Information Quality in Online Social Networks: A Fast Unsupervised Social Spam Detection Method for Trending Topics

Mahdi Washha[1], Dania Shilleh[2], Yara Ghawadrah[2], Reem Jazi[2] and Florence Sedes[1]
[1]*IRIT Laboratory, University of Toulouse, Toulouse, France*
[2]*Department of Electrical and Computer Engineering, Birzeit University, Ramallah, Palestine*

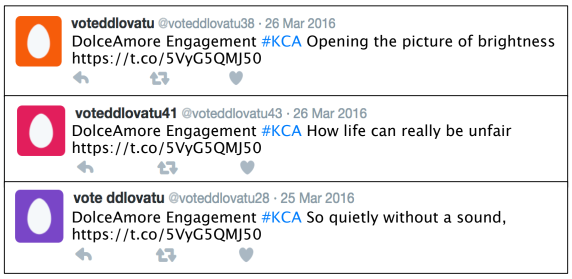
Keywords: Twitter, Social Networks, Spam.

Abstract: Online social networks (OSNs) provide data valuable for a tremendous range of applications such as search engines and recommendation systems. However, the easy-to-use interactive interfaces and the low barriers of publications have exposed various information quality (IQ) problems, decreasing the quality of user-generated content (UGC) in such networks. The existence of a particular kind of ill-intentioned users, so-called social spammers, imposes challenges to maintain an acceptable level of information quality. Social spammers simply misuse all services provided by social networks to post spam contents in an automated way. As a natural reaction, various detection methods have been designed, which inspect individual posts or accounts for the existence of spam. These methods have a major limitation in exploiting the supervised learning approach in which ground truth datasets are required at building model time. Moreover, the account-based detection methods are not practical for processing "crawled" large collections of social posts, requiring months to process such collections. Post-level detection methods also have another drawback in adapting the dynamic behavior of spammers robustly, because of the weakness of the features of this level in discriminating among spam and non-spam tweets. Hence, in this paper, we introduce a design of an unsupervised learning approach dedicated for detecting spam accounts (or users) existing in large collections of Twitter trending topics. More precisely, our method leverages the available simple meta-data about users and the published posts (tweets) related to a topic, as a collective heuristic information, to find any behavioral correlation among spam users acting as a spam campaign. Compared to the account-based spam detection methods, our experimental evaluation demonstrates the efficiency of predicting spam accounts (users) in terms of accuracy, precision, recall, and F-measure performance metrics.

1 INTRODUCTION

Online social networks (OSNs) have appeared, nowadays, as a powerful communication media in which users have the ability to share links, discuss and connect with each other. Such networks have attracted a tremendous amount of research interest and media (Nazir et al., 2008). One key feature of OSNs is their reliance on users as primary contributors in generating and publishing content, so-called as a user-generated content (UGC). The reliance on users' contributions might be leveraged in positive ways, including understanding users' needs for legal marketing purposes, studying users' opinions, and improving information retrieval algorithms. However, the easy-to-use interactive interfaces and the low barriers to publication characteristics have exposed information quality (IQ) problems (e.g., social spam, and information overload) (Agarwal and Yiliyasi, 2010), increasing the difficulty of obtaining accurate and relevant information. These characteristics have made OSNs vulnerable to various attacks by a particular kind of ill-intentioned users, so-called as social spammers, to post social spam contents. Social spammers post gibberish or non-sensical content in a particular context. For instance, posting a tweet talking about "how to lose your weight in five days." under the "#Trump" topic is a spam tweet because this tweet has no relation with the given topic at all. More generally, a wide range of goals drives social spammers to publish a spam content, summarized in (Benevenuto et al., 2010): (i) spreading advertisements to generate sales and gain illegal profits; (ii) disseminating porn materials; (iii) publishing viruses and malware; (iv) and creating phishing websites to reveal sensitive information.

Washha, M., Shilleh, D., Ghawadrah, Y., Jazi, R. and Sedes, F.
Information Quality in Online Social Networks: A Fast Unsupervised Social Spam Detection Method for Trending Topics.
DOI: 10.5220/0006372006630675
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 663-675
ISBN: 978-989-758-248-6
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved


voteddlovatu @voteddlovatu38 · 26 Mar 2016
DolceAmore Engagement #KCA Opening the picture of brightness https://t.co/5VyG5QMJ50

voteddlovatu41 @voteddlovatu43 · 26 Mar 2016
DolceAmore Engagement #KCA How life can really be unfair https://t.co/5VyG5QMJ50

vote ddlovatu @voteddlovatu28 · 25 Mar 2016
DolceAmore Engagement #KCA So quietly without a sound, https://t.co/5VyG5QMJ50

Figure 1: An example of three correlated spam tweets related to the same spam campaign.

Motivation and Problem. In the battle of detecting Twitter social spam, a considerable set of methods has been designed for detecting spam campaigns and individual spam accounts, with little efforts dedicated for individual spam tweets. These methods work by using the feature extraction concept combined with the supervised learning approach to build a predictive model based on a given annotated dataset (ground-truth). However, relying on the supervised learning approach in designing predictive models becomes less useful in detecting spam users or spam contents due to the fast evolution of social spammers (Hu et al., 2013; Washha et al., 2016). Also, social spammers adopt dynamic content patterns in OSNs as well as they change their spamming strategies to pretend as normal users. Consequently, the static anti-spamming methods have a considerable lag in adapting the fast evolution of social spammers' behaviors.

Beyond the obvious dynamicity of social spammers, the current spam detection methods of campaign-based and individual account-based are not suitable for "crawled" large-scale datasets of Twitter trending topics, requiring months to process such volume of data. Given the fact that the tweet object consists of simple meta-data (e.g. username, and creation date), these methods have been designed based on extracting advanced features (e.g. user's tweets timeline) requiring additional information from Twitter's servers. Twitter provides functions to retrieve further information (e.g., user's followers and followees) for a particular object (e.g., tweet, or user) using REST APIs[1]. However, Twitter constrains the number of API calls allowed to a defined time window (e.g., 40 calls in 15 minutes). For instance, retrieving information, including the meta-data of users' followers and followees, for half million users posted one million tweets may take about three months.

Collective Perspective. In manipulating spam accounts on OSNs, social spammers launch their spam campaigns (bots) through creating first thousands of spam accounts in an automated way. Then, spammers leverage the REST APIs provided by Twitter to coordinate their spam accounts automatically. For instance, spammers can automate the posting behavior of each account through tweeting at a fixed frequency. Moreover, REST APIs provide the current trending topics that are circulated among users, facilitating spammers' campaigns in attacking the trending topics by a spam content. Hence, given the fact that each spammer may employ thousands of spam accounts to spread a particular spam content, the probability is relatively high to find a correlation among spam tweets as well as their accounts. For example, Figure 1 shows three spam tweets posted by three different correlated spam accounts under the "#KCA" trending topic. The similarity in writing style of the three spam tweets, including the way of naming accounts, gives a strong indication that an individual social spammer controls these three spam accounts to act as a single spam campaign. Thus, shifting the perspective from inspecting individual tweets or accounts for the existence of spam to a collective one increases the effectiveness and the efficiency of detecting spam accounts in a fast way, especially when targeting large-scale collections.

