Digital Ecosystems and Business Intelligence Institute

Addressing the New Generation of Spam (Spam 2.0) Through Web Usage Models

Pedram Hayati

This thesis is presented for the Degree of

Doctor of Philosophy

of

Curtin University

July 2011

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signed: Leeann Hayden

Date: 1 JULY 2011

Abstract

New Internet collaborative media introduce new ways of communicating that are not immune to abuse. A fake eye-catching profile in social networking websites, a promotional review, a response to a thread in online forums with unsolicited content or a manipulated Wiki page, are examples of new the generation of spam on the web, referred to as Web 2.0 Spam or Spam 2.0. Spam 2.0 is defined as the propagation of unsolicited, anonymous, mass content to infiltrate legitimate Web 2.0 applications.

The current literature does not address Spam 2.0 in depth and the outcome of efforts to date are inadequate. The aim of this research is to formalise a definition for Spam 2.0 and provide Spam 2.0 filtering solutions. Early-detection, extendibility, robustness and adaptability are key factors in the design of the proposed method.

This dissertation provides a comprehensive survey of the state-of-the-art web spam and Spam 2.0 filtering methods to highlight the unresolved issues and open problems, while at the same time effectively capturing the knowledge in the domain of spam filtering.

This dissertation proposes three solutions in the area of Spam 2.0 filtering including: (1) characterising and profiling Spam 2.0, (2) Early-Detection based Spam 2.0 Filtering (EDSF) approach, and (3) On-the-Fly Spam 2.0 Filtering (OFSF) approach. All the proposed solutions are tested against real-world datasets and their performance is compared with that of existing Spam 2.0 filtering methods.

This work has coined the term ‘Spam 2.0’, provided insight into the nature of Spam 2.0, and proposed filtering mechanisms to address this new and rapidly evolving problem.

Acknowledgements

It is a great pleasure to thank everyone who helped me write my dissertation successfully.

I am sincerely and heartily grateful to my supervisor, Dr. Vidyasagar Potdar. His enthusiasm, inspiration, and patient efforts to explain things clearly and simply, provided me with encouragement, sound advice and teaching, good company, and numerous ideas. I would have been lost without him.

I also extend my deep gratitude to my co-supervisor, Dr. Alex Talveski, for his valuable critique, insights and suggestions for improving my research.

I am obliged to many of my colleagues for their helpful suggestions and peer reviews of my research. In particular, I would like to express my gratitude to Prof. William F. Smyth, Prof. Elizabeth Chang , Prof. Tharam Dillon, Dr. Chen Wu, Dr. D. Sculley, Prof. Takayuki Ito, Dr. Andrew Over, Prof. Costas S. Iliopoulos, Prof. Ian Sommerville, Prof. Robert Meersman, Dr. Fedja Hazdic, Dr. Jaipal Singh, Dr. Song Han, Prof. Heinz Dreher, Dr. Peter Dell, Dr. Ashley Aitken, Dr. Michael Brodie, Prof. Jianwei Han and my best friend, Kevin Chai.

I am truly indebted and thankful to my parents, Heshmatollah Hayati and Dr. Mina Davoodi, for their faith in me and allowing me to be as ambitious as I wanted. It was under their watchful eye and with their encouragement that I remained motivated, committed, and able to tackle challenges head on.

Lastly, I am grateful to my sister, my colleagues at the Institute for Advanced Studies in Basic Sciences, and my close friends for all their emotional support, camaraderie, entertainment and care.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter

- Provides an introduction to different types of spam.
- Gives an overview of current anti-spam technologies.
- Presents the focus of this research which is the new generation of spam called *Spam 2.0*.
- Explains the significance and importance of Spam 2.0 management.
- Explains the motivation and objective of this research.
- Presents the structure of this dissertation.

1.1 Introduction

The rapid adoption of the Internet in the day-to-day lives of people has provided a platform for information generation and consumption. The Internet is used on a daily basis to search for information and acquire knowledge. For instance, *blogs* are well-known online diary applications, *wikis* are known as huge knowledge bases, online discussion boards (*forums*) are places for knowledge sharing, brain storming and discussions, and *online communities* make it possible for people to create a virtual identity for themselves on the Internet. As newer platforms are created for the Internet, content generation and consumption become faster and easier than ever before. However, the ease with which content can be generated and published has also made it easier to create inappropriate, unsolicited, and irrelevant content. *Spam*, as this content is known, does not add value to the web user.

1.2 What is spam?

There is no exact definition of spam. Generally speaking, spam can be termed as an "*unwanted email message*". However, not all the unwanted emails are spam (e.g. emails that come through mailing lists). Another term could be "*promotional commercial emails*", but again not all the promotional commercial emails are spam (e.g. newsletters for software updates that the user has requested to receive in his/her inbox). Spam can also be referred to as "*junk email*" but this begs the question, "*what is junk email?*"

One general and acceptable definition of spam is: *mass, unsolicited and unwanted messages* (Junod 1997).

1.3 Origin of the term 'spam'

The origin of the term 'spam' comes from a 1970's popular TV show called "*Monty Python's Flying Circus*". During one of the sketches, a customer orders a meal from a restaurant. In the restaurant's food menu all meals contained a canned, pre-processed meat called Spam®. The restaurant's waitress insists on serving unwanted Spam® to customers (Pfleeger & Bloom 2005). Therefore, 'spam' is related to something unwanted which causes annoyance.

1.4 Different forms of spam

The most widely recognised form of spam is the spam in email communications. However, spam exists in other media such as instant messagers (Spam in Instant Messenger or SPIM), Internet Telephony (Spam in Internet Telephony or SPIT), mobile phone (M-spam), Websites (Web spam), and finally Web 2.0 (Spam 2.0). Figure 1.1 presents adaptability of different spam types to different platforms. The figure illustrates different types of spam (Spam, SPIM, M-Spam, Web Spam, SPIT and Spam 2.0) for different media (Email, Instant Messenger, Mobile, Search Engines, Internet Telephony and Web 2.0). A detailed description of each type of spam is given in the remainder of this section.

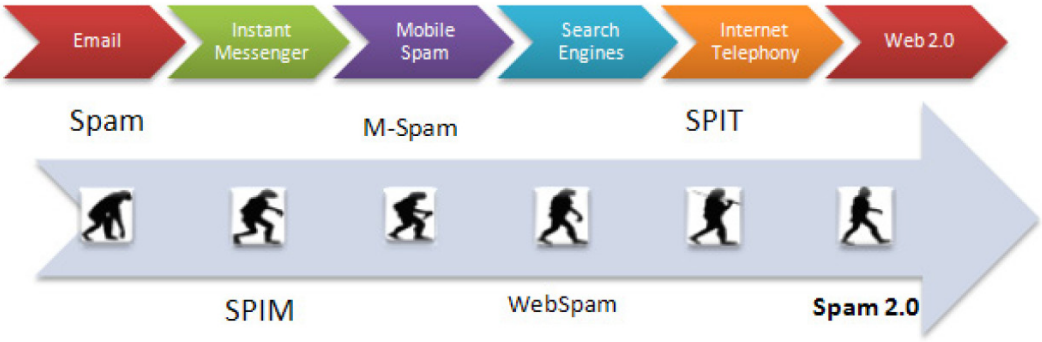


Figure 1.1: Evolution and adaptability of spam once new media emerge

Although spam has evolved over time, most research efforts have focused on earlier types of spam such as email spam. Figure 1.2 illustrates the number of hits for different spam types on the Internet. This figure has been derived from the number of results for each spam type, from major search engines on the Internet[1].

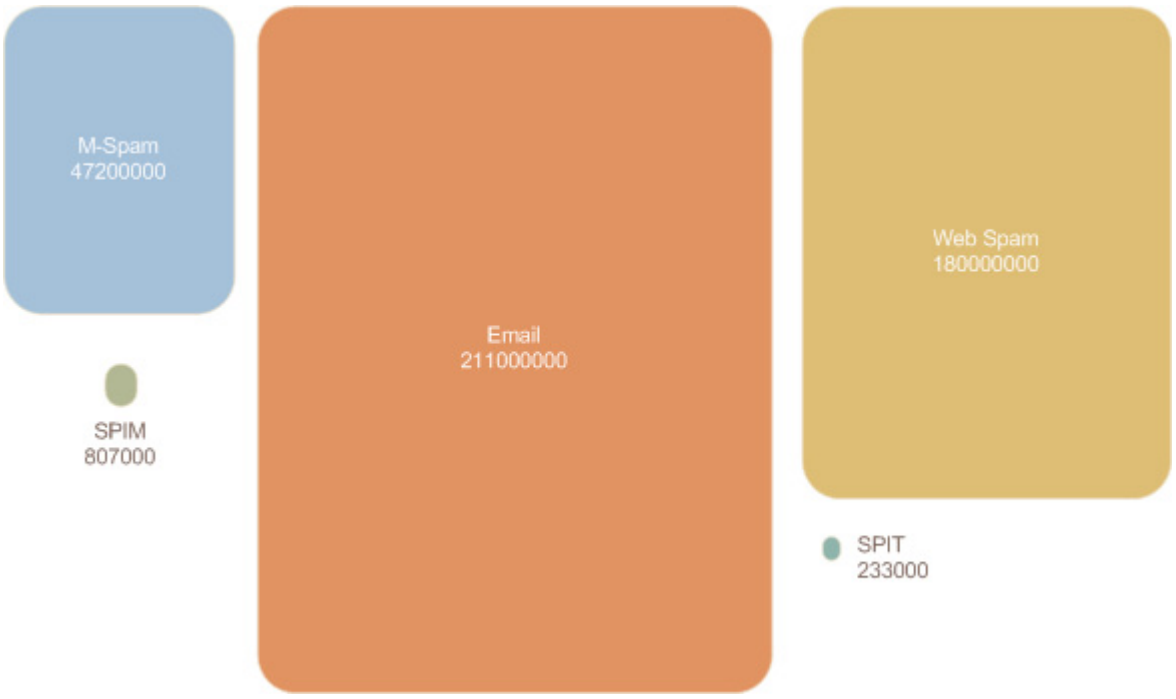


Figure 1.2: Number of hits for different types of spam.

[1] There is a possibility of repeated results in the figure; however, it would affect all types of spam and they would nullify each other.

1.4.1 Email Spam

Email spam is the most common type of spam that targets the email system. Email spam is referred to simply as ‘spam’.

1.4.2 Instant Messenger Spam (SPIM)

Spam in instant messaging programs such as *Windows Live Messenger*, *AIM*, *Skype*, *Yahoo! Messenger*, *XMPP* etc is referred to as SPIM (Zhijun Liu et al. 2005) (Jun Bi et al. 2008). Users of such systems are often solicited by unwanted messages and links.

1.4.3 Mobile Phone Spam (M-spam)

The abuse of the text messaging service of mobile phones for the purpose of spam is referred to as M-spam or SMS spam (Cormack et al. 2007). As the cost of text services is going to be relatively cheaper, the impact of m-spam will increase rapidly in the future.

1.4.4 Webpage/Website Spam (Web spam or Spamdexing)

Web spam is defined as web pages that are deliberately created to trick search engines into offering unsolicited, redundant and misleading search results as well as mislead users to unwanted and unsolicited web pages (Gyongi & Garcia-Molina 2005). This form of spam is also known as *Spamdexing* or *Black Hat SEO* since it targets Search Engines Optimisation (SEO) algorithms. SEO are techniques that are designed to improve the ranking of a particular website or a web page in a search engine indexing result in an ethical way (e.g. more links to a website can improve its importance and ranking).

Web spam is quite widespread and there have been many efforts (ref. Chapter 2) in the literature to prevent propagation of web spam as well as removing spam websites from search engines indexes.

1.4.5 Spam over Internet Telephony (SPIT)

Voice over Internet Protocol (VoIP) is a communication protocol for the transmission of voice and multimedia through Internet Protocol (IP)-based networks. Since, VoIP communications can reduce both infrastructure and communication costs, they are rapidly becoming popular. Unsolicited phone calls received via the VoIP system are known as SPIT (Quittek et al. 2008). This form of spam is

more annoying than other forms since it directly targets VoIP users' phone devices, causing distraction and discomfort.

So far, the most common spam types, i.e. SPIM, SPIT, M-spam, and Web spam, have been explained. In the next section, a detailed description of the next generation of spam i.e. Spam 2.0 is given. This is the main focus of this research study.

1.5 Spam in Web 2.0 Applications (Spam 2.0)

Web 2.0 is commonly associated with web applications that facilitate interactive information sharing, interoperability, user-centred design and collaboration on the World Wide Web (Oreilly 2007). The read/write Web 2.0 concepts are the backbone of the vast majority of web services. In contrast to non-interactive websites, Web 2.0 places greater emphasis on human collaboration through a participation architecture that encourages users to add value to web applications as they use them (Oreilly 2007). Today, Web 2.0 functions are commonly found in web-based communities, applications, social-networking sites, media-sharing sites, wikis, blogs, mashups, and folksonomies. They are widely provided by government, public, private and individual entities.

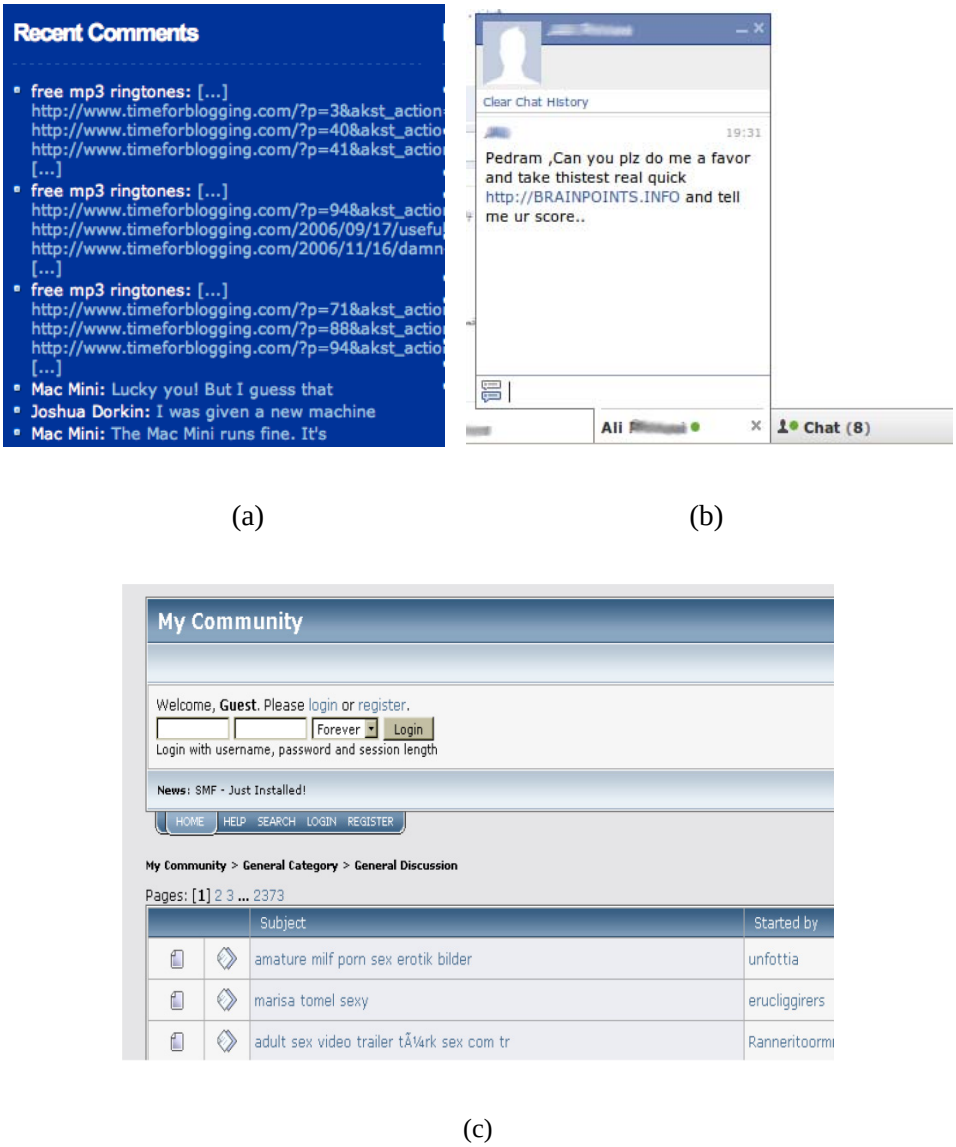
New media brings in a new means of communication that is not immune to unsolicited concerns. A fake eye-catching profile in social networking websites, a promotional review, a response to thread in online forums with unsolicited content and a manipulated wiki page etc, are examples of a new form of spam on the web that is referred to as Web 2.0 Spam or Spam 2.0 (Hayati, Potdar, Sarenche, et al. 2010). Spam 2.0 (or Web 2.0 Spam) is defined as below.

1.5.1 Definition of Spam 2.0

"Spam 2.0 is defined as the propagation of unsolicited, anonymous, mass content to infiltrate legitimate Web 2.0 applications" (Hayati, Potdar, Sarenche et al. 2010). The key term in this definition of Spam 2.0 is the word "legitimate" because this type of spam is distributed through legitimate websites. This differentiates Spam 2.0 from other types of spam (Hayati, Potdar, Sarenche, et al. 2010).

Previously, spammers used media such as email, instant messengers, internet telephony etc to spread spam; hence, such media serves as a communication medium between spammer and genuine users. However, Spam 2.0 acts differently as it uses legitimate Web 2.0 applications to host spam. In Spam

2.0, spammers no longer host their own email/web servers. Instead, they post spam on legitimate websites like a genuine users blog. Legitimate websites here refer to those genuine website used by users including government departments, universities, companies, homepages websites etc. Spam 2.0 infiltrates legitimate websites by posting/hosting spam content on them. Figure 1.3 illustrates an example of Spam 2.0 in legitimate website: blog comments, forum posts and online communities.



(a) (b)

(c)

Figure 1.3: Examples of Spam 2.0. (a) A comment spam, (b) an online community spam, and (c) a forum spam

Spam 2.0 has the following characteristics:

Spam 2.0 targets Web 2.0 platforms. Spam 2.0 spreads through any web applications that allow users the privilege of contributing to content generation including forums, comment systems, wikis, online communities etc.

Spam 2.0 post is parasitic in nature. Spam 2.0 is hosted on legitimate and often official websites. If such information persists, the credibility of such websites is diminished, and many users can be misdirected to scams and computer malware; websites may be blacklisted, thereby depriving all other users from accessing legitimate content.

Spam 2.0 aims to deceive web users. Spam 2.0 posts can be attractive enough to deceive and misdirect users to scam or malware (e.g. a comment in user's profile page). This affects the user's experience of surfing the web, causes frustration, and diminishes the quality and credibility of the web content.

Spam 2.0 can propagate web spam. Spam 2.0 can be generated using web spam tactics in order to manipulate the result of search engine ranking. URLs for spam websites can be distributed via numerous public Web 2.0 platforms. Hence, it would tamper with targeted-website ranking in search engine results (ref. Section 2.2.3, Web spam tactics).

Spam 2.0 distribution is automated. Typically, Spam 2.0 is distributed through automated tools. This allows Spam 2.0 to increase more rapidly and infiltrate a significant number of targets. Additionally, there is evidence of real human spamming activities through the hiring of low-income employees (Workathome 2009).

Spam 2.0 wastes network and web resources. Network bandwidth and web storage is misused for the distribution and hosting of Spam 2.0. Additionally, this may have ramifications for computational and power resources used by network facilities to handle Spam 2.0 content.

To date, as a result of the success and impact rates of Spam 2.0, it is far more popular and has a far greater negative socio-economic impact. The next section discusses the significance and nature of Spam 2.0 compared with previous types of spam.

1.5.2 Spam 2.0 vs. Email Spam

Spam 2.0 offers a far more attractive proposition compared with traditional spam, specifically email spam (Hayati & Potdar 2009). Web 2.0 applications can be discovered through a simple search engine query that contains "domain keywords" and a "web application name". Email addresses are procured

in a similar fashion except that email address details are commonly tightly controlled online and are far more difficult to source. In contrast, Web 2.0-based websites try to promote themselves so that web users are able to easily find them and contribute. Such applications rely upon relatively anonymous social network users to freely interact online.

A single Spam 2.0 attack may reach many targeted and domain-specific users, whereas a single email message would potentially reach only one random individual if the email address is real and not stopped by today's effective email spam filters (Hayati, Potdar, Sarenche et al. 2010).

Furthermore, when individuals discover an email message that has bypassed their filters, they are able to remove it. However, online messages typically cannot be deleted by regular users and persist online until a website operator deals with them, often impacting on many users in the meantime.

1.5.3 Spam 2.0 vs. Web spam

In contrast to web spam, in Spam 2.0, spam posts are spread in legitimate websites. Spam 2.0 posts are posted to publicly accessible blogs, discussion boards, wiki etc.

Furthermore, Spam 2.0 not only targets search engines, but can also be formulated to mislead human users. For example, a comment as illustrated in Figure 1.4 in a blog post might mislead human users to visit the associated link.

Thanks, I am interested in your post. Could you please let me know what you think about this article?

WWW.SPAMWEBSITE.COM

Figure 1.4: Example of misleading comment spam

To date, spam has seriously compromised the quality of the web. This problem has been exacerbated by the emergence of Spam 2.0 that infiltrates the content of legitimate websites. Hence, research in Spam 2.0 management is of prime importance if web content is to maintain its quality and credibility.

1.6 Anti-Spam Solutions (Spam Filtering)

1.6.1 Importance of spam filtering

The significant increase in the amount of spam on the web and web applications has made it imperative to find reliable anti-spam solutions. A report carried out by Sophos in 2008 revealed that every 3 seconds a new spam-related web page is created (net-security.org 2008). In another words, every day 23,300 spam-related web pages are hosted on the web. *Live Spam Zeitgeist* has shown an increase of over 50% in the number of spam messages (e.g. spam comments in blogs) since 2008 (Live-Spam-Zeitgeist 2009). This report showed a dramatic increase in the amount of Spam 2.0. Additionally (Kolari et al. 2006), pointed out that over 75% of blogs are splogs and this figure may have increased even more by now.

1.6.2 Anti-Spam concepts

To provide a better understating of the concepts involved in the area of anti-spam, an analogy between spam filtering approaches and the *clinical service chain* in a health care system is discussed here. Figure 1.5 illustrates 4 main groups of activities in the clinical service chain – prevention, diagnosis (or detection), treatment, and recovery. An explanation of these concepts in terms of anti-spam is as follows.

Figure 1.5: Anti-spam solutions main activities

1.6.2.1 Prevention

In the health care system, 'prevention' refers to those measures taken to prevent infection or disease (Horstkotte et al. 2004). Similarly, in the domain of spam management, prevention constitutes the measures taken to *prevent* spam rather than detecting and removing spam from the system (ref. Section 1.6.4.3). Prevention activities restrict spam content by increasing distribution costs through time, computation, and economic challenges.

1.6.2.2 Diagnosis (detection)

Regarding health care, 'diagnosis' is the process of identifying diseases (Horstkotte et al. 2004). Similarly, in terms of spam management, diagnosis or detection includes any measures taken to identify spam patterns. This task can be done through an analysis of spam content or behaviour patterns (ref. Section 1.6.4.1). The spam patterns within content can be identified by extracting spam keywords from email messages, analysing request header information etc.

1.6.2.3 Treatment

In the health care system, 'treatment' consists of those measures taken to cure a disease or health problem (Horstkotte et al. 2004). Similarly, with spam, treatment focuses on removing spam from the

system and enables system operators or system users to effectively remove the infiltrating spam. For example, a feature in email systems that facilitates the removal of spam in bulk can be considered as a treatment activity. This area of research is quite young and not many studies have been undertaken so far.

1.6.2.4 Recovery

The restoration of health to a normal condition after a health problem has been treated, is the recovery process in the health care system (Horstkotte et al. 2004). Similarly, in the spam management domain, recovery is any kind of method that returns the system to its former status and retains the details of recent spam distribution for future spam filtering. For example, a system can blacklist the IP address of origin of recent spam. Similar to treatment, this area of research has not received much attention to date.

Based on the concepts explained above, the next section discusses the existing approaches to spam filtering.

There is a broad spectrum of potential solutions for spam filtering, ranging from legal to technical filters. The aim of each and every solution is to make spam unprofitable. The following sections discuss each solution.

1.6.3 Legal Approach

Legal solutions strive to regulate spam through appropriate legislation and enforcements. For example, according to the *Australian Spam Act 2003* it is illegal to "send or cause to be sent" spam through email, instant messengers, and mobile phones. Although legislation makes it easier for spam to be filtered out, there are no word-wide legal agreements regarding spam filtering. Therefore, anti-spam regulations might not be not effective enough (e.g. spam that originated from a location that is not part of spam-relaying countries might not be easily prosecuted) (B. Whitworth & E. Whitworth 2004).

1.6.4 Technical Approach

Technical solutions are practical real-world solutions that can be used in softwares. Technical solutions can be categorised according to four approaches – detection-based, link-based, early-detection based, and prevention approaches.

With regard to *clinical service chain*, detection-based, link-based, and earl-detection based approaches are similar to *diagnosis* step while prevention approaches are similar to *prevention* step. The details of each approach are discussed in the following sections.

1.6.4.1 Detection-based approach

Methods in this category deal with the content of spam in order to discover spam patterns (Paul et al. 2007). These detection methods search for spam words, templates, attachments etc inside the content. For example, an email with a subject "$$$$ BIG MONEY $$$$" would be marked as spam by a keyword-based spam filter (Ion et al. 2000). As shown in Figure 1.6, detection methods follow 3 steps. The first step is the input content from different systems (e.g. email, sms). The second step is to mine features such as keywords from content and meta-content. Finally, the last step is to perform classification which results in spam or non-spam content.

Classification criteria can be rule-based or statistics-based. The former employ heuristic rules such as whether or not there is organisation of the content, existence of specific keywords etc. Rules lists need to be updated regularly with new rule sets, otherwise the detection system cannot perform effectively.

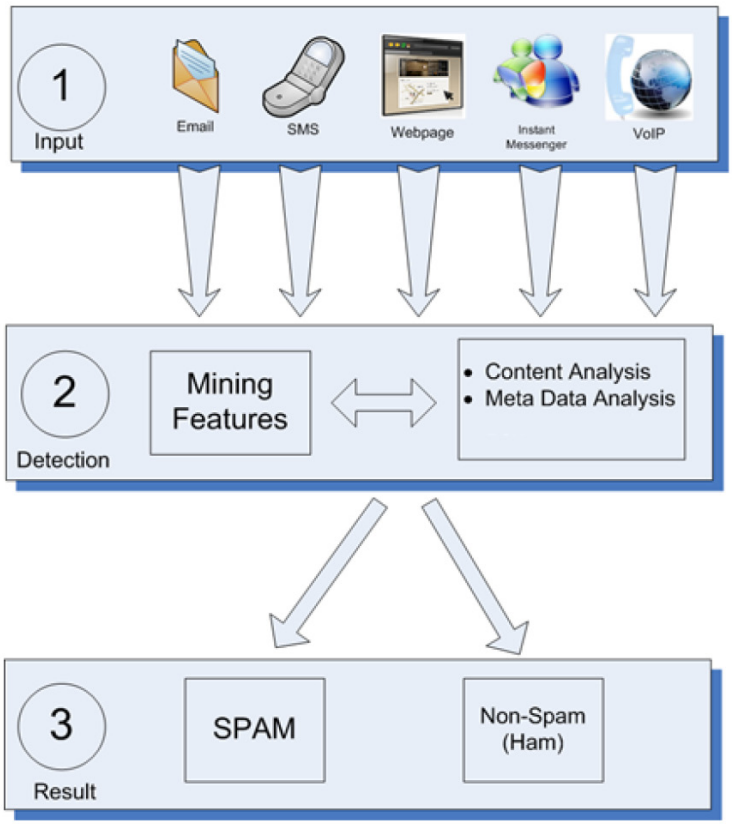


Figure 1.6: The procedure for the detection-based spam filtering approach

Detection-based spam filtering methods can investigate different parts of the content in order to find spam patterns. These parts are platform-dependent. Table 1.1 lists examples of such parts in the email, web page and blog comment.

Table 1.1: Examples of spam features in different platforms.

Platform	Where to look (part of content)
Email	Message content Headers (origin and destination address) Message path route
Web page	Title of page Structure of the page Content of the page Amount of content vs. hyperlinks
Blog Comment	No. of links per comment content. Similarity of comment content and blog post.
Forum Post	Content of post No. of images per post No. of links per content

1.6.4.2 Link-based approach

In the link-based approach, spam filtering methods investigate various link-analysis, graph-analysis or network analysis techniques to separate spam nodes from genuine ones (Caverlee et al. 2009). By constructing a graph-view of the web/email communication and looking for differentiate parameters (e.g. number of link between two web page), such methods are capable of discovering spam websites (Gyongyi et al. 2004). Figure 1.7 illustrates a sample procedure in link-based approaches. The vertices of the graph can be web pages, websites, hostnames, email contacts etc and edges can be defined accordingly.

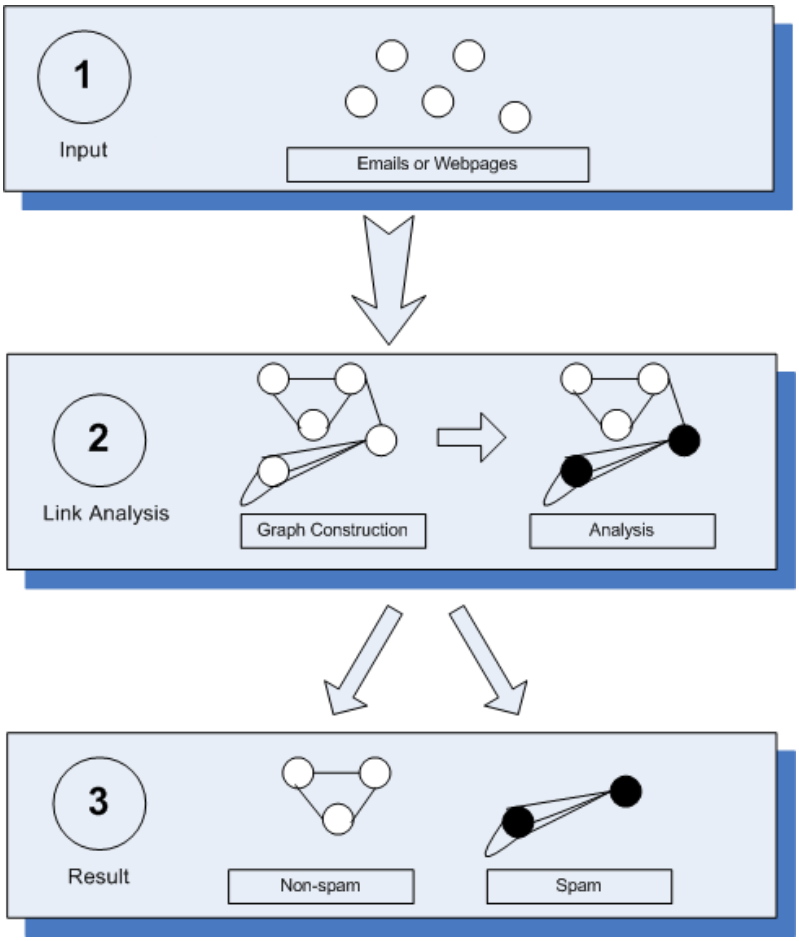


Figure 1.7: The procedure of the link-based spam filtering approach

Link-based approaches have been widely studied in web spam filtering (ref. Section 2.2.5) while there are some efforts in email spam filtering as well. Table 1.2 presents several sample parameters used in the link-based approaches.

Table 1.2: Sample parameters in link-based spam filtering approaches

Platform	Where to look (parameter)
Email	No. of sent emails vs. the number of received emails
	No. of emails a user has sent within a specific period of time
	No. of emails a user has sent at a particular time of day
Web	No. of links that a web page is linked to vs. # of links a web page is linked from
	No. of suspicious links that a web page is linked from
	No. of links that a web page is linked from a specific host

1.6.4.3 Prevention-based approach

Prevention methods strive to limit automated access to the system (Paul et al. 2007) (Hayati & Potdar 2008). By utilising computational, time, and economical challenges, prevention methods make spam distribution difficult (Paul et al. 2007). As illustrated in Figure 1.8, prevention methods allow the genuine user to enter the system while keeping the spammer out of the system.

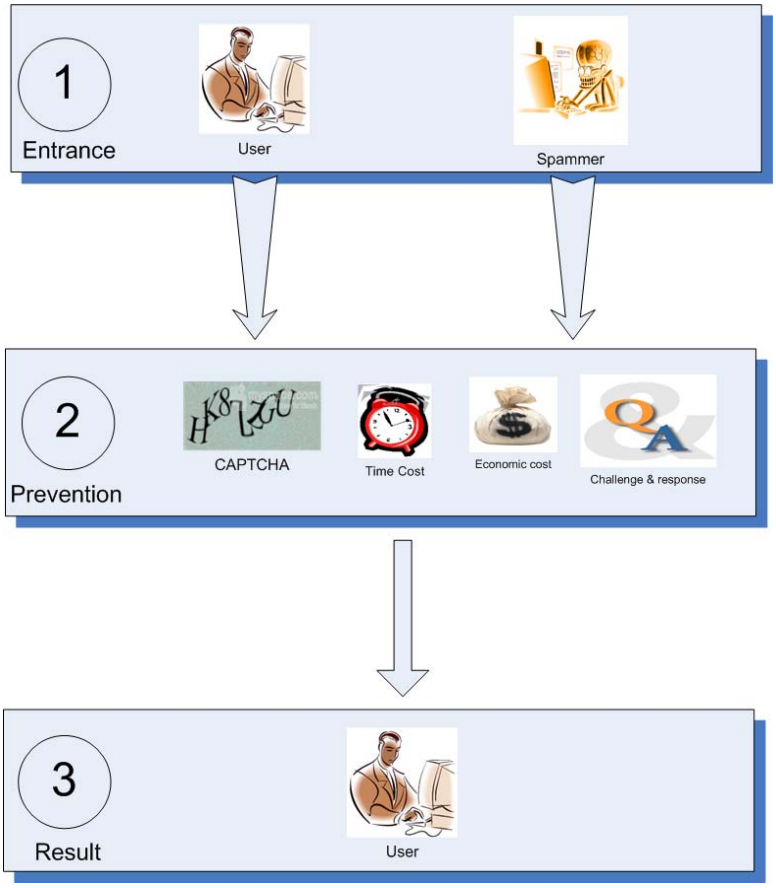


Figure 1.8: The procedure for the prevention-based spam filtering approach.

Examples of widely used prevention-based spam filtering approaches are as follows.

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA): a challenge and response test designed to distinguish human users and automate generated requests (Luis von et al. 2004). CAPTCHA tests needs to be automatically generated that are easy for human users to solve but hard for machine to bypass. Figure 1.9 illustrates example of recent CAPTCHA known as reCAPTCHA (von Ahn et al. 2008). Such tests have been widely used in email and web. Most of the current web applications include CAPTCHA as part of their content submission and registration process.

Figure 1.9: A recent CAPTCHA challenge known as reCAPTCHA.

Proof-of-Works (POW): The idea of this test is based on dissymmetry i.e. on the client side, the task is feasibly hard to perform but can be easily verified on the server side (Paul et al. 2007). HashCash is an example of a POW test that has been designed to limit email spam. For example, the sender of the email is required to calculate a stamp prior to sending the email (Back 2002). Stamp generation requires CPU time calculation, which possibly increases the cost per email message (more details in Section 2.3.4.2).

Challenge-response (CR): A CR test requires the sender to successfully overcome a challenge before allowing the message to pass through to the destination (Paul et al. 2007). CR is normally used in email systems. For example, the sender of unknown incoming email needs to click on a specific URL in the auto-generated reply email from the spam filter.

Gray listing: This test requires the sender to re-submit the message by assigning delay or rejecting a message for the first time (Levine 2005). Typically, gray listing is used in email systems to verify the sender.

Time challenges limit the amount of access for the actions in the system based on time constraints. For instance *Flood control* is a type of time challenge that limits the number of requests a client can send to a forum within a specified time interval. Another example is when a user creates a new thread in a forum, s/he may have to wait for some time before creating another thread.

Economic challenges are designed to make spam distribution non-economical, thereby making the sending of spam unaffordable as the cost per spam message is more than the potential revenue to be generated. Techniques such as E-stamp require the sender of the email message to pay a specific amount of money to the receiver (Fahlman 2002). Once the receiver accepts the sender's email message, this payment would be refunded.

1.6.4.4 Early-detection based approach

Early-detection methods prevent the distribution of spam throughout the system. Early-detection methods typically examine the nature and behaviour of spam generation (Hayati, Potdar, Chai et al. 2010). By studying the behaviour of content distribution and investigating behavioural features, early-detection methods are able to distinguish spam from legitimate content. Such methods are not detection-based since they do not investigate content to find spam patterns; nor do they impose costs on user actions to limit the access to the system. Figure 1.10 shows the process of the early-detection based approach that investigates users' behaviour and interaction inside the system for the detection of spam users.

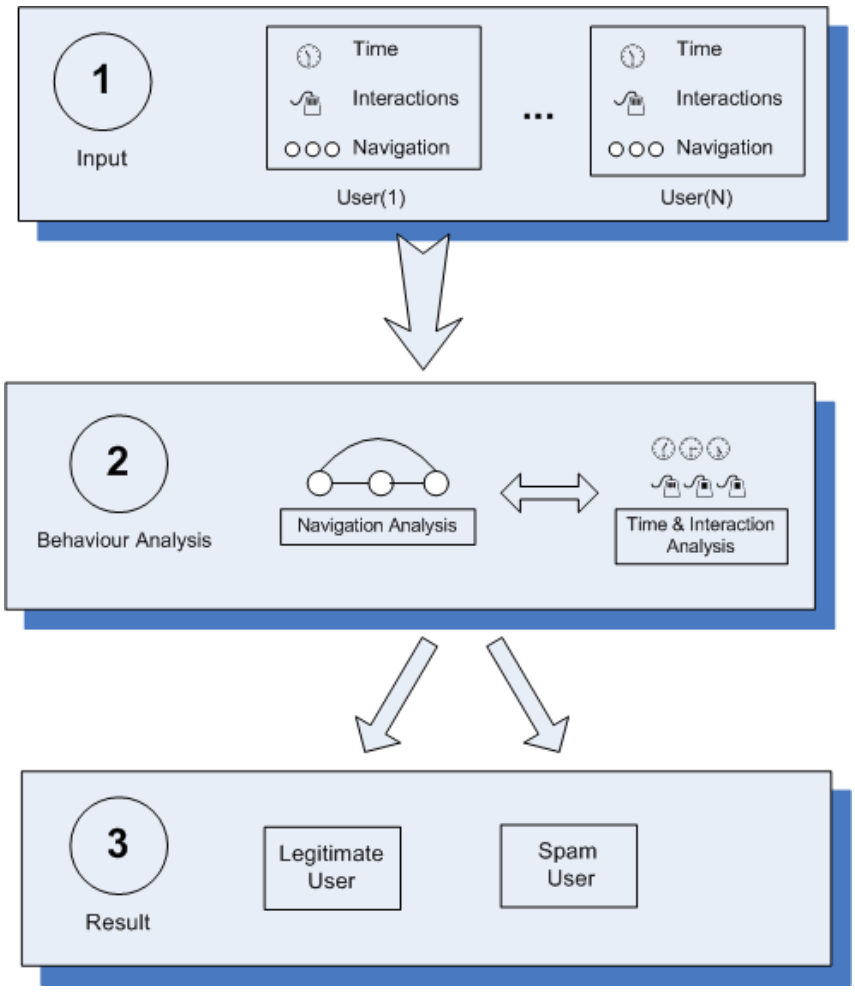


Figure 1.10: The procedure of early-detection based spam filtering approach.

Behavioural features vary depending on the different platforms. Table 1.3 lists some of the behavioural features for blog comment and general Web 2.0 application platforms.

Table 1.3: Examples of behavioural features in early-detection approach.

Platform	Behavioural Feature
Blog Comment	Existence of keyboard interaction while filling up comment form. Existence of mouse interaction while moving across comment form's fields. Amount of time user spends on submitting comment form.
General Web 2.0 Application	Navigation pattern. Amount of time user spends on each page. Amount of time user spends on filling up each file in the form.

The existing approaches to combat spam have been discussed. The next section explains the main research motivations for this dissertation.

1.7 Research Motivations

The main purpose of this research is to address the issue of Spam 2.0 filtering and develop a framework for early-detection and prevention of spam on Web 2.0 platforms. However, it should be duly noted that Spam 2.0 filtering has many applications including, but not limited to, preventing scam and online fraud, and improving content quality. Current literature highlights some key problems associated with Spam 2.0 filtering; these problems serve as the motivation for this study. These key problems include:

1. Spam 2.0 is not addressed as a new type of spam.
2. Spam 2.0 is parasitic by nature.
3. Spam 2.0 causes user inconvenience.
4. Existing detection approaches are not effective in filtering Spam 2.0.
5. The increasing rate of Spam 2.0 distribution is out of control.
6. Online fraud and scam are facilitated by Spam 2.0.
7. Trust, credibility and quality of information on Web 2.0 platforms are compromised because of Spam 2.0.

1.7.1 Spam 2.0 is not addressed as a new type of spam

The existing literature focuses mainly on spam filtering in different web applications but does not address Spam 2.0 as a new generation of spam. For example, some works have addressed platform-dependent spam filtering in social bookmaking, video sharing, and comment systems; however, there has been no effort to address Spam 2.0 as a whole big picture (Hayati & Potdar 2009).

In order to effectively combat Spam 2.0, it is necessary to approach it as a new type of spam which has its own characteristics. Therefore, a specific solution is needed to address such characteristics. Otherwise, an anti-Spam 2.0 solution would be limited to specific platforms only (i.e. platform dependent) and could not be extended to other web applications. Therefore, solutions might not be effective enough for combating spam.

1.7.2 Spam 2.0 is parasitic by nature

Spam 2.0 is different in nature from previous types of spam. For instance, Spam 2.0 has parasitic nature that infiltrates legitimate applications. Hence, traditional spam filters might not be adequate for combating this new type of spam. To date, most of the efforts on Spam 2.0 filtering have been concerned with adapting old methods applied to email spam filtering to detect Spam 2.0. Therefore, the outcome of such efforts is inadequate for Spam 2.0 filtering (Zinman & Donath 2007) (Jindal & Bing 2007) (Benevenuto et al. 2008).

1.7.3 Spam 2.0 causes user inconvenience

Some anti-spam approaches, mainly prevention-based, decrease the number of users. For instance, typical CAPTCHA requires users to read and write phrases shown inside images. Often this image is deformed and it is not easy to read. This is inconvenient for humans (Figure 1.11). However, as computers become more powerful, they might decipher such prevention techniques better than humans can. Additionally, prevention-based solutions are extremely challenging for people with a disability (May 2005).

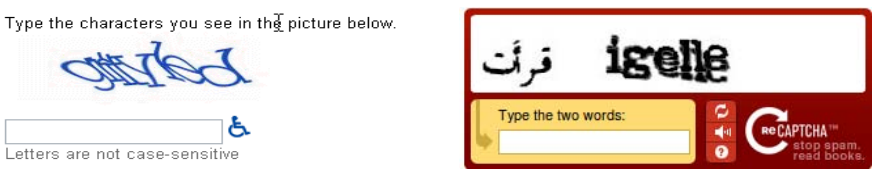


Figure 1.11: Two sample CAPTCHAs that are difficult even for humans to decipher

1.7.4 Existing detection approaches are not effective in filtering Spam 2.0

Since most of the current efforts adapt old techniques for Spam 2.0 filtering, the outcome has not been effective enough and most current techniques have low accuracy in detecting Spam 2.0 (Nitin & Bing 2008) (Zinman & Donath 2007) (Jindal & Bing 2007) (Benevenuto et al. 2008).

1.7.5 The increasing rate of Spam 2.0 distribution is out of control

As Web 2.0 increases, there are more opportunities for Spam 2.0 to flourish. The distribution of Spam 2.0 can become faster by using automated tools such as *Auto submitters* and *Web Spambots*.

An auto submitter is an automated technique used to perform automated tasks on the web (Hayati, Potdar, Sarenche et al. 2010). Such tools are used to distribute spam on many websites quickly and within a short period. They target online input forms such as contact forms or comment forms for multiple consecutive submissions.

Web spambots (or simply *spambots*) as a type of *web robots* are another automated but more sophisticated technique in Spam 2.0 (Hayati et al. 2009). Web robots, also called Internet robots, are automated agents or scripts that perform specific tasks over the web in an automatic fashion (Koster 1993). They execute structured repetitive tasks faster than humans can do. Since the web is increasing enormously in size and value, web robots play an essential role in the Internet community (Tan & Kumar 2002).

Web robot is a double-edged sword: human assistance or a threat to web applications (Koster 1995). The early version of spambots were email spambots that harvested email addresses from web pages, mailing lists, directory databases, and chat rooms in order to send numerous unwanted emails (Tan & Kumar 2002). Email spambot can send many emails efficiently and reliably. However, as Web 2.0 opens new avenues for interacting with people, new versions of spambot known as Web spambots has been developed to infiltrate Web 2.0 applications. Web spambots are intended to submit spam content to Web 2.0 applications. Web spambots surf the web to find Web 2.0 applications and spread Spam 2.0. They can target one specific Web 2.0 application, such as wikis, as well as focus on one particular website such as MySpace (Hayati et al. 2009).

Automated techniques have increased the stream of Spam 2.0 but have not been thoroughly investigated. Therefore, an efficient Spam 2.0 solution requires countermeasures to hinder such automated techniques.

1.7.6 Online fraud and scam are facilitated by Spam 2.0

Some might think that spam itself is not a significant enough problem to merit real attention. However, the rapid increase in the amount of online fraud and scam highlights the need for effective spam filtering. To date, the key factor that facilitates online fraud is its fast distribution over a small timeframe. This can be achieved by utilising spam techniques to target as many people as possible. Hence, Spam 2.0 filtering can significantly and indirectly reduce the amount of scam and online fraud.

1.7.7 Trust, credibility and quality of information on Web 2.0 platforms are compromised because of Spam 2.0

Spam 2.0 causes frustration for users who are searching for good quality information, for website operators who have to filter spam content and let the genuine content to come through, for network administrators having to block unwanted spam traffic in order to use valuable network resources, and for hosting providers who need to save their storages from spam content. As more spam content infiltrates Web 2.0, the quality and credibility of the content decrease; moreover, the users' search experience is degraded.

Based on the foregoing issues, it is clear that this area of research is quite young but needs immediate attention. There is an urgent need for an efficient and practical filtering mechanism for Spam 2.0. Policies for current filters need to be updated. Web application developers need to be aware of current Spam 2.0 techniques and a way to secure their applications etc.

This research tackles a unique spam filtering approach: it examines spam-content generation behaviour on the Web 2.0 platform in order to distinguish spam from legitimate content. The research also provides a definition for this new type of spam.

1.8 Significance of Research

The significance of the problems addressed in this thesis is threefold and the benefits resulting from this research include social, economic and technical (scientific) advantages.

1.8.1 Social

1.8.1.1 Improve quality of the content on the web

Spam 2.0 leads users to draw a misinformed conclusion or make an unwise decision, spend more time to find their desired content, visit an unsolicited website, and become angry or frustrated. The elimination of false information on the web is important to protect organizations from the adverse impacts of poor information quality. For instance, in a recent case, incorrect news about United Airlines filing for a second bankruptcy led to its shares tumbling (Berti-Equille et al. 2009) (Xiaoxin et al. 2007).

This research will help to improve the quality of information on the web by identifying and eliminating information which is not truthful or which can be categorised as spam.

1.8.1.2 Immune Web 2.0 platforms against Spam 2.0

Spam 2.0 filtering can be a key factor in keeping the Web 2.0 platform clean from junk, non-related and unwanted content, saving resources (e.g. network bandwidth, storage, computation etc), and protecting genuine websites from being infiltrated by spam.

By providing an early-detection method, this research can stop the propagation of Spam 2.0 inside the web, thereby conserving many valuable resources.

1.8.1.3 Provide inputs for Australian Spam Act 2003

Currently, the Australian Spam Act 2003 does not fully cover the vast majority of today's spam concerns. This research will provide inputs for Australian Spam Act 2003 by: *firstly* providing a comprehensive study on the new generation of spam that can be an input for the Australian Spam Act; *secondly* proposing a methodology to be integrated with web application development for spam filtering.

1.8.1.4 Contribution to Australian National Broadband Network (NBN) initiative

The main outcome of this research is intended to combati spam, online fraud and scam. This result is also in line with a *National Research Priority*, namely "Safeguarding Australia" through securing Critical Infrastructure like the NBN Initiative.

1.8.2 Economic

1.8.2.1 Reducing individual, organisations and government monetary costs

In terms of monetary costs, Spam 2.0 constrains web operators to spend time in content moderation and ignoring spam content, network administrators to block spam content by implementing monitoring facilities to track network traffic, web hosting providers to filter out creation of spam pages on their hosting, users to spend more time finding out quality content, governments to investigate the source of spam generation, and search engines to improve their algorithm to avoid spam content.

Briefly, such costs include storage, network bandwidth, computation, and time usage as well as purchasing the latest filtering products and hiring people to manage spam. Some studies (Takemura & Ebara 2008) also show that email spam decreases GDP and productivity. Although other research has not confirmed the cost of Spam 2.0, we believe it far exceeds email spam.

1.8.2.2 Reducing online fraud and scam

As mentioned earlier, spam is a key factor in the propagation of online fraud and scam on the internet. Spam 2.0 facilitates online fraud and scam toward Web 2.0 platforms through fast and anonymous propagation of scam links and web pages. Therefore, it can target more people more quickly. Online fraud and scams highlight the importance of an efficient spam management, specifically Spam 2.0 filtering.

Australians lost over $70 million to online fraud and scams in 2009 (AdelaideNow 2011)**. However, such figures represent only a part of the reported proportion of total scam that occur on today's web. Economically, this research can assist individuals, organisations, government to save Spam 2.0 monetary costs as well as reduce the amount of online fraud and scam.**

1.8.3 Technical

1.8.3.1 Coining a new concept of Spam 2.0

There is no comprehensive study in the current literature on the new generation of spam in Web 2.0 applications. The major works on Spam 2.0 focus on one particular type of spam in web applications (e.g. spam in social network website, spam in comments, and spam in video sharing website etc.). Hence, there is the need to define Spam 2.0 and explicitly describe its particular characteristics. The main technical significance of this research is to introduce and define the concept of Spam 2.0.

1.8.3.2 Proposing an early-detection Spam 2.0 filtering approach

Many existing studies on Spam 2.0 filtering are based mainly on detection methods and do not provide satisfactory results. Other prevention methods such as CAPTCHA decrease user convenience and are about to expire (ref. Chapter 2). Additionally, there is no work on early-detection-based Spam 2.0 filters.

This research provides an early-detection-based Spam 2.0 filter to combat Spam 2.0.

1.8.3.3 Web spambot characterising and behaviour analysis

As mentioned earlier, spambots are automated agents used for Spam 2.0 propagations. Spambots make spam distribution much faster and easier. This research provides a first step in characterising spambot and utilising this knowledge for further spambot detection.

1.8.3.4 Proposing automated early-detection Spam 2.0 filtering approach

This research proposes an early-detection-based Spam 2.0 filtering approach. The proposed approach utilises both rule-based and statistical techniques to exclude automated requests and spambots from Web 2.0 platforms.

1.9 Research Objectives

The main research objective of this study is to develop a framework to study Spam 2.0 and propose a Spam 2.0 filtering solution. This will require an evaluation of the related literature, the development of solutions to specific research issues, the construction of models for these solutions, and the validation of the model using real-world data. Seven research objectives have been derived from the main goal of this dissertation as follows.

Objective 1: Provide an introduction to Spam 2.0 and its unique features

The existing literature focuses mainly on spam filtering in different web applications and does not address Spam 2.0 as a new generation of spam. The first objective focuses on providing an introduction to Spam 2.0 and showing how it is different from other types of spam.

Objective 2: Capture Spam 2.0 generation behaviour by creating a dataset

The current literature lacks any systematic method for tracking, investigating and studying Spam 2.0 generation behaviour. For the second objective of this research, a monitoring framework is proposed to investigate a Spam 2.0 generation model.

Objective 3: Profile spam behaviour (i.e. Spam 2.0)

In this objective, this research aims to characterise Spam 2.0 behaviour and its model. The outcome of this would be used for further Spam 2.0 filtering solutions.

Objective 4: Develop Spam 2.0 filtering mechanism that is extendible to any Web 2.0 platform

In this objective, an early-detection-based Spam 2.0 filtering method is proposed to address current gaps in the Spam 2.0 filtering methods and associated Spam 2.0 filtering issues.

Objective 5: Develop on-the-fly spambot filtering method for early Spam 2.0 detection

In this objective, a practical and spontaneous Spam 2.0 filtering method is proposed to identify spam users on-the-fly and while they are interacting with any Web 2.0 platforms.

Objective 6: Provide a publicly available dataset to serve as a standard benchmark for research on Spam 2.0 filtering

In this objective, two data sets containing Spam 2.0 content and Spam 2.0 behavioural data are proposed. These datasets are publicly accessible online though *Anti-Spam Research Lab* website (http://asrl.debii.curtin.edu.au).

Objective 7: Validate the proposed methods using a number of operational Web 2.0 platforms and evaluate their performance against existing models of Spam 2.0 filtering techniques.

In this objective, proposed Spam 2.0 filtering solutions are compared against existing common Spam 2.0 filtering method. The performance of each method is then compared.

1.10 Structure of Dissertation

The remainder of this dissertation is organized as follows.

Chapter 2 presents a comprehensive survey of recent related research on spam filtering approaches for web spam and Spam 2.0, and provides an empirical comparison of such approaches.

Chapter 3 defines 3 research problems – problems of detection-based, prevention-based, and early-detection based Spam 2.0 filtering methods – based on insights gained from the literature review in Chapter 2. It defines several preliminary concepts that are used in defining solution requirements. Given the clearly-defined problems, issues, and requirements, it then discusses the science and engineering approach that will be used to solve these two problems and address these research issues.

Chapter 4 proposes the Spam 2.0 filtering conceptual framework that addresses the selected research issues. The platform infrastructure is defined based on requirements stated in Chapter 3. Conceptual models are constructed and 7 different steps are provided in order to illustrate the conceptual framework.

Chapter 5 presents a mechanism named *HoneySpam 2.0* to monitor, track and store Spam 2.0. This information is later on analysed and investigated to discover the characteristics of Spam 2.0 and its distribution model. HoneySpam 2.0 is tested against real-world Spam 2.0 data, and its performance is compared with those of existing solutions.

Chapter 6 presents a framework to investigate, formulate, and cluster Spam 2.0 behaviours. Based on initial results of HoneySpam 2.0 (Section 5.8), three feature sