to the classical method confirmed that increasing size of the training data only cannot bring more improvements to the detection model. In ad-

dition, updating dataset based on the output of a trained model cannot bring more improvements to the asymmetric self-learning method. We explain this in general by the fact that due to the new spamming activities and strategies the distribution of the extracted features changes during the time, while the distribution of training dataset features stays the same. In more details, with the introduced experimental results, comparisons and discussions, several implications and conclusions can be inferred: (i) the closeness performance of different learning algorithms proves that the major issue in spam detection is directly related to the selected features, (ii) due to the spam drift, the distribution of the features adopted in the literature for real-time spam tweet detection cannot be robust and hold for long time; (iii) the low spam recall values obtained by the classical and asymmetric self-learning classification models ensure the dynamicity of spam contents in Twitter and thus adopting these models are *not* an efficient solution at all, (iv) relying on increasing the size of the training dataset only cannot bring more improvements to the detection model, (v) the urgent need for an automatic online method to label new streamed tweets for the purpose of providing updated dataset, and (vi) retraining classification models periodically in batch mode using updated training dataset reduces the problem of features distribution making the framework capable of capturing new spamming behaviors and thus can reduce the spam drift problems.

Due to lack of time, many different improvements, adaptations, studies, and experiments have been left as future work which could be summarized in five dimensions: (i) introducing other tweet content features, (ii) study the effect of feature engineering methods, (iii) testing other clustering methods, (iv) reducing the effect of class imbalance dataset, and (v) handling the growth of the collected training dataset. For the first dimension, since spamming contents are usually similar with malicious topics or words, we intended to employ dynamic feature representations of the textual content of tweets such as Term Frequency-Inverse Document Frequency TF-IDF, bag-of-words, and sparse learning. In the second dimension, we will study the effect of using feature discretization and feature selection in terms of detection performance and time. In fact, discretization can be useful when creating probability mass/density functions and also many machine learning methods produce better results when discretizing continuous attributes (Kotsiantis & Kanellopoulos, 2005). On the other hand, features selection methods produce simplified models that have shorter training and operational time and also more general in order to reduce the problem of overfitting (Miao & Niu, 2016). For the third dimension, we can experiment other clustering algorithms like agglomerative clustering which is widely used in information retrieval. In addition, the community classification function (stage five in our framework) can be improved by making the annotation based on two stages. The first stage uses blacklisting method, and the second stage uses the community function defined in Eq. 14. The blacklist stage tests the URLs embedded in the message and the tweet will be predicted as spam if at least one of the embedded URLs are blacklisted, otherwise (e.g. tweets didn't contain URLs or all URLs are not blacklisted) the prediction will be based on the proposed community function. In the fourth dimension, since 11% of the collected tweets are spam and machine learning algorithms usually have better performance when classifying the majority class than the minority class, there is a need to handle the class imbalance dataset (Wu, Wen et al., 2017). Therefore, several techniques can be tested to reduce the effect of this problem including random sampling without replacement, random sampling with replacement, cluster sample, and stratified sample (Rout, Mishra, & Mallick, 2018). Finally, we will address an important design issue regarding the growth of the collected training dataset. As the effectiveness of very old spam contents will decrease in the long-term run, we will work on reducing the size of the collected data by dropping the very old tweets in order to quickly adapts classifiers to capture new spamming behaviors and to reduce training time.

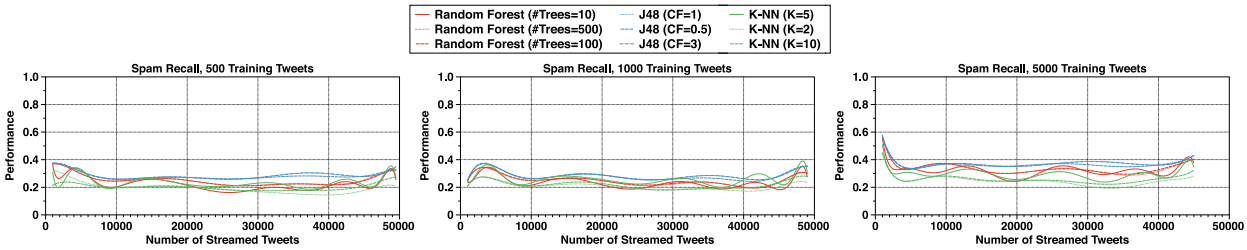
7. Conclusion

As spammers becoming more smarter and crafty through using complex spamming strategies, characteristics of the statistical properties of spam tweets keeps changing over time making the existing machine learning based detection method not an efficient solution. In this work, we have introduced a framework for dynamic retraining of supervised real-time tweet-level spam detection model to reduce the effect of the spam drift problem. The proposed framework composed of two main modules where the first one works in a batch mode and exploits the strength of the unsupervised learning method to periodically provide an up-to-date annotated datasets. The first module learned from unlabeled tweets through studying and analyzing the collective prescriptive of streamed tweets and their user's behavior. The second module has a real-time tweet-level classification model trained based on 17 lightweight features and retrained periodically using the up-to-date annotated dataset prepared by the first module. We have experimented our framework and other two related methods on our collected dataset which consists of more 2 million tweets annotated by the suspended account-based method. Results show that increasing only the size of the training data cannot bring more improvements to the spam classification model. In addition, our approach has a superior and a controllable spam recall performance, compared to the classical and asymmetric classification methods which in turn has a significant effect in reducing spam drift.

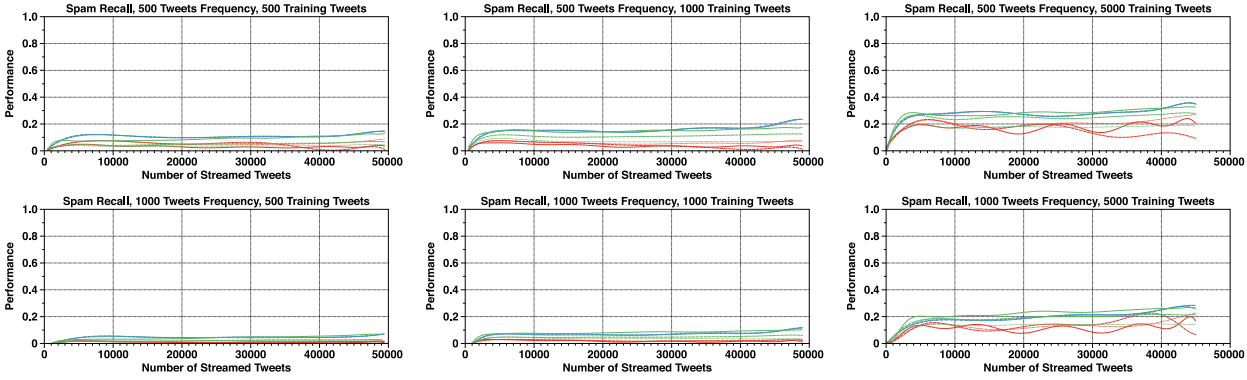
As a main strength point or contribution existing in our method, the proposed framework provides an online unsupervised learning method that does not require a human intervention in the way of periodically preparing annotated training datasets, which showing a significant difference from other real-time spam detection methods proposed in the literature for handling spam drift problem. In addition, our method provides a lightweight tweet-level spam detector that works on a real-time basis and updated itself periodically in batch mode where the proposed detector did not require a pre-information like a blacklist of spamming domains, initial annotated dataset, or pre-trained classifier. Moreover, the high recall values make our approach adoptable for Twitter-based researchers and industries to stream only high-quality tweets which required by a set of intelligent tweet-based application such as tweet sentiment analysis.

There is also a limitation in our proposed framework. The framework did not address the growth of the collected training dataset. In fact, the purpose of increasing the size of the training data is to eliminate the effect of spam drift problem by incorporating new spamming activities. However, the contribution of very old spam contents will decrease as the correlation of these contents becomes less with the new spam contents in the long-term run. Therefore, without efficient control to the growth, the classifier will not be adapted quickly to capture new spamming behaviors. In addition, classifiers will require more training time. Moreover, it may increase the side effect of class imbalance dataset. As future work, we will work on reducing the size of the collected data by dropping the very old tweets especially non-spam tweets after a certain time. Another limitation is regarding the annotation method used in our collected dataset in which suspended accounts may have non-spam tweets. However, there is no public dataset or common evaluation framework suitable to evaluate our framework and manually labeling large collection of tweets require a great effort and resources.

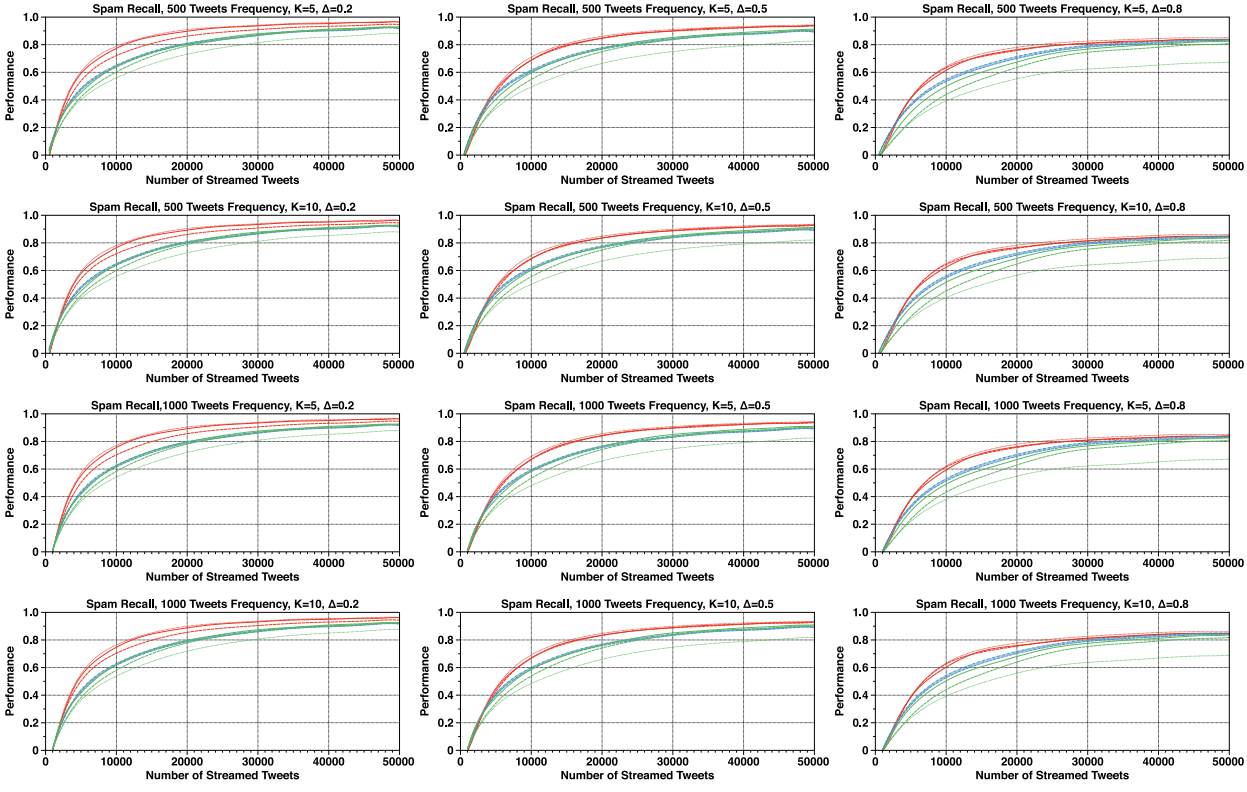
Appendix A. Advanced performance results



(a) Spam class recall performance results of the classical method, reported at different training set sizes, and various learning algorithms.

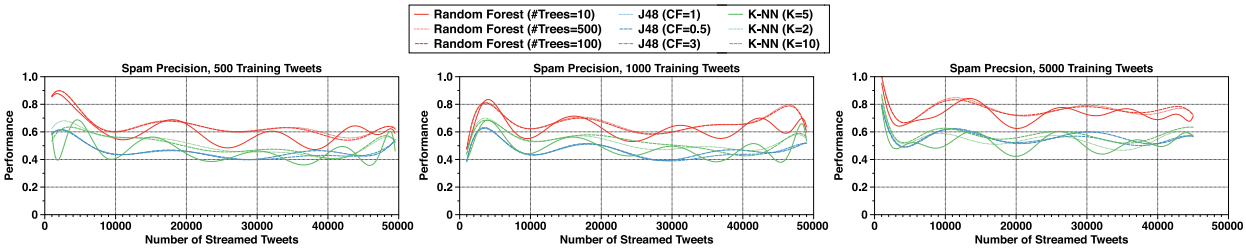


(b) Spam class recall performance results of the asymmetric method, reported at different training set sizes, two tweet frequencies, and various learning algorithms.

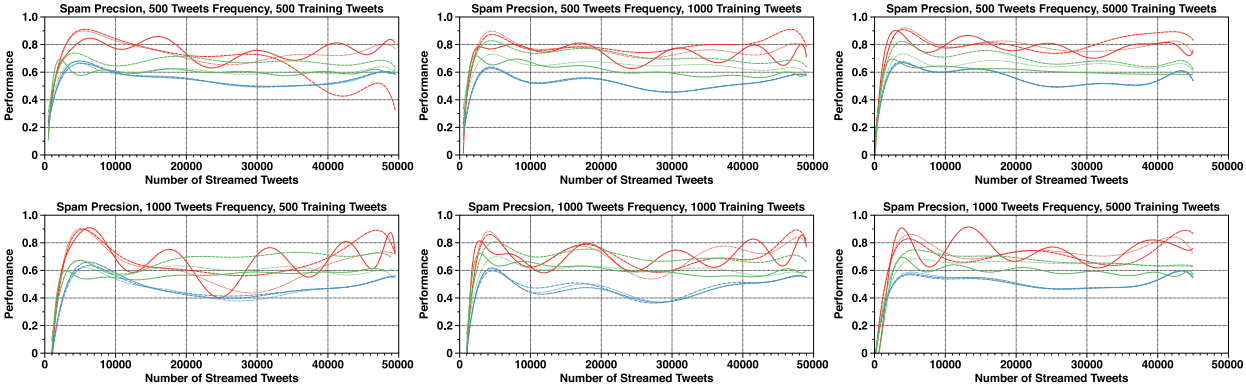


(c) Spam class recall performance results of our collective-based method, reported at two tweet frequencies, three classification thresholds, two numbers of communities, and various learning algorithms.

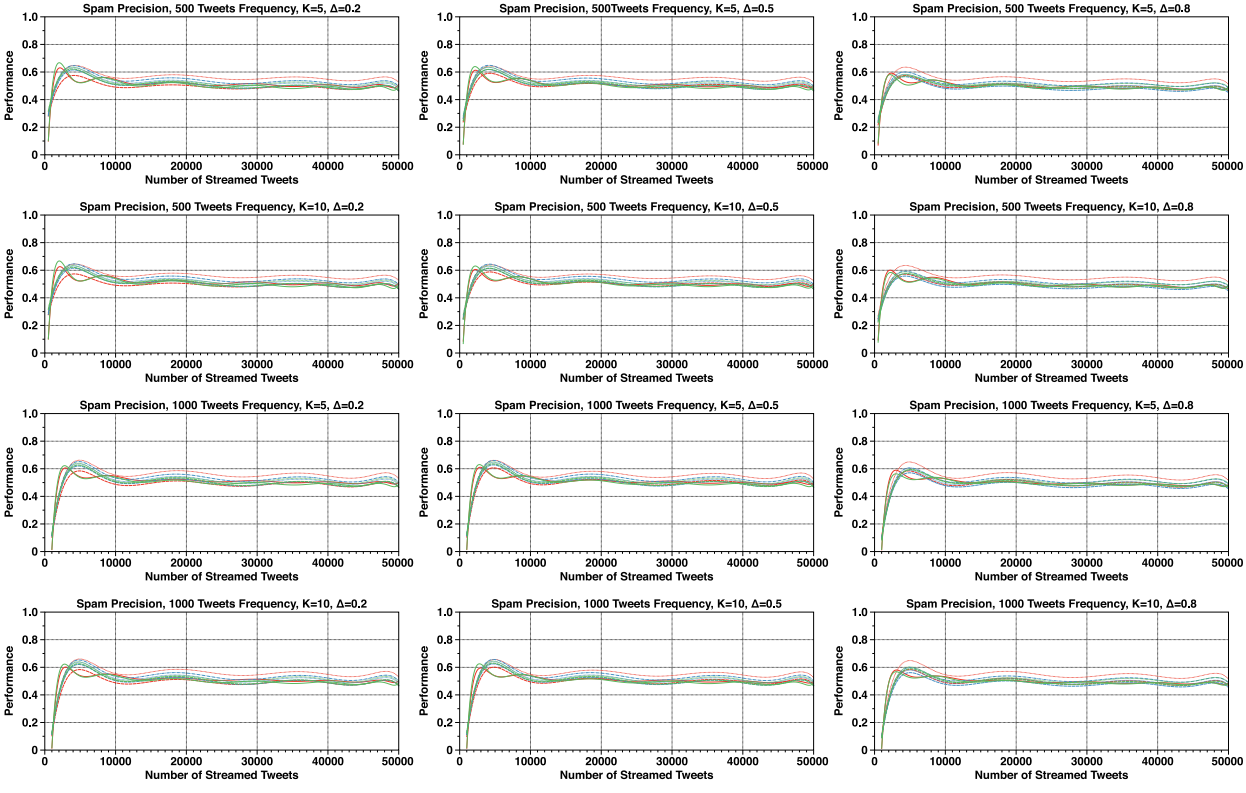
Fig. A1. Spam class recall performance results of the three real-time spam detection methods, including classical, asymmetric, and our collective-based methods, drawn at different configurations for each method.



(a) Spam class precision performance results of the classical method, reported at different training set sizes, and various learning algorithms.

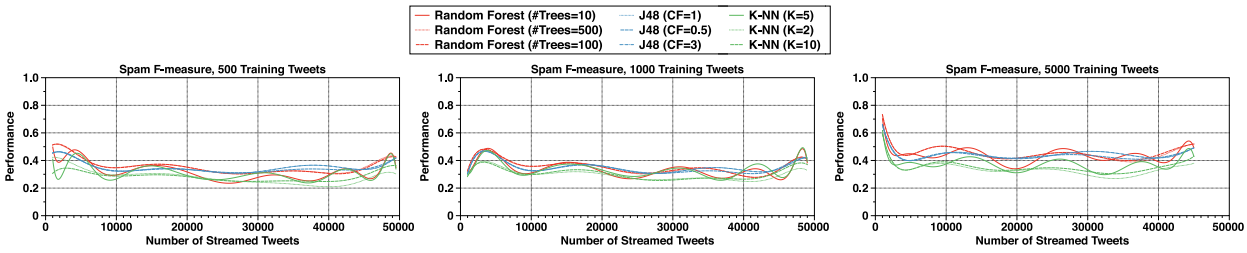


(b) Spam class precision performance results of the asymmetric method, reported at different training set sizes, two tweet frequencies, and various learning algorithms.

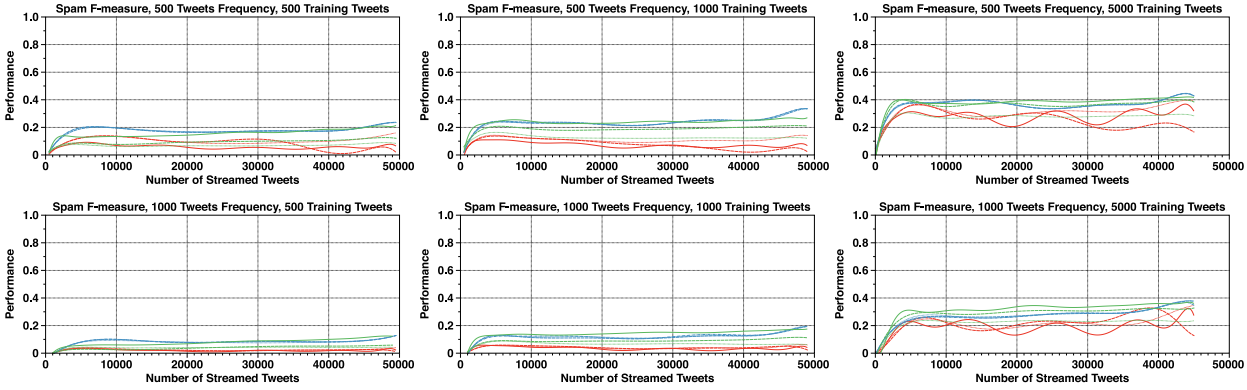


(c) Spam class precision performance results of our collective-based method, reported at two tweet frequencies, three classification thresholds, two numbers of communities, and various learning algorithms.

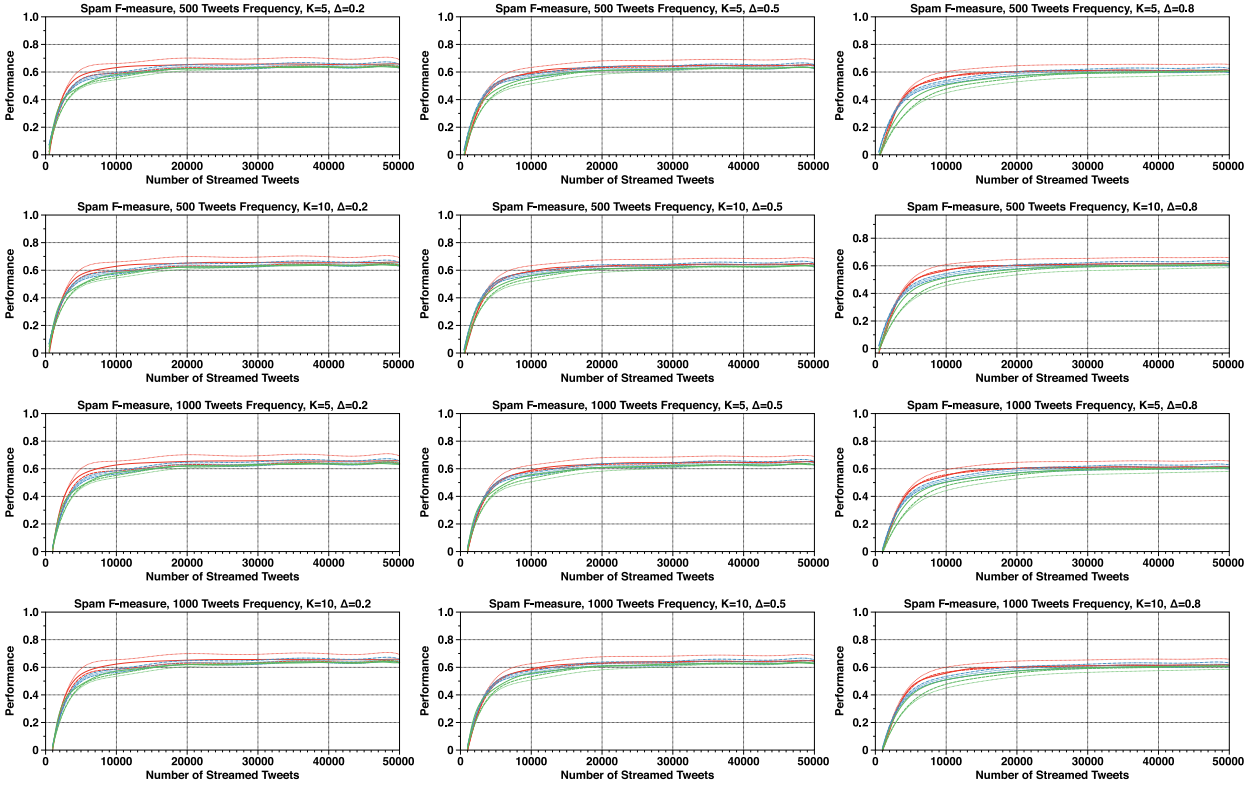
Fig. A2. Spam class precision performance results of the three real-time spam detection methods, including classical, asymmetric, and our collective-based methods, drawn at different configurations for each method.



(a) Spam class F-measure performance results of the classical method, reported at different training set sizes, and various learning algorithms.



(b) Spam class F-measure performance results of the asymmetric method, reported at different training set sizes, two tweet frequencies, and various learning algorithms.



(c) Spam class F-measure performance results of our collective-based method, reported at two tweet frequencies, three classification thresholds, two numbers of communities, and various learning algorithms.

Fig. A3. Spam class F-measure performance results of the three real-time spam detection methods, including classical, asymmetric, and our collective-based methods, drawn at different configurations for each method.

Conflict of interest

Authors declare that they have no conflict of interest.

Credit authorship contribution statement

Mahdi Washha: Conceptualization, Methodology, Software, Writing - review & editing, Supervision. **Aziz Qaroush:** Data curation, Writing - original draft. **Manel Mezghani:** Visualization, Investigation. **Florence Sedes:** Supervision.

References

Abascal-Mena, R., Lema, R., & Sèdes, F. (2015). Detecting sociosemantic communities by applying social network analysis in tweets. *Social Network Analysis Mining, 5*(1), 38:1–38:17. doi:10.1007/s13278-015-0280-2.

Agarwal, N., & Yiliyasi, Y. (2010). Information quality challenges in social media. In *International conference on information quality (ICIQ)* (pp. 234–248).

Almaatouq, A., Shmueli, E., Nouh, M., Alabdulkareem, A., Singh, V. K., Alsaleh, M., et al. (2016). If it looks like a spammer and behaves like a spammer, it must be a spammer: Analysis and detection of microblogging spam accounts. *International Journal of Information Security, 15*(5), 475–491.

Bara, I.-A., Fung, C. J., & Dinh, T. (2015). Enhancing twitter spam accounts discovery using cross-account pattern mining. In *Integrated network management (IM), 2015 IFIP/IEEE international symposium on* (pp. 491–496). IEEE.

Benevenuto, F., Magno, G., Rodrigues, T., & Almeida, V. (2010). Detecting spammers on twitter. In *In collaboration, electronic messaging, anti-abuse and spam conference (CEAS)* (p. 12).

Cao, C., & Caverlee, J. (2015). Detecting spam urls in social media via behavioral analysis. In *Advances in information retrieval* (pp. 703–714). Springer.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann Arbor MI, 48113*(2), 161–175.

Chellal, A., Boughanem, M., & Dousset, B. (2016). Multi-criterion real time tweet summarization based upon adaptive threshold. In *2016 IEEE/WIC/ACM international conference on web intelligence, WI 2016, Omaha, NE, USA, October 13–16, 2016* (pp. 264–271). doi:10.1109/WI.2016.0045.

Chen, C., Wang, Y., Zhang, J., Xiang, Y., Zhou, W., & Min, G. (2017). Statistical features-based real-time detection of drifted twitter spam. *IEEE Transactions on Information Forensics and Security, 12*(4), 914–925. doi:10.1109/TIFS.2016.2621888.

Chen, C., Zhang, J., Chen, X., Xiang, Y., & Zhou, W. (2015). 6 million spam tweets: A large ground truth for timely twitter spam detection. In *2015 IEEE international conference on communications (ICC)* (pp. 7065–7070). doi:10.1109/ICC.2015.7249453.

Chen, C., Zhang, J., Xiang, Y., & Zhou, W. (2015). Asymmetric self-learning for tackling twitter spam drift. In *Computer communications workshops (infocom wkshps), 2015 IEEE conference on* (pp. 208–213). IEEE.

Chen, C., Zhang, J., Xiang, Y., Zhou, W., & Oliver, J. (2016). Spammers are becoming "smarter" on twitter. *IT Professional, 18*(2), 66–70. doi:10.1109/MITP.2016.36.

Chen, C., Zhang, J., Xie, Y., Xiang, Y., Zhou, W., Hassan, M. M., et al. (2015). A performance evaluation of machine learning-based streaming spam tweets detection. *IEEE Transactions on Computational Social Systems, 2*(3), 65–76.

Chu, Z., Gianvecchio, S., Wang, H., & Jajodia, S. (2010). Who is tweeting on twitter: Human, bot, or cyborg? In *Proceedings of the 26th annual computer security applications conference* In *ACSAC'10* (pp. 21–30). New York, NY, USA: ACM. doi:10.1145/1920261.1920265.

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

Chu, Z., Widjaja, I., & Wang, H. (2012). Detecting social spam campaigns on twitter. In *Applied cryptography and network security* (pp. 455–472). Springer.

Giachanou, A., & Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys, 49*, 1–41. doi:10.1145/2938640.

Grier, C., Thomas, K., Paxson, V., & Zhang, M. (2010). @spam: The underground on 140 characters or less. In *Proceedings of the 17th ACM conference on computer and communications security*. In *CCS '10* (pp. 27–37). New York, NY, USA: ACM. doi:10.1145/1866307.1866311.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations Newsletter, 11*(1), 10–18. doi:10.1145/1656274.1656278.

Hoang, T. B. N., & Mothe, J. (2016). Building a knowledge base using microblogs: The case of cultural MicroBlog contextualization collection (regular paper). In K. Balog, L. Cappellato, N. Ferro, & C. Macdonald (Eds.), *Conference on multilingual and multimodal information access evaluation (CLEF), Evora, Portugal, 05/09/2016-08/09/2016: 1609* (pp. 1226–1237). http://CEUR-WS.org: CEUR Workshop Proceedings.

Hu, X., Tang, J., & Liu, H. (2014). Online social spammer detection.. In *AAAI* (pp. 59–65).

Hu, X., Tang, J., Zhang, Y., & Liu, H. (2013). Social spammer detection in microblogging.. In *IJCAI: 13* (pp. 2633–2639). Citeseer.

Imran, M., Castillo, C., Diaz, F., & Vieweg, S. (2015). Processing social media messages in mass emergency: A survey. *ACM Computing Survey, 47*(4), 67:1–67:38. doi:10.1145/2771588.

Inuwa-Dutse, I., Liptrott, M., & Korkontzelos, I. (2018). Detection of spam-posting accounts on twitter. *Neurocomputing, 315*. doi:10.1016/j.neucom.2018.07.044.

Kabakus, A. T., & Kara, R. (2017). A survey of spam detection methods on twitter. *International Journal of Advanced Computer Science and Applications, 8*.

Kotsiantis, S., & Kanellopoulos, D. (2005). Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering, 32*, 47–58.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics, 22*(1), 79–86.

Lee, K., Caverlee, J., & Webb, S. (2010). Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval*. In *SIGIR '10* (pp. 435–442). New York, NY, USA: ACM. doi:10.1145/1835449.1835522.

Lee, S., & Kim, J. (2012). Warningbird: Detecting suspicious urls in twitter stream.. In *NDSS: 12* (pp. 1–13).

Lee, S., & Kim, J. (2013). Warningbird: A near real-time detection system for suspicious urls in twitter stream. *IEEE Transactions on Dependable and Secure Computing, 10*(3), 183–195. doi:10.1109/TDSC.2013.3.

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). Learning to detect malicious urls. *ACM Transactions on Intelligent Systems and Technology, 2*(3), 30:1–30:24. doi:10.1145/1961189.1961202.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Martinez-Romo, J., & Araujo, L. (2013). Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications, 40*(8), 2992–3000.

McCord, M., & Chuah, M. (2011). Spam detection on twitter using traditional classifiers. In *Proceedings of the 8th international conference on autonomic and trusted computing*. In *ATC'11* (pp. 175–186). Springer-Verlag.

Meda, C., Ragusa, E., Gianoglio, C., Zunino, R., Ottaviano, A., Scillia, E., et al. (2016). Spam detection of twitter traffic: A framework based on random forests and non-uniform feature sampling. In *Advances in social networks analysis and mining (asonam), 2016 IEEE/ACM international conference on* (pp. 811–817). IEEE.

Mezghani, M., On-at, S., Péninou, A., Canut, M., Zayani, C. A., Amous, I., et al. (2015). A case study on the influence of the user profile enrichment on buzz propagation in social media: Experiments on delicious. In *New trends in databases and information systems - ADBIS 2015 short papers and workshops, bigdap, dcsa, gid, mebis, oais, sw4ch, wisard, poitiers, France, September 8–11, 2015. Proceedings* (pp. 567–577).

Mezghani, M., Zayani, C. A., Amous, I., Péninou, A., & Sèdes, F. (2014). Dynamic enrichment of social users' interests. In *IEEE 8th international conference on research challenges in information science, RCIS 2014, Marrakech, Morocco, May 28–30, 2014* (pp. 1–11). doi:10.1109/RCIS.2014.6861066.

Miao, J., & Niu, L. (2016). A survey on feature selection. *Procedia Computer Science, 91*, 919–926. doi:10.1016/j.procs.2016.07.111. Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016)

Mittal, A. (2011). *Stock prediction using twitter sentiment analysis*.

Oppenheim, A. V. (1999). *Discrete-time signal processing*. Pearson Education India.

Rout, N., Mishra, D., & Mallick, M. K. (2018). Handling imbalanced data: A survey. In M. S. Reddy, K. Viswanath, & S. P. K. M. (Eds.), *International proceedings on advances in soft computing, intelligent systems and applications* (pp. 431–443). Singapore: Springer Singapore.

Sedhai, S., & Sun, A. (2015). Hspam14: A collection of 14 million tweets for hashtag-oriented spam research. In *SIGIR 2015*.

Sedhai, S., & Sun, A. (2017a). An analysis of 14 million tweets on hashtag-oriented spamming. *Journal of the Association for Information Science and Technology, 68*(7), 1638–1651. doi:10.1002/asi.23836.

Sedhai, S., & Sun, A. (2017b). Semi-supervised spam detection in twitter stream. *IEEE Transactions on Computational Social Systems, PP*. doi:10.1109/TCSS.2017.2773581.

Stringhini, G., Kruegel, C., & Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference*. In *ACSAC '10* (pp. 1–9). New York, NY, USA: ACM. doi:10.1145/1920261.1920263.

Thomas, K., Grier, C., Ma, J., Paxson, V., & Song, D. (2011). Design and evaluation of a real-time url spam filtering service. In *2011 IEEE symposium on security and privacy* (pp. 447–462). doi:10.1109/SP.2011.25.

Thomas, K., Grier, C., Song, D. X., & Paxson, V. (2011). Suspended accounts in retrospect: An analysis of twitter spam. In *Internet measurement conference*.

Twitter (2016). The twitter rules. https://support.twitter.com/articles/18311#.[Online; accessed 1-March-2016].

Wang, A. H. (2010). Don't follow me: Spam detection in twitter. In *Security and cryptography (secrypt), proceedings of the 2010 international conference on* (pp. 1–10).

Wang, B., Zubiaga, A., Liakata, M., & Procter, R. (2015). Making the most of tweet-inherent features for social spam detection on twitter. *CoRR, abs/1503.07405*.

Wang, D., & Pu, C. (2015). Bean: a behavior analysis approach of url spam filtering in twitter. In *Information reuse and integration (IRI), 2015 IEEE international conference on* (pp. 403–410). IEEE.

Washha, M., Qaroush, A., Mezghani, M., & Sèdes, F. (2017a). Information quality in social networks: A collaborative method for detecting spam tweets in trending topics. In *Advances in artificial intelligence: From theory to practice - 30th international conference on industrial engineering and other applications of applied intelligent systems, IEA/AIE 2017, Arras, France, June 27–30, 2017, proceedings, part II* (pp. 211–223). doi:10.1007/978-3-319-60045-1_24.

Washha, M., Qaroush, A., Mezghani, M., & Sèdes, F. (2017b). Information quality in social networks: Predicting spammy naming patterns for retrieving twitter spam accounts. In *ICEIS 2017 - proceedings of the 19th international conference*

on enterprise information systems, volume 2, Porto, Portugal, 26–29 April, 2017 (pp. 610–622). SciTePress.

Washha, M., Qaroush, A., & Sèdes, F. (2016). Leveraging time for spammers detection on twitter. In *Proceedings of the 8th international conference on management of digital ecosystems* (pp. 109–116). ACM.

Washha, M., Shilleh, D., Ghawadrah, Y., Jazi, R., & Sèdes, F. (2017). Information quality in online social networks: A fast unsupervised social spam detection method for trending topics. In *ICEIS 2017 - proceedings of the 19th international conference on enterprise information systems, volume 2, Porto, Portugal, 26–29 April, 2017* (pp. 663–675). SciTePress.

Wu, T., Liu, S., Zhang, J., & Xiang, Y. (2017). Twitter spam detection based on deep learning. In *Proceedings of the Australasian computer science week multiconference* (p. 3). ACM.

Wu, T., Wen, S., Xiang, Y., & Zhou, W. (2017). Twitter spam detection: Survey of new approaches and comparative study. *Computers and Security, 76*. doi:10.1016/j.cose.2017.11.013.

Yang, C., Harkreader, R., & Gu, G. (2013). Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security, 8*(8), 1280–1293. doi:10.1109/TIFS.2013.2267732.

Yang, C., Harkreader, R., Zhang, J., Shin, S., & Gu, G. (2012). Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st international conference on world wide web*. In *WWW '12* (pp. 71–80). New York, NY, USA: ACM. doi:10.1145/2187836.2187847.

Yang, C., Harkreader, R. C., & Gu, G. (2011). Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers. In *Proceedings of the 14th international conference on recent advances in intrusion detection*. In *RAID'11* (pp. 318–337). Berlin, Heidelberg: Springer-Verlag.

Yang, J., & Leskovec, J. (2013). Overlapping community detection at scale: A non-negative matrix factorization approach. In *Proceedings of the sixth ACM international conference on web search and data mining*. In *WSDM '13* (pp. 587–596). New York, NY, USA: ACM. doi:10.1145/2433396.2433471.

Zubiaga, A., Spina, D., Amigó, E., & Gonzalo, J. (2012). Towards real-time summarization of scheduled events from twitter streams. In *23rd ACM conference on hypertext and social media, HT '12, Milwaukee, WI, USA, June 25–28, 2012* (pp. 319–320). doi:10.1145/2309996.2310053.

Zubiaga, A., Spina, D., Fresno, V., & Martínez-Unanue, R. (2011). Classifying trending topics: A typology of conversation triggers on twitter. In *Proceedings of the 20th ACM conference on information and knowledge management, CIKM 2011, Glasgow, United Kingdom, October 24–28, 2011* (pp. 2461–2464). doi:10.1145/2063576.2063992.