Contributions. In this paper, we design an unsupervised method for filtering out social spam accounts (users) existing in large-scale datasets of Twitter trending topics. More precisely, we leverage *only* the available meta-data of accounts and tweets posted in trending topics, without retrieving any information from Twitter's servers. Our method detects spam accounts through passing tweets of a trending topic into four consecutive stages, distributed among clustering, community inference, feature extraction, and classification. We demonstrate the effectiveness of our unsupervised method through a series of experiments conducted on a crawled and an annotated dataset containing more than six million tweets belonging to 100 trending topics. The experimental results show that our method has superior performance in terms of four common and standard metrics (Accuracy, Precision, Recall, and F-measure), compared to supervised-based baseline methods. With the results obtained, our method might be leveraged in different directions:

- A wide range of OSNs (e.g., event detection, and tweet summarization) considers Twitter as a source of information and a field for performing their experiments in which Twitter trending topics datasets are adopted in a massive way. Hence, our method is suitable for these researches to increase the quality of collections in a fast and a practical way.
- Twitter can integrate our method with its anti-spam mechanism to detect spam campaigns in trending

[1]https://dev.twitter.com/rest/public

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

topics.

The remainder of the paper is organized as follows. Section 2 presents the Twitter's anti-spam mechanism as well as the Twitter social spam detection methods proposed in the literature. Section 3 presents notations, problem formalization, and our method design. Section 4 details a dataset used in experimenting and validating our approach. The experimental setup and a series of experiments evaluating the proposed approach are described in section 5. At last, section 6 concludes the work with giving directions as a future work.

2 BACKGROUND AND RELATED WORK

Social Spam Definition. Social spam is defined as a nonsensical or a gibberish text content appearing on OSNs and any website dealing with user-generated content such as chats and comments (Agarwal and Yiliyasi, 2010). Social spam may take tremendous range of forms, including profanity, insults, hate speeches, fraudulent reviews, personally identifiable information, fake friends, bulk messages, phishing and malicious links, and porn materials. One might view the social spam as an irrelevant information; however, this interpretation is quite not accurate. We justify this misinterpretation through the definition of information retrieval (IR) systems (Manning et al., 2008). The relevancy of documents in IR systems is dependent on the input search query. Thus, the irrelevant documents with respect to an input query are "not" necessary to be a spam content. Hence, the social spam can be further defined as irrelevant information that doesn't have an interpretation in any context as long as the input query is not a spam one.

Social Spammers' Principles. Social Spammers misuse all legal methods or services supported by social networks to spread their spam contents. Regardless the targeted social network, spammers adopt same principles in their goals and behaviors, summarized in (Washha et al., 2016):

- Social spammers are goal-oriented persons targeting to achieve unethical goals (e.g., promote products), and thus they use their smartness to accomplish their spamming tasks in an effective and a quick way.
- Social spammers often create and launch a campaign of spam accounts in a short period (e.g., one day), to maximize their monetary profits and speedup their spamming behavior.
- As a set of APIs is provided by social networks, spammers leverage them to automate their spamming tasks in a systematic way (e.g., tweeting every 10 minutes). More precisely, they avoid the random posting behavior because it may decrease the target profit and decelerate their spamming behavior.

Twitter's Anti-spam Mechanism. Twitter fights social spammers through allowing users to report spam accounts simply by clicking on "Report: they are posting spam" option available in the user's account page. When a user reports a particular account, Twitter's administrators manually review that account to make a suspension decision. However, adopting such a method for combating spammers needs great efforts from both users and administrators. Moreover, not all reports are trustworthy, meaning that some reported accounts might belong for legitimate users, not spammers. Besides this manual reporting mechanism, Twitter has defined some general rules (e.g., not allowed to post porn materials) for public to reduce the social spam problem as much as possible with suspending permanently the accounts that violate those rules (Twitter, 2016). However, Twitter's rules are easy to bypass by spammers. For instance, spammers may coordinate multiple accounts with distributing the desired workload among them to mislead the detection process. These accounts distributed tend to exhibit an invisible spam behavior. Consequently, these shortcomings have motivated researchers to propose more robust methods for the applications that use Twitter as an information source. Hence, we categorize the spam detection approaches dedicated for Twitter into two different types based on the automation detection level: (i) machine learning level as a fully automated approach; (ii) and social honeypot as a manual approach requiring human interactions.

Machine Learning Approach. In this approach, researchers have built their methods through employing three levels of detection, distributed between tweet-level detection, account-level detection, and campaign-level detection.

Tweet-Level. Martinez-Romo and Araujo (Martinez-Romo and Araujo, 2013) have identified the spam tweets by using probabilistic language models by which the topical divergence is measured for each tweet. Using statistical features as a representation for the tweet object, Benevenuto (Benevenuto et al., 2010) has identified spam tweet only by leveraging some features extracted from the tweet text such as the number of words and the number of characters. Then, the well-known SVM learning algorithm has been applied to a manually created dataset to learn a binary classifier. The main strength of this level is in having lightweight features suitable for real-time spam detection. However, adopting the supervised learning approach to have a static classification model is not a useful solution because of the high evolving and changing over time in the spam contents. In other words, hundred millions of ground truth spam tweets are required to have a robust model, which is not possible to have such a model.

Account-Level. The works introduced in (Wang, 2010; Benevenuto et al., 2010; McCord and Chuah, 2011; Stringhini et al., 2010; Washha et al., 2016) have turned the attention towards account-based features, including the number of friends, number of followers, similarity between tweets, and ratio of URLs in tweets. In more dedicated studies, the work proposed in (Cao and Caverlee, 2015) have identified the spam URLs through analyzing the shorten URLs behavior like the number of clicks. However, the ease of manipulation in this type of features by spammers has given a motivation to extract more complex features by using the graph theory. For instance, the studies presented in (Yang et al., 2011; Yang et al., 2012) have examined the relation among users using some graph metrics to measure three features, including the 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 Twitter-based applications because of the huge volume of data that must be retrieved from Twitter's servers.

Campaign-Level. Chu et al. (Chu et al., 2012b) have treated the spam problem from the collective perspective view. They have clustered a set of desired accounts according to the URLs available in the posted tweets, and then a defined set of features is extracted from the accounts clustered to be incorporated in identifying spam campaign using machine learning algorithms. Chu et al. (Chu et al., 2012a) have proposed a classification model to capture the difference among bot, human, and cyborg with considering the content of tweets, and the tweeting behavior. Indeed, the methods belonging to this detection level have a major drawback that the methods use features requiring a great number of REST API calls to obtain information like users' tweets and followers. Consequently, exploiting the current version of campaign level methods is not appropriate for filtering large-scale collections of tweets or users.

Beyond the features design level, the works introduced in (Hu et al., 2014; Hu et al., 2013) have proposed two optimization frameworks which use the content of tweets and basic network information to detect spam users using efficient online learning approach. However, the main limitations of such works are the need for information about the network from Twitter, raising the scalability problem again.

Honeypot Approach. Social honeypot is viewed as an information system resource that can monitor social spammers' behavior through logging their information such as the information of accounts and any available content (Lee et al., 2010). In fact, there is no significant difference between Twitter's anti-spam mechanism and the social honeypot approach. Both of them need an administrative control to produce a decision about the accounts that have fallen into the honeypot trap. The necessity of administrative control is to reduce the false positive rate, as an alternative solution to blindly classifying all users dropped in the trap as spam users.

3 THE PREDICTIVE MODEL

In this section, we introduce first definitions and notations used in modeling our solution. Then, we present the design of our method for detecting spam accounts existing in large-scale datasets of Twitter trending topics.

3.1 Terminology Definition and Problem Formalization

The concept of "Topic"[2] can be defined as a representation of hidden semantic structures in a text collection (e.g., textual documents, or tweets). "Trending Topic" is a word or phrase (e.g., #Trump, #KCA, and TopChef) that is mentioned at a greater rate than others. Trending topics become popular either because of a concerted effort by users, or due to an event that encourages users to talk about a particular topic. The main purpose of trending topics is to help users in understanding what is occurring in the world and what users' opinions are about it in a real-time manner. Trending topics are automatically identified by an algorithm that identifies topics that are massively circulated among users more than other topics.

As multiple users might do tweeting about similar topics, we model a particular trending topic, T, as a finite set of distinct users, defined as $Topic(U_T) = \{u_1, u_2, ...\}$, where the user (or account) element $u_\bullet$ is further defined by a quadruple-tuple of attributes, $u_\bullet = (UN, SN, UA, TS)$. Each attribute inside the user element is defined as follows:

Username (UN): Twitter allows users to name their accounts with maximum length of 20 characters.

[2]https://support.twitter.com/articles/101125#

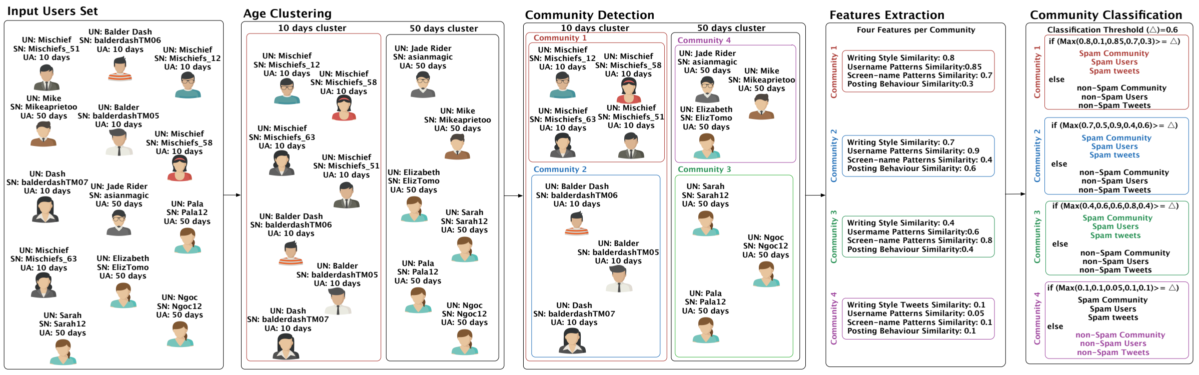

Input Users Set
Age Clustering
10 days cluster
50 days cluster
Community Detection
Community 1
Community 2
Community 3
Community 4
Features Extraction
Four Features per Community
Writing Style Similarity: 0.8
Username Patterns Similarity:0.85
Screen-name Patterns Similarity: 0.7
Posting Behaviour Similarity:0.3
Writing Style Similarity: 0.7
Username Patterns Similarity: 0.9
Screen-name Patterns Similarity: 0.4
Posting Behaviour Similarity: 0.6
Writing Style Similarity: 0.4
Usermame Patterns Similarity:0.6
Screen-name Patterns Similarity: 0.8
Posting Behaviour Similarity:0.4
Writing Style Tweets Similarity: 0.1
Username Patterns Similarity: 0.05
Screen-name Patterns Similarity: 0.1
Posting Behaviour Similarity: 0.1
Community Classification
Classification Threshold (△)=0.6
if (Max(0.8,0.1,0.85,0.7,0.3)>= △)
Spam Community
Spam Users
Spam tweets
else
non-Spam Community
non-Spam Users
non-Spam Tweets
if (Max(0.7,0.5,0.9,0.4,0.6)>= △)
if (Max(0.4,0.6,0.6,0.8,0.4)>= △)
if (Max(0.1,0.1,0.05,0.1,0.1)>= △)
UN: Mischief
SN: Mischiefs_51
UA: 10 days
UN: Balder Dash
SN: balderdashTM06
UN: Jade Rider
SN: asianmagic
UA: 50 days
UN: Mike
SN: Mikeaprietoo
UN: Elizabeth
SN: ElizTomo
UN: Sarah
SN: Sarah12
UN: Pala
SN: Pala12
UN: Ngoc
SN: Ngoc12
UN: Dash
SN: balderdashTM07
UN: Balder
SN: balderdashTM05

Figure 2: An example detailing the functionality of the four-stages, starting by a set of users posted tweets in a topic, and ending by the classification of the input users.

Users can use whitespaces, symbols, special characters, and numeric numbers in filling the username attribute. This field is not necessary for being unique and thus users can name their accounts by already used names. We model this attribute as a set of ordered characters, defined as $UN = \{(1,a_1),...,(i,a_i)\}$, where $i \in \mathbb{Z}_{\geq 0}$ is the character, $a_\bullet \in \{Printable\ Characters\}$[3], 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 not used previously by other users, with maximum length of 16 characters. Twitter also constrains the space of allowed characters to alphabetical letters, numbers, and "_" character. Similarly to the username attribute, we model this field as an ordered set of characters, defined as $SN = \{(1,a_1),...,(i,a_i)\}$, where $i \in \mathbb{Z}_{\geq 0}$ is the character, $a_\bullet \in \{Alphanumeric\ Characters\} \cup \{\_\}$, position inside the screen name 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 permissions to modify it in future. We exploit the creation date, as an accessible and available property in 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.

Tweets Set (TS): Each user $u_\bullet \in Topic(U_T)$ might post more than one tweet in the topic, T, where each tweet may describe user's thoughts, and interests. Thus, we model this field as a finite set of tweets, defined as $TS = \{TW_1,...,TW_n\}$, where n represents the number of tweets published by the user $u_\bullet$. Also, we model each tweet by double-tuple of attributes, $TW_\bullet = (Text, Time)$, where $Text = \{(1,w_1),(2,w_2),...\}$ is a finite set of string words, and $Time \in \mathbb{Z}_{\geq 0}$ is the publication date of the tweet in *minutes* time unit computed since January 1, 1970, 00:00:00 GMT.

Problem Formalization. Based on the above definitions and notations, we formlizae our problem in this work as follows. Given a set of users, U_T, such that each user has posted one or more tweets in a trending topic T. Our main problem is to predict the type (spam or non-spam) of each user in the given set U_T, without requiring any prior knowledge in advance such as the relation between users (e.g. followers and followees of users). More formally, we aim at designing a function y such that it processes and handles the given set of users, U_T, to predict the class label of considered users set, defined as $y : u_\bullet \rightarrow \{spam, non-spam\}, u_\bullet \in U_T$.

3.2 Four-stages Model Design

Leveraging Twitter REST APIs to retrieve more information about users is the optimal solution for filtering out accurately spam users (accounts). However, the impractically of this approach in regards of time brings serious challenges to design a method suitable for processing large-scale Twitter data-sets. Hence, beyond inspecting each user or tweet individually, we overcome these shortcomings through searching for a correlation among spam accounts at different levels and in an unsupervised way (i.e. no training phase required). We design four-stages method illustrated by an example in Figure 2. For a particular set of users posted tweets in a trending topic, the first stage performs clustering based on the age of users (accounts). In the next stage, for each cluster resulted, users' communities are detected through an optimization process. In the third stage, we extract *four*

[3]http://web.itu.edu.tr/sgunduz/courses/mikroisl/ascii.html

community-based features from users' attributes and tweets. The last stage exploits the value of each feature to make a decision (spammy or non-spammy) about each community identified using feature-based classification function, where each user of a spammy community will be directly labeled as "spam".

3.2.1 User Age-based Clustering

Social spammers have the ability to create hundreds and thousands of Twitter accounts in a short period not exceeding few days, for launching their spam campaigns. In such a case, the creation date of accounts may contribute in grouping and isolating the spam accounts that might have correlation between each other. Hence, we cluster the input users set based on the user age (UA) attribute. In a formal way, let $C_{age} = \{u | u \in U_T, u.UA = age\}$ be a cluster containing the users that have user age equaling $age \in Ages$ where $Ages = \{u.UA | u \in U_T\}$ is a set of distinct users ages. Obviously, the number of age clusters equals exactly to the size of the $Ages$ set (i.e. $|Ages|$).

3.2.2 Community Detection

Several uncorrelated spam campaigns might be created in the same time by spammers. Indeed, this means that we might have age clusters containing spam accounts belonging to different spam campaigns. Also, many non-spam users join Twitter daily, increasing the probability to have non-spam users created in the same day of spam ones. Hence, to distinguish among different uncorrelated spam campaigns and non-spam accounts, we perform community detection on each cluster resulted by Age-based clustering stage. More precisely, we view each spam campaign as a community having high correlation among its users. The correlation in a community might be at naming accounts level, duplicated tweets content, or similar posting behavior.

In this paper, we exploit the use of non-negative matrix factorization (NMF) method, as an unsupervised way, to infer communities' structure because of its outstanding performance in clustering (Yang and Leskovec, 2013). NMF works through partitioning an information matrix into hidden factor matrices for an age cluster, C_{age}, of users, defined mathematically as an optimization minimization problem:

$$\min_{H \geq 0} ||\mathbf{X} - \mathbf{H}\mathbf{H}^T||_F^2 \tag{1}$$

where $|| \bullet ||_F$ is the Frobenius norm of the considered matrix, $\mathbf{X} \in R^{|C_{age}| \times |C_{age}|}$ is an information matrix representing the strength of social connections between users, $\mathbf{H} \in R^{|C_{age}| \times K}$ is the community structure hidden factor matrix of K communities. The entry $X(i, j)$ reflects the strength of the social connection between the $u_i \in C_{age}$ user and $u_j \in C_{age}$ user. The entry $H(i, j)$ in the hidden factor matrix can be interpreted as the confidence degree of user $u_i \in C_{age}$ 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 **H** requires a formal definition of the information matrix **X**. For example, **X** might be an adjacency matrix representing the social connections or links among users of the given age cluster C_{age}. However, obtaining the adjacency matrix in our case is not possible because the available information about users are limited to simple meta-data describing accounts without providing information about the followers and followees. Hence, in this paper, we leverage the available and accessible information to estimate social connections among users through proposing two definitions of the information matrix **X** denoted as $\mathbf{X}^{SN}$, and $\mathbf{X}^{UN}$, where each of which is formally defined as follows:

Screen Name Similarity ($\mathbf{X}^{SN}$): As the screen name field must be unique, spammers tend to adopt a particular fixed pattern when creating multiple accounts to act as a spam campaigns. For instance, in Figure 1, the spammer has adopted the name "voteddlovatu" as a fixed pattern for the screen name field. Intuitively, the high overlapping or matching in the screen name among users 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 in the screen name attribute. More precisely, given two users $u_i, u_j \in C_{age}$, the degree of matching for a particular entry in the matrix $\mathbf{X}^{SN}$ is defined as:

$$\mathbf{X}^{SN}(i, j) = \frac{max\{|m| : m \in Patterns)\}}{min(|u_i.SN|, |u_j.SN|)}$$

$$Patterns = \bigcup_{N \in Max} N - gram(u_i.SN) \cap N - gram(u_j.SN) \tag{2}$$

where $| \bullet |$ is the cardinality of the considered set, $Max = \{1, ..., min(|u_i.SN|, |u_j.SN|)\}$ is a set consisting of positive integers representing the potential number of characters that have overlapping between the given names, $N - gram(\bullet)$ is a function that returns a set of contiguous sequence of characters for the given name (set of ordered characters) based on the value of N. For better understanding, the 3-gram (or tri-gram) of this screen name "vote" is $\{\{(1, v), (2, o), (3, t)\}, \{(1, o), (2, t), (3, e)\}\}$. 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 equation 2 is around $(\frac{4}{6})$66.6%, resulted from the use of pattern "vote", regardless the position of the pattern.

User Name Similarity ($\mathbf{X}^{UN}$): Differently from the screen name attribute, spammers may duplicate username attribute as many they wish. They exploit representative (not random) names to attract non-spam users. Therefore, the full or partial matching among users in such attribute increases the performance of community detection. We define the information matrix $\mathbf{X}^{UN}$ to measure the degree of similarity among users in the user name attribute. Formally, given two users $u_i, u_j \in C_{age}$, the degree of similarity is defined as:

$$\mathbf{X}^{UN}(i,j) = \frac{max\{|m| : m \in Patterns)\}}{min(|u_i.UN|, |u_j.UN|)}$$

$$Patterns = \bigcup_{N \in Max} N-gram(u_i.UN) \cap N-gram(u_j.UN) \tag{3}$$

where here $Max = \{1, ..., min(|u_i.UN|, |u_j.UN|)\}$.

Combining Information Matrices. With these two information matrices, NMF method allows to integrate them together in the same objective function. Thus, the new version of the objective function is defined as:

$$\min_{H \geq 0} ||\mathbf{X}^{SN} - \mathbf{H}\mathbf{H}^T||_F^2 + ||\mathbf{X}^{UN} - \mathbf{H}\mathbf{H}^T||_F^2 \tag{4}$$

Obviously, equation 4 infers the hidden factor matrix H to represent the consistent community structure of the users.

Optimization Method. The objective function is not jointly convex and no closed form solution exists. Hence, we propose the use of gradient descent as an alternative optimization approach. As 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 to the objective function given in equation 4 . At the iteration τ, the updating equation is given as:

$$\begin{aligned}\mathbf{H}^{\tau} &= \mathbf{H}^{\tau-1} - \eta . \frac{\partial \mathcal{L}(\mathbf{H}^{\tau-1})}{\partial(\mathbf{H})} \\ &= \mathbf{H}^{\tau-1} - 2\eta\big(6\mathbf{H}^{\tau-1}(\mathbf{H}^{\tau-1})^T\mathbf{H}^{\tau-1} \\ &\qquad - (\mathbf{X}^{SN} + \mathbf{X}^{UN})\mathbf{H}^{\tau-1} \\ &\qquad - ((\mathbf{X}^{SN})^T + (\mathbf{X}^{UN})^T)\mathbf{H}^{\tau-1}\big)\end{aligned} \tag{5}$$

where the parameter η denotes the gradient descent step in updating the matrix $\mathbf{H}$. We assign the value of η to a small constant value (i.e. 0.05). As the gradient descent method is an iterative process, a stop condition is required in such a case. Thus, we exploit two stop conditions: (i) the number of iterations, denoted as M; (ii) and the absolute change in the H matrix in two consecutive iterations to be less than a threshold, i.e. $|(||H^{\tau}||_F - ||H^{\tau-1}||_F)| \leq \varepsilon$.

3.2.3 Community-based Features Extraction

In order to classify each community, a set of features is required to be extracted for discriminating among spam or non-spam communities. No single feature is capable for discriminating effectively between spam and non-spam communities. Also, the target design of the features must maintain the strong condition that retrieving information about users from Twitter's servers is not allowed at all. Hence, we propose a design of four features that examine the collective perspective of users, with leveraging only the available information about users. Our features are distributed between tweet-based and user-based features, listed as: username patterns similarity (**UNPS**), scree-name patterns similarity (**SNPS**), tweets writing style similarity (**TsWSS**), and tweets posting behavior similarity (**TPBS**). We formalize the j^{th} community features as 6-tuple of attributes, $C_j = \{U, UNPS, SNPS, TsWSS, TPBS, L\}$ where U is a set of users belonging to the j^{th} community that can be extracted from $\mathbf{H}$ matrix, and $L \in \{spam, non-spam\}$ is the class label of the j^{th} community.

Username Patterns Similarity (UNPS) and Screen-name Patterns Similarity (SNPS): Spammers may adopt a pattern (e.g. "voteddlovatu") in creating their spam campaigns. Thus, when a community is biased toward a particular pattern used in creating accounts, that community has high probability to be a spam campaign and consequently all users inside that community are spam users (accounts). We model this spamming behavior through finding first all possible patterns used in naming accounts, extracted from username and screen name attributes. Then, we compute the probability distribution of patterns extracted from either username or screen name. At last, we compare the computed probability distribution of an attribute with the uniform distribution of the extracted patterns. Indeed, we hypothesize that the probability distribution of non-spam communities patterns must be close to the uniform distribution.

In a formal way, let PT^{UN} and PT^{SN} be two finite set of string patterns extracted from username and screen name attributes of j^{th} community users, respectively. Also, let P_D^{UN} and P_D^{SN} be the probability distributions of username and screen name attributes patterns, respectively. For the uniform distribution, let $P_{uniform}$ be a corresponding uniform distribution of the considered attribute patterns. For instance, for a particular community, let $PT^{SN} = \{"mischief", "isch", "\_12", "\_14"\}$, $P_D^{SN} = \{("mischief", 0.7), ("\_15", 0.1), ("\_14", 0.1), ("\_12", 0.1)\}$ be a set of screen name patterns with its probability distribution, the uniform probability distribution of these patterns is $\{("mischief", 0.25), ("\_15", 0.25), ("\_14", 0.25), ("\_12", 0.25)\}$.

To extract string patterns from username or screen name attribute, we apply the N-gram character method on user name or screen name in the inferred community. As spammers may define patterns varying in both length and position, to catch all or most potential patterns, we use different values of N ranging from three to the length of the name. We avoid $N \in \{1,2\}$ since it is meaningless to have one or two characters pattern. Formally, for a given name in form of Name = $\{(1,a_1),(2,a_2),...\}$, the potential patterns of which are extracted by

$$Patterns(Name) = \bigcup_{N \in Max} N-gram(Name) \qquad (6)$$

where $Max = \{3,...,\mid Name \mid\}$ is a finite set of possible values of N.

With the introduced definition, the string patterns sets of j^{th} community are given as:

$$\begin{aligned} PT^{UN} &= \bigcup_{u \in c_j \cdot U} Patterns(u \cdot UN) \\ PT^{SN} &= \bigcup_{u \in c_j \cdot U} Patterns(u \cdot SN) \end{aligned} \qquad (7)$$

We quantify the similarity between distributions through performing cross-correlation between the probability distribution of patterns set associated with a community and the corresponding uniform distribution of the considered patterns set. Formally, we compute the probability distribution similarity for username and screen name attributes through the following formulas:

$$UNPS(C_j) = 1 - \frac{Area(P_D^{UN} \star P_{uniform})}{Area(P_{uniform} \star P_{uniform})} \qquad (8)$$

$$SNPS(C_j) = 1 - \frac{Area(P_D^{SN} \star P_{uniform})}{Area(P_{uniform} \star P_{uniform})} \qquad (9)$$

where $Area(\bullet)$ is a function that computes the area under the new resulting distribution by the correlation operation. The key idea of performing autocorrelation between uniform probability distribution and itself is to normalize the area value that comes from cross-correlation operation, ranging the features between zero and 1. The values of these two features have direct correlation with the probability of being a spam community.

Tweets Writing Style Similarity (TsWSS): Each community being inferred has a set of tweets posted by its users. Given the fact that spammers automate their spam campaigns, the probability of finding a correlation in the writing style among the considered tweets is high. For instance, the spam tweets of a campaign in Figure 1 has a common style structure in writing tweets (word, word, hashtag, word, word,word, word, word, and then URL). In this instance, spammers had been tricky in writing tweets so that they avoid the duplication in text; however, they follow the same writing style.

We model this feature through transforming first the texts of tweets to a higher level of abstraction. Then, we measure the writing style similarity among tweets of the j^{th} community using Jaccard similarity index. A transformation function, $Type(ST) \in \{W,H,U,M\}$, is defined that takes ST string as a parameter and returns the type of the input string(**W**ord, **H**ashtag, **U**rl, and **M**ention).

Hence, for a tweet $TW_{\bullet}$, the new abstraction representation set is $Trans(TW_{\bullet}) = \{(i, Type(S)) | (i,S) \in TW_{\bullet}.Text\}$ where i is the position of the word string in the tweet text and S is a word string that requires transformation. With these definitions, we compute the writing style similarity among a set of tweets as follows:

$$TsWSS(C_j) = \frac{\sum_{T_1 \in Tweets_j} \sum_{T_2 \in Tweets_j} \frac{|Trans(T_1) \cap Trans(T_2)|}{|Trans(T_1) \cup Trans(T_2)|}}{(|Tweets_j|)(|Tweets_j|-1)} \qquad (10)$$

where $Tweets_j = \bigcup_{u \in C_j} u.TS$ is a unification of all tweets posted by the users of the j^{th} inferred community. The upper and lower values of this feature is one and zero, receptively. High value of this feature means that the probability of j^{th} community for being a spam is high because of the closeness of tweets in writing style.

Tweets Posting Behavior Similarity (TPBS): The posting behavior (e.g. positing every 5 min) of tweets at timing level might be an additional important clue in identifying spam communities. Thus, we propose a feature that measures the correlation among users' posting behavior. We model this behavior through computing first the posting time probability distribution of each user belonging to a particular community. Then, for each pair of users, we measure the similarity of their posting time probability distributions using cross-correlation concept, resulting a single real value ranging between 0 and 1. After that we compute the probability distribution of posting time similarity to compare it with a uniform distribution drawn on the interval [0,1]. More formally, let P_{TS}^{u} be a probability distribution of posting time of user's u tweets. P_{TS}^{u} can be drawn simply from

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

the tweets time of the user u who already posted them in the topic T. We compute the similarity between two posting time distributions of two different users belonging to $u_1, u_2 \in C^{j^{th}}$ community, as follows:

$$PostSim(u_1,u_2) = \frac{Area(P_{TS}^{u_1} \star P_{TS}^{u_2})}{Min(Area(P_{TS}^{u_1} \star P_{TS}^{u_1}), Area(P_{TS}^{u_2} \star P_{TS}^{u_2}))} \tag{11}$$

where $Area(\bullet)$ is a function that computes the area under the new resulting distribution, $Min(\bullet,\bullet)$ is a minimum operation that selects the minimum area. The key point of taking the min operation is to normalize the area that results from doing cross-correlation. The low value of *PostSim* means that the two input users have low correlation in posting time behavior.

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 quantity the different between the distributions through performing cross-correlation between them, defined as:

$$TPBS(C_j) = 1 - \frac{Area(P_{PostSim} \star P_{PostSim}^{Uniform})}{Area(P_{PostSim}^{Uniform} \star P_{PostSim}^{Uniform})} \tag{12}$$

where the high value (close to 1) of $TPBS$ means that all users of the j^{th} community have almost same posting behavior (i.e. almost same posting frequency) and thus that community has 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 low probability for being a spam campaign.

3.2.4 Classification Function

We leverage the four community-based features designed to predict the class label of each user posted tweet(s) in the topic T. Simply, we classify users of a community to spam if and only if one of the three features be greater than a particular threshold, named as Δ. The intuition behind this proposition is that the features designed are to detect spam communities, meaning that having at least one high feature value (i.e., $TsWss(C_j) \geq \Delta$ or $TPBS(C_j) \geq \Delta$ or $SNPS(C_j) \geq \Delta$ or $UNPS(C_j) \geq \Delta$) is enough to judge on the C_j community as spam. Otherwise, we label the C_j as non-spam. Once the C_j community is classified, the corresponding users will be given the same label of the community.

Table 1: Statistics of the annotated users (accounts) and tweets of 100 trending topics.

	Spam	non-Spam
Number of Tweets	763,555 (11.8%)	5,707,254 (88,2%)
Number of Users	185,843(8.9%)	1,902,288(91.1%)

4 DATA-SET DESCRIPTION AND GROUND TRUTH

The data-sets used at tweet level detection (Benevenuto et al., 2010; Martinez-Romo and Araujo, 2013) are not publicly available for research use. Also, for privacy reasons, social networks researchers provide only the interested in object IDs (e.g. tweets, accounts) to retrieve them from servers of the target social network. However, inspired by the nature of spam problem, providing IDs of spam tweets or accounts is not enough because Twitter might already have suspended the corresponding accounts.

Crawling Method. Hence, we exploit our research team crawler to collect accounts and tweets, launched since 1/Jan/2015. The streaming method is used to get an access for 1% of global tweets, as an unbiased crawling way. Such a method is commonly exploited in the literature to collect and create data-set in social networks researches.

Data-set Description. Using our team Twitter data-set, we clustered the collected tweets based on the topic available in the tweet with ignoring the tweets that do not contain any topic. Then, we selected the tweets of 100 trending topics (e.g. #Trump) randomly sampled to conduct our experiments.

Ground Truth Data-set. To evaluate the effectiveness of the spammy string patterns in retrieving spam accounts, we created an annotated data-set through labeling each account (user) as a spam or non-spam. However, with the huge amount of accounts, using manual annotation approach to have labeled data-sets is an impractical solution. Hence, we leverage a widely followed annotation process in the social spam detection researches. The process checks whether the user of each tweet was suspended by Twitter. In case of suspension, both the user with his tweets are labeled as a spam; otherwise we assign non-spam for both of them. In total, as reported in Table 1, we have found that more than 760,000 tweets were classified as spam, posted by almost 185,800 spam accounts.

Table 2: Performance results of baseline A and baseline B in terms of different metrics.

Learning Algorithm	Accuracy	Precision	Recall	F-Measure	Avg. Precision	Avg.Recall	Avg. F-Measure
Baseline (A): All Tweets Labeled as Non-Spam							
——————	91.1%	0.0%	0.0%	0.0.%	91.1%	91.1%	91.1%
Baseline (B): Supervised Machine Learning Approach							
Naive Bayes	81.2%	13.7%	10.5%	11.9%	79.0%	81.2%	80.1%
Random Forest (#Trees=100)	86.4%	13.2%	2.8%	4.6%	79.0%	86.4%	80.1%
Random Forest (#Trees=500)	86.5%	12.6%	2.6%	4.7%	79.4%	86.5%	82.8%
J48 (Confidence Factor=0.2)	86.4%	13.8%	2.9%	4.9%	79.6%	86.4%	82.5%
SVM (Gamma=0.5)	87.2%	15.7%	0.2%	0.4%	78.3%	87.2%	82.5%
SVM (Gamma=1.0)	87.0%	15.9%	0.1%	0.3%	77.9%	87.0%	82.2%

5 RESULTS AND EVALUATIONS

5.1 Experimental Setup

Performance Metrics. As the ground truth class label about each user is available, we exploit the accuracy, precision, recall, F-measure, average precision, average recall, and average F-measure, computed according to the confusion matrix of Weka tool (Hall et al., 2009), as commonly used metrics in classification problems. As our problem is two-class (binary) classification, we compute the precision, recall, and F-measure for the ”spam” class, while the average metrics combine both classes based on the fraction of each class (e.g., 4.9% * ”spam precision” + 95.1% * ”non-spam precision”).

Baselines. We define two baselines to compare our method with: (i) baseline ”A” which represents the results when classifying all users as non-spam directly without doing any kind of classification; (ii) baseline ”B” which reflects the results obtained when applying supervised machine learning algorithms on state of the art ”tweet” features described in (Benevenuto et al., 2010; McCord and Chuah, 2011; Martinez-Romo and Araujo, 2013) to use them in predicting the class label of users. It is important to mention that we adopt voting method to determine users' class labels. More precisely, for each user, we predict the class label of each tweet that the user had posted and then we perform voting on the tweet labels to compute the final class label of the considered user. In case of non-dominant class (e.g., 5 spam tweets and 5 non-spam tweets predicted) among user's tweets, we classify such user as spam one. As many learning algorithms are provided by Weka tool, we exploit Naive Bayes, Random Forest, J48, and support vector machine (SVM) as well-known supervised learning methods to evaluate the performance of the mentioned state of the art features.

Baselines Methods Settings. For the Naive Bayes method, we set the ”useKernelEstimator” and ”useSupervisedDiscretization” options to false value as default values set by Weka. For Random Forest, we set the option max depth to 0 (unlimited), with studying the effect of changing number of trees $\in \{100, 500\}$. For $J48$ method, we set the minimum number of instances per leaf to 2, number of folds to 3, and confidence factor to 0.2. For the SVM method, we use the LibSVM (Chang and Lin, 2011) implementation integrated with Weka tool with setting the kernel function to Radial Basis and examining the impact of gamma $\in \{0.5, 1\}$, where the rest parameters are set to the default ones.

Our Method Settings. For community detection stage, we set η = 0.001, M = 10,000, and $\varepsilon = 0.0001$ as values for the learning rate, number of iterations, and the threshold of absolute change in the hidden matrix H, respectively. For the number of communities K, we experiment our method at three different values, $K \in \{5, 10\}$, to study its effect. For the size of information matrices $\mathbf{X}$, we consider all distinct users (accounts) of each hashtag without excluding any user available in the testing collection. As an iterative algorithm is used for solving the community detection optimization problem, we initialize each entry of the hidden matrix $\mathbf{H}$ by a small positive real value drawn from a uniform distribution on the interval $[0, 1]$. For the threshold Δ, we study the impact of changing its value through performing experiments at different values of $\Delta \in [0.1, 1.0]$ with 0.1 increment step.

Experiment Procedure. For each trending topic in our data-set, we perform the following steps: (i) we extract and select randomly the users who posted tweets related to the considered topic; (ii) we cluster the users extracted based on their account's ages; (iii) then, for each cluster, we apply the community detection method on the extracted users set using a predfined number of communities K; (iv) afterward each community is labeled as spam and non-spam according to the designed classification function using a particular threshold (Δ), applied on the degree of similarity UN, degree of similarity SN, tweets writing style similarity, and posting behavior similarity; (vi) as the ground truth class label about each user, we

Table 3: Our method performance results in terms of various metrics, computed at different values of number of communities $K \in \{5, 10\}$, and at various classification threshold $\Delta \in \{0.1, 1\}$.

Model (Δ)	**Accuracy**	**Spam Precision**	**Spam Recall**	**Spam F-measure**	**Avg. Precision**	**Avg. Recall**	**Avg. F-measure**
Number of Communities ($K = 5$)							
Δ=0.1	63.0%	14.6%	**65.8%**	23.9%	**87.9%**	63.0%	73.4%
Δ=0.2	66.3%	15.4%	63.0%	24.8%	87.8%	66.3%	75.6%
Δ=0.3	68.1%	15.9%	60.7%	**25.2%**	87.8%	68.1%	76.7%
Δ=0.4	69.7%	16.1%	57.5%	25.1%	87.5%	69.7%	76.7%
Δ=0.5	77.4%	18.2%	44.3%	25.8%	87.0%	77.4%	81.9%
Δ=0.6	78.2%	17.7%	40.3%	24.6%	86.7%	78.2%	82.2%
Δ=0.7	82.0%	18.6%	30.9%	23.3%	86.3%	82.0%	84.0%
Δ=0.8	85.6%	19.7%	20.2%	19.9%	85.8%	85.6%	85.7%
Δ=0.9	89.1%	**20.3%**	7.6%	11.1%	85.2%	89.1%	**87.1%**
Δ=1.0	**91.1%**	0.0%	0.0%	0.0%	83.1%	**91.1%**	86.9%
Number of Communities ($K = 10$)							
Δ=0.1	69.0%	15.9%	**58.6%**	25.1%	87.6%	69.0%	77.2%
Δ=0.2	71.5%	16.7%	56.0%	25.8%	**87.6%**	71.5%	78.8%
Δ=0.3	73.3%	17.2%	53.3%	**26.0%**	87.5%	73.3%	79.7%
Δ=0.4	75.1%	17.5%	49.0%	25.8%	87.2%	75.1%	80.7%
Δ=0.5	81.3%	19.5%	35.7%	25.2%	86.7%	81.3%	83.9%
Δ=0.6	82.1%	19.0%	31.5%	23.7%	86.3%	82.1%	84.2%
Δ=0.7	85.1%	19.8%	22.3%	21.0%	85.9%	85.1%	85.5%
Δ=0.8	87.7%	**20.7%**	13.5%	16.3%	85.6%	87.7%	86.6%
Δ=0.9	90.0%	20.7%	4.0%	6.6%	85.0%	90.0%	**87.4%**
Δ=1.0	**91.1%**	0.0%	0.0%	0.0%	83.1%	**91.1%**	86.9%

compute the accuracy, spam precision, spam recall, spam F-measure, average precision, and average recall, computed according to the confusion matrix of Weka tool.

Obviously, each topic produces a confusion matrix. Thus, in computing the final performance metrics, we sum the confusion matrices of the 100 trending topics to have an one single confusion matrix by which the proposed metrics are computed.

5.2 Experimental Results

Baseline Results. We report the performance results of the baseline methods in Table 2. Based on the values of recall metric (4th column), the supervised classification models have a strong failure in filtering out the spam users that exist in the 100 trending topics, where the high value is obtained by *NaiveBayes* learning algorithm. The 10.5% of spam recall obtained by *NaiveBayes* means that less than 19,000 of spam accounts can be detected from more than 185,000 spam accounts existing in our data-set. The low spam precision values also give an indication that a significant number of non-spam accounts has been classified into spam ones. Subsequently, as spam F-measure is dependent on recall and precision metrics, the values of spam F-measure are definitely low. The accuracy values of baseline "B" are relatively close to the accuracy of baseline method "A". However, given the low values of spam precision and spam recall, the accuracy metric in this case is not too much an indicative and useful metric to judge on the supervised learning as a winner approach. More precisely, the supervised learning approach does not add significant contributions in increasing the quality of the 100 trending topics. The key idea of using different machine learning algorithms with playing in their parameters is to highlight the badness of the state-of-the-art tweet features as simple ones to judge on spam users. Overall, the results obtained by the learning models draw various conclusions: (i) the state-of-the-art features are not discriminative among spam and non-spam, ensuring the dynamicity of spam contents in OSNs; (ii) spammers tend to publish tweets pretending as non-spam ones; (iii) adopting a supervised approach to perform training on an annotated data-set of trending topics and applying the classification model on future or not annotated trending topics are *not* the solution at all.

Δ's Effect. Taking a look at our method performance results in Table 3, the behavior is completely different in recalling (classifying) spam accounts, especially when the value of Δ gets lower. The recall results are completely consistent with the equation 14 designed for classifying users. For low values of Δ, there are some non-spam communities classified as spam. Indeed, this explains the dramatic degradation in the accuracy when decreasing the value of Δ, as well as the degradation in the spam precision. Although of high recall values, the spam precision values of our method are almost similar to the supervised learning approach ones.

***K*'s Impact.** Number of communities, K, has direct and obvious impact on the accuracy, spam precision, and spam recall metrics. Indeed, the accuracy

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

and spam precision increase as long as the number of communities increases. The justification for this behavior is that the experimented 100 trending topics had been attacked by many uncorrelated spam campaigns. Subsequently, increasing the number of communities allows to detect spam campaigns precisely and accurately. At spam recall level, the behavior inversely correlates with the number of communities. Using higher number of communities than the real (unknown) number of spam campaigns leads to separate those campaigns on more communities. As spam communities might contain non-spam users (accounts), in such a case, the values of features decrease, classifying those communities as ”non-spam”.

False Positive v.s. High Quality. In email spam filtering, great efforts are directed toward the false positive problem that occurs when a truly ”non-spam” email is classified as ”spam”. However, in the context of social spam, the false positive problem is less important because of the availability of large-scale data collections, meaning that classifying non-spam user as spam is not a serious problem. Thus, the attention is turned in OSNs context to increase the quality of data where a wide range of Twitter-based applications (e.g. tweet summarization) has high priority to work on noise free collections. Also, the computational time aspect is significant when targeting large-scale collections. Hence, our method is completely suitable to process large-scale collections with providing high quality collections. For instance, the time required to process our Twitter data-set is no more than one day. At last, as various experiments are given for different Δ values where no optimal value can satisfy all performance metrics, the selection is mainly dependent on the desired requirements of the final collection. For instance, low Δ value is recommended to have too high quality collection with having high probability to lose not noisy information.

6 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we have designed an unsupervised approach for filtering out spam users (accounts) existing in large-scale collections of trending topics. Our method takes the collective perspective in detecting spam users through discovering the correlations among them. Our work brings two additional benefits to the information quality field: (i) filtering out spam users without needing for annotated datasets; (ii) and performing the filtration process in a fast way because of the dependency on the available meta-data only, without needing for retrieving information from the Twitter’s servers. With this new idea, we plan as a future work to study the impact of performing collaboration with other social networks to improve the current results. Also, we intend to design more collective-based robust features such as the sentiment of tweets.

REFERENCES

Agarwal, N. and Yiliyasi, Y. (2010). Information quality challenges in social media. In *International Conference on Information Quality (ICIQ)*, pages 234–248.

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. (2010). Detecting spammers on twitter. In *In Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, page 12.

Cao, C. and Caverlee, J. (2015). Detecting spam urls in social media via behavioral analysis. In *Advances in Information Retrieval*, pages 703–714. Springer.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012a). 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., and Wang, H. (2012b). Detecting social spam campaigns on twitter. In *Applied Cryptography and Network Security*, pages 455–472. Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

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

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

Lee, K., Caverlee, J., and 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*, SIGIR ’10, pages 435–442, New York, NY, USA. ACM.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Martinez-Romo, J. and 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. and Chuah, M. (2011). Spam detection on twitter using traditional classifiers. In *Proceedings of the 8th International Conference on Autonomic and Trusted Computing*, ATC’11, pages 175–186. Springer-Verlag.

Nazir, A., Raza, S., and Chuah, C.-N. (2008). Unveiling facebook: a measurement study of social network based applications. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 43–56. ACM.

Stringhini, G., Kruegel, C., and Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA. ACM.

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*, pages 1–10.

Washha, M., Qaroush, A., and Sedes, F. (2016). Leveraging time for spammers detection on twitter. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems, MEDES 2016, Biarritz, France, November 1-4, 2016*, pages 109–116.

Yang, C., Harkreader, R., Zhang, J., Shin, S., and 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*, WWW '12, pages 71–80, New York, NY, USA. ACM.

Yang, C., Harkreader, R. C., and 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*, RAID'11, pages 318–337, Berlin, Heidelberg. Springer-Verlag.

Yang, J. and Leskovec, J. (2013). Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 587–596, New York, NY, USA. ACM.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS