You Shall Not Pass: Detecting Malicious Users at Registration Time

Christian Kater
Leibniz University Hannover
ch.kater@gmail.com

Robert Jäschke
Leibniz University Hannover
jaeschke@L3S.de

ABSTRACT

Spam is a widespread problem for many online services. The use case in this paper is the social bookmarking system BibSonomy, which received over 150 times more registrations from spam users than from normal users over the last ten years.

A common approach to fight spam is to use machine learning to classify the users into good or malicious users. Based on information the users provide to the service in form of profile information or posts, features are created from which a classifier can make its decision. However, this often means that the accounts of the spam users are already active and can post their spam. In this work we propose an approach for deciding at registration time whether a user is malicious or not. In order to achieve this goal, we extracted 177 features from the information the users provide during the registration process, their IP address, and registration time. With these features we used state-of-the-art classifiers to identify users as spammers or regular users. With the best classifier, we could reach an AUC of 0.912.

CCS Concepts

•Information systems → Spam detection;

Keywords

Spam Detection; Social Bookmarking

1. INTRODUCTION

Social bookmarking services are web-based systems that allow their users to manage their bookmarks (web links) online. One of the established systems is BibSonomy[1] which since 2005 is developed and operated by the team of the second author [2]. Unfortunately, such systems also attract users that post links to – typically commercial or dubious – web sites with the sole purpose to increase the visibility and number of visitors of these web pages. This kind of posts is considered to be *web spam* [5]. The goal of the operators of such systems is to avoid that such content is posted or visible, since it reduces the usability and the attractiveness of the system for regular users. More generally, content that violates the terms and conditions of the corresponding system should be automatically identified and removed. From our experience, web spam is the most frequent kind of such content. Therefore, it is essential to implement measures that reduce, remove, or avoid web spam. The main challenge is to devise approaches that are capable of distinguishing between good content and web spam, or – as we aim in this paper – between good users and fraudulent users. Detecting spam posts is hard, since it is difficult to precisely define what spam is and therefore to decide whether a web page or a post linking to it is spam or not. Detecting spam *users* is even more challenging, since they might hide their spam posts among other posts.

Since 2008 we have developed a spam detection framework for BibSonomy which uses machine learning technologies to automatically classify users based on the content they post [6]. As soon as a new user has added a post to BibSonomy, the framework classifies him/her based on his/her post(s). Posts of users which are classified as spammers are then hidden and the access of the users to the system is restricted. The classifier is regularly trained on a large corpus of manually annotated users, which is extended by manual checks and interventions that are performed by the administrators.

The drawback of the existing approach is that spammers are only detected *after* they have added content to BibSonomy. Therefore, measures to reduce the amount of spam posts that are active for users that are flagged as spammers (e.g., the requirement to enter a captcha before each new post) are not effective before their first post. In this work we present an approach that aims to classify users immediately after they have completed the registration process, such that accounts of potential spammers can be flagged before they are able to post web spam. Such an approach can reduce the amount of malicious posts added to the system by enabling the early application of counter-measures, e.g., blocking malicious users. Reducing the amount of spam posts is beneficial for the performance of the system, since they produce a considerable amount of data that needs to be handled by the storage and indexing infrastructure. To the best of our knowledge, this is the first published research which analyzes approaches for detecting spam users at registration time in social bookmarking systems.

This paper is organized as follows: in Section 2 we present

[1]http://www.bibsonomy.org/

ACM ISBN 978-1-4503-4364-0/16/05.
DOI: http://dx.doi.org/10.1145/2915368.2915370

an overview on the state of the art, in Section 3 we present our approach, and in Section 4 we present the results of our analysis. We conclude the paper in Section 5.

2. STATE OF THE ART

Many works on spam focus on spam detection in emails. Due to the sensitive nature of the topic and the difficulty of performing appropriate experiments, fewer works focus on the detection of malicious users.

Krause et al. [6] developed the foundations for the automatic spam detection framework for BibSonomy. They used profile information of users, like their user name, email address, homepage, real name, as well as the tags and resources they had posted. From this data they extracted 25 features grouped into five categories. Among the *profile* features the length and the occurrences of digits in the users' name and email were modeled. In addition, they checked if the real name has two or three parts. They also used the number of other users with the same registration IP address, email domain, or top-level domain – the so-called *location* features. For the IP address they only counted the number of spammers. They also considered features that reflect the activity of the users, e.g., the number of tags per post, or semantic features like the users' co-occurrences with spam users using the same tag. They reached the best result with an SVM classifier with an AUC of 0.936.

Zafarani et al. [9] describe an approach to detect malicious users with a minimal amount of information, namely only the user name. Their assumption is that users use complex and diverse information to choose their user name. They used information surprise as a measure for complexity and the number and proportion of digits within the name as a measure for diversity. Another group of features modeled demographic information like age, gender, language, and knowledge. They also used the entropy of the input, since they argued that malicious activities often require anonymity. To achieve maximal anonymity, the input has to have maximal entropy. As language features they used normalized character-level bi-grams of user names. In addition, they also used the number and proportion of starting digits, the unique number of alphabet letters, and the maximal number of repetitions of a letter. Another aspect they focused on was the efficiency of the user. Therefore, they created featured based on the typing behavior: (i) the percentage of keys using the same hand as the previous key, (ii) the percentage of keys using the same finger as the previous key, (iii) the percentage of keys typed on each finger, (iv) the percentage of keys typed on each row, and (v) the approximate distance traveled for typing the word. They tested their approach amongst other classifier with an L1-regularized Logistic Regression and got an AUC of 0.9971.

Large online services like Google Mail do not rely only on a correct password. They use machine learning to check whether a user corresponds to the account he/she is logging in or not [3]. Having the correct password is one of many signals to decide whether to grant access or not.

Overall, several methods for detecting spam in web-based systems have been proposed, e.g., by Zhang et al. [10] who developed a method to detect spam and promotion campaigns on Twitter. However, most methods require data which is only available after the user has created the first post. We aim at detecting potential malicious users before they have added any content. In particular, since only few approaches and practically no data exist for comparison, our aim is to implement and extend features proposed by Krause et al. [6] and Zafarani et al. [9] and to test whether they they help to identify malicious users already at registration time.

3. APPROACH

We leverage machine learning to classify newly registered users based on features that are derived from the user data that is available at registration time. The classification approach relies on a training dataset of annotated users which we describe in Section 3.1.

In contrast to spam *posts* we want to classify spam *users*. Spam posts are posts which violate the terms and conditions of a service. Spam users are users which potentially add such spam posts or any other kind of spam to the service. Determining whether someone is a spam user is more difficult in our scenario, since we do not have any information about such violations or spam posts, yet. For training the classifier, defining who the spam users are is easy: Any user which was declared a spammer by the administrators of BibSonomy (typically after he/she has posted some spam) is regarded as a spam user.

Overall, 177 features were generated. A part of these features is derived from existing works (cf. Section 2). We here introduce and discuss the novel features which we extracted. We have grouped the features into the categories *language patterns*, *environment*, *keyboard*, and *population-based* which we present in Sections 3.2 to 3.5. We distinguish between *features* and *meta features*. Meta features refer to a set of features which are created in the same way from different information or by using different parameters. For instance, the meta feature *length* represents the features *nameLength*, *emailLength*, *homepageLength*, and *realnameLength*, which have the length of the user's name, email address, homepage, or real name, respectively, as values. In the tables showing the meta features we provide for each meta feature the number of features that are derived from it in the # column. For each new feature we specify its name, how it is created, and which range of values it has ($\mathbb{N}$ = natural number, $\mathbb{R}$ = real number, $\mathbb{C}$ = categorical data, $\mathbb{B} = \{0, 1\}$).

3.1 Dataset

Our dataset was created from a subset of the BibSonomy database in November 2015. We only considered users that were marked as spammer or normal user by the administration team of BibSonomy. All users that were automatically classified by the existing spam detection framework were ignored. An overview of the data is provided in Table 1. The features we extracted are based on the information the users provide during the registration process. These are the *name*, *email address*, an optional *realname* and an optional *homepage*. In addition, we used the *IP address* from which the user registered, as well as the registration *date* and *time*. We did not consider the *password* of the user. From a technical perspective the password would not provide useful information, since a salt and a hash function were applied to it. From an ethical perspective we think that the password of a user is something we should note use, because it is a very sensitive piece of information. To deal with optional data (e.g., the homepage) we marked the values of features where no data was provided by the user with a missing value flag.

Like Krause et al. [6] mention, determining whether a user is malicious or not is a subjective process. Different peo-

Table 1: Dataset Statistics.

	spammer	non spammer
total	180,376	3,614
with homepage	65,295	736
with realname	131,687	1,775
earliest registration	15 Nov 2006	3 Dec 2006
latest registration.	10 Nov 2015	8 Nov 2015

Table 2: Language Pattern (Meta) Features.

name	description	type	#
length	length	ℕ	4
propDigit	proportion of digits	ℝ	4
digit	contains digits	𝔹	4
numDigit	number of digits	ℕ	4
numStartDigits	number of starting digits	ℕ	4
propStartDigit	proportion of starting digits	ℝ	4
uniqueLetters	number of unique alphabet letters	ℕ	4
maxLetterRep	maximal times a letter is repeated	ℕ	4
nameMail	name and local part of email address are equal	𝔹	1
nameMailReal	name, local part of email address, and realname are equal	𝔹	1
realname2	two parts in realname	𝔹	1
realname3	three parts in realname	𝔹	1

ple have different judgments about what is considered to be spam or not. Therefore, we can not avoid a certain amount of noise resulting from the fact that the decision to mark a user as spammer or not was taken by different people.

3.2 Language Patterns

In an initial analysis of our dataset we found that spam users, besides using a higher number of digits in their input than regular users [6], also tend to have a higher proportion of digits. Therefore, we have 37 features that are extracted from the user input that is generated during registration. We analyze each user input and count the number of digits it contains and their proportion with respect to the length (number of characters) of the input. All meta features in Table 2 (except for the last four concrete features) were applied on the *name*, *email address*, *realname*, and *homepage*.

3.3 Environment

A large part of BibSonomy's users are academics. Thus, we assume that many of them are using an email address from their university. (We also assume that university email addresses are only given to legitimate users.) Thus, the feature *uniMail* indicates whether an email address is from a university. We used a publicly available list[2] which claims to contain the domains from most universities of the world. We can easily extract the domain from the email address and check it against this list. This approach is not perfect. First of all, we can not be completely sure that the list contains the domains of all universities. Second, not all institutes follow the pattern `<user>@<institute initial>.<university domain>`. And third, for various reasons (e.g., security problems with the mail server, phishing, etc.) malicious users could have used university addresses for their registration. Overall, such issues should be rare.

[2]https://github.com/Hipo/university-domains-list

Table 3: Environment Features.

name	description	type
country	country of user	ℂ
regHour	registration hour in user's local time	ℂ
regDay	registration day in user's' local time	ℂ
uniMail	is the email address from a university	𝔹

Analyses have shown that users from some countries tend to produce more spam than users from other countries [1]. Therefore, we used the *country* of the users as a feature. Since the country is not directly provided by the user, we approximate it using the user's IP address. We used the free ip-to-geo database GeoLite2 from Maxmind.[3] The values of this feature are the ISO 3166 Alpha-2 country codes.

Apart from the country we also used the local hour (*regHour*) and local week day (*regDay*) of the registration time. To transform the stored server time into the user's local time we approximated the user's timezone using the IP address. An analysis of this data showed that most regular users register between 7 a.m. and 10 p.m. Although spam users have also most registrations in that time period, a significant number of registrations is constantly performed at night. We assume that this could be the result of an automated registration process which takes place all day long. The weekday did not show any significant difference between normal users and spammers. The *regHour* is categorical data coded as an integer from 0 to 23. The *regDay* is categorical data coded as an integer from 0 to 7. An overview on the environment features is provided in Table 3.

3.4 Population-Based

We assume that a part of the spam users uses automated techniques to register at the services where they want to spread their spam. Then these techniques probably fall back to a pool of names from which they generate their inputs. A first simple approach is to normalize the input and count how many other users share this input. Normalized in this case means to convert the input to lower case and remove all non-alphabetic characters. With this approach we created features to count the local part of the email address, real name, first name (first part of the real name) and last name (last part of the real name). We also counted the number of domains and top-level domains of the homepage shared with other users, as well as the number of spam users which registered using the same IP address. In addition, we also used the ratios between normal and malicious users for the above-mentioned features. An overview on the population-based meta features is provided in Table 4.

3.5 Keyboard Features

We applied the features introduced by Zafarani et al. [9] based on keyboard layouts (Table 5) on the *name*, *email address*, *real name*, and *homepage*. As keyboard layouts we used QWERTY and Dvorak.

4. EVALUATION

[3]https://dev.maxmind.com/geoip/geoip2/geolite2/

Table 4: Population-Based Meta Features.

name	description	type	#
count	number of users with the same property	ℕ	6
ratio	ratio between normal and spam users with the same property	ℝ	6
spamIP	number of spam users with this IP	ℕ	1

Table 5: Keyboard Meta Features.

name	description	type	#
propRow	proportion of keys in a row	ℝ	32
propFinger	proportion of keys with a finger	ℝ	64
propSameFingerAsPrev	proportion of keys using the same finger as previous	ℝ	8
propSameHandAsPrev	proportion of keys using the same hand as previous	ℝ	8
distanceOnKeyboard	the approximate distance traveled for typing (assuming keys have width and height equals to one)	ℝ	8

Table 6: Top 20 Features.

feature name	group	info. gain
nameRatio	Pop.	0.13531
ipRatio	Pop.	0.12382
spamIP	Pop.	0.12382
realnameRatio	Pop.	0.04394
country	Env.	0.03917
firstnameRatio	Pop.	0.02406
lastnameRatio	Pop.	0.02058
uniMail	Env.	0.01849
emailDistanceOnKeyboardDvorak	Key.	0.01428
domainRatio	Pop.	0.01322
emailDistanceOnKeyboardQwerty	Key.	0.01283
emailProp4RowQwerty	Key.	0.00882
emailProp2RowDvorak	Key.	0.00879
emailPropLittleFingerRightDvorak	Key.	0.00621
emailPropMiddleFingerLeftDvorak	Key.	0.00591
realnameLength	Lan.	0.00469
emailProp3RowDvorak	Key.	0.00462
realnameEntropy	Lan.	0.00458
emailProp1RowQwerty	Key.	0.00404

4.1 Classifier

To classify our data we used Weka [8] with the algorithms Naive Bayes, Logistic Regression, the open source implementation of C4.5 – J48, and the Support Vector Machine (SVM) of the LibLinear package [4]. From LibLinear we used L1-regularized L2-loss SVM (called L1 L2 SVM) and L2-regularized L1-loss SVM (dual) (called L2 L1 SVM). All settings were set to Weka's default values.

4.2 Evaluation Setup

We first used a 10-fold cross-validation [7] to evaluate our features from Section 3. This resulted in very good results with an AUC score close to 1. But testing with independently created training and test datasets differed strongly from this. We realized that features that are based on the data of the whole population, like the features described in Section 3.4, can distort the result of the classic n-fold cross-validation approach. Having a closer look at the feature *spamIP* makes this clear: If the test and training data are created both from the same data, for each spam user's IP address the number of spam users with that specific IP address would at least be one. Therefore, every user with a spamIP value equal to zero can not be a spammer. In a real setting, however, this is not the case. A user with a value equal to zero could be a spammer with an IP address not in the dataset or only regulars users are in the dataset with this IP address. In the end, this feature would be overrated by the n-fold cross-validation, but in reality, the classification will not perform as good as predicted. To avoid this problem, we have to ensure that the population-based features are computed only on the training data.

To implement this, we modified the n-fold cross-validation such that we can ensure that the feature data is only created from the population of the training data. Therefore, we map the created feature data to the raw data. With this mapping we can in each fold recreate the population of the training data, rebuild the population-based features, and then update the corresponding data in the training and test set. Figure 1 shows this process. Cylinders represent sets of data, rectangles instances that manipulate data. First of all, the raw data, in our case the user information, is transformed with all features to a data set and a map from the created data to the corresponding raw data. This data set shall be evaluated through n-fold cross-validation. Therefore, the data set is split into a training and a test set. For each population-based feature we have to update the training and test set. First, we have to build the internal data of the population-based features with the raw data of the training set. To get this raw data, we use the map, produced at the creation of the dataset. Now we update each instance in the training and test set with this feature. After all population-based features are updated, we can build the classifier with the training set and test it with the test set. If there are any folds left, we repeat the process beginning with splitting the dataset.

4.3 Results

First, we analyze which features perform best and therefore computed their information gain (cf. Table 6). As can be seen, the most important feature is *spamRatio*, followed by several other population-based features. As explained in Section 4.2, we have to be careful interpreting these values. They likely perform worse than the information gain might suggest. We discuss the results of this group of features later in this section. The next best, non-population-based features, are *country* and *uniMail*. They clearly reflect the demographics of BibSonomy's user and spammer population. They are followed by features which only rely on the email address, real name, and name of the users, mostly using the techniques proposed by Zafarani et al. [9].

Table 7 shows the performance of the different feature groups, where for each group the results of the best performing classifier are shown. As could be expected from the analysis of the individual features in Table 6, environment and keyboard features perform best, where the environment

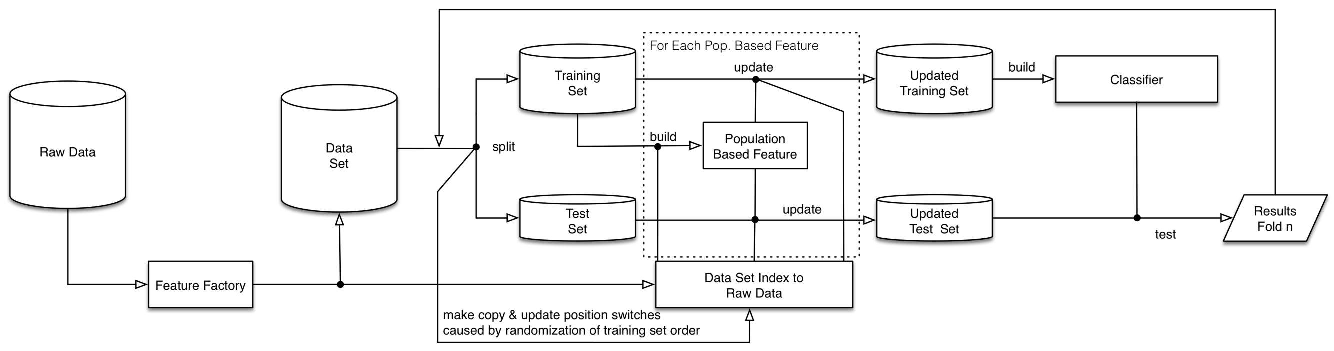


Figure 1: Modified n-fold Cross-Validation.

Table 7: Evaluation Feature Groups.

feature group	classifier	F1	AUC
Lang.-Env.-Keyb.	Logistic Regression	0.981	0.912
Environment	Naive Bayes	0.978	0.899
Keyboard	Logistic Regression	0.972	0.791
Language	Logistic Regression	0.970	0.731
Population-Based	L2 L1 SVM	0.933	0.527

Table 8: Evaluation values of all features.

classifier	F1	AUC	FP	FN
Logistic Regression	0.963	0.868	2478	5551
Naive Bayes	0.843	0.735	1345	43836
J48	0.975	0.601	3011	881
L1 L2 SVM	0.973	0.598	2894	1649
L2 L1 SVM	0.928	0.557	2901	16076

features clearly have the best F1 and AUC.

The overall performance of the classification approaches can be seen in Table 8. The best F1 and AUC is achieved by Logistic Regression on the categories language, environment and keyboard with an F1 of 0.981 and an AUC of 0.912. This is achieved by an excellent number of false negatives (FN), although the number of false positives (FP) is considerably higher than for Naive Bayes. In reality, the threshold for classifying users as malicious would have to be higher, such that the number of false positives is lower. We can see in the ROC curves in Figure 3e that this is possible: Logistic Regression can classify 50% of the malicious users correctly, before more than 1% of regular users are misclassified.

A comparison of the different feature groups and algorithms can be seen in Figure 3. We can see that the categories language, environment, and keyboard differ significant from a random classifier. However, this is not the case for the population-based features. The classification with features of this group does not significantly differ from a random classification. Therefore, we also evaluated all except the population-based features and got better results than with all features before. We assume that the bad performance is caused by some error but could not find the source so far.

Straight lines in the ROC curves are averages over groups of instances that occur when all group members have the

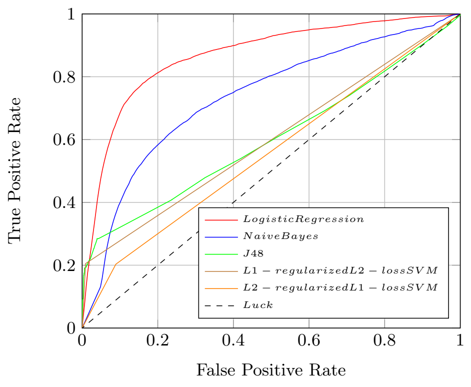


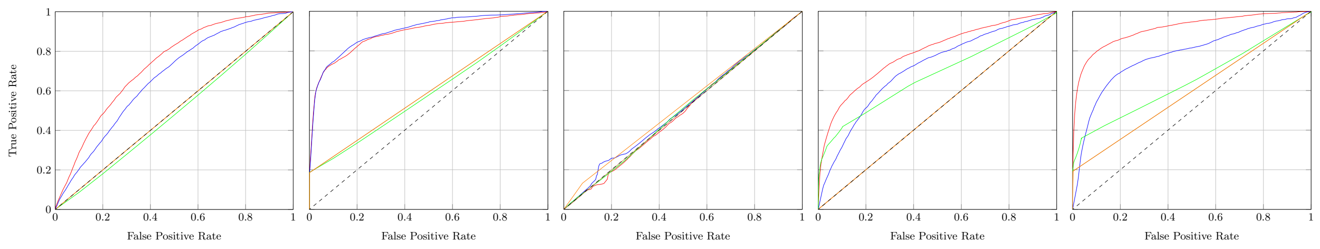
Figure 2: The ROC Curves for all Features.

same probability of being a spammer. For these instances the order is unknown and thus a random order is chosen.

5. DISCUSSION

Overall, our approach is almost as good as the approach by Krause et al. [6], but we can detect spam users at registration time and do not rely on spam posts. Zafarani et al. [9] achieved much better results with the best AUC of 0.99 vs. 0.91 in our case. This shows that there is potential to improve our approach.

We could show that it is possible to block a reasonable amount of spam users at registration time. This raises ethical questions. At the time of registration the users have not harmed the service. We would punish users for malicious actions that they might do in the future. We have to give them a chance to proof that they are regular users. Assuming that spammers create more than one account to spread spam, we could force them to authenticate using a short message service. If we only allow to use a phone number once, this could incur too much effort. Another idea is to increase the costs (time) for suspected spammers, by forcing them through an interactive tutorial explaining the system's basic functionality. Such a tutorial could also be used to collect data to improve the classification, which in turn would make it more difficult for spam users to create an account. In addition, such a tutorial could also help normal users to better understand the system, with the difference that they could skip it. Currently we only use information entered by the users into input forms for classification. We could ex-



(a) Language (b) Environment (c) Population-Based (d) Keyboard (e) Lang.-Envir.-Keyb.

Figure 3: The ROC Curves for the Different Feature Groups.

pand our feature set by including *how* the information was entered by the users. This is possible by tracking all mouse and keyboard actions of the users on the registration form.

By agreeing to the terms and conditions of BibSonomy, the users have agreed that their data can be used for research purposes. Nevertheless, the options for sharing this sensible data are very restricted. Therefore, we are not able to release the dataset in the public domain. Researchers interested in the data should therefore contact the second author such that we can discuss individual solutions.

6. REFERENCES

[1] IBM X-Force Threat Intelligence Quarterly, 2Q 2015. Technical report, IBM Corporation, June 2015.

[2] D. Benz, A. Hotho, R. Jäschke, B. Krause, F. Mitzlaff, C. Schmitz, and G. Stumme. The social bookmark and publication management system BibSonomy. *The VLDB Journal*, 19(6):849–875, 2010.

[3] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. Passwords and the evolution of imperfect authentication. *Comm. ACM*, 58(7):78–87, 2015.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9, 2008.

[5] Zoltán Gyöngyi and Hector Garcia-Molina. Web spam taxonomy. In *AIRWeb*, pages 39–47, 2005.

[6] B. Krause, C. Schmitz, A. Hotho, and G. Stumme. The anti-social tagger – detecting spam in social bookmarking systems. In *Proc. of the Fourth International Workshop on Adversarial Information Retrieval on the Web*, 2008.

[7] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. In *Encyclopedia of Database Systems*, pages 532–538. Springer, 2009.

[8] I. H. Witten and E. Frank. *Data Mining – Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann Publishers, 2000.

[9] R. Zafarani and H. Liu. 10 bits of surprise: Detecting malicious users with minimum information. In *Proceedings CIKM*, pages 423–431. ACM, 2015.

[10] X. Zhang, Z. Li, S. Zhu, and W. Liang. Detecting spam and promoting campaigns in twitter. *TWEB*, 10(1):4, 2016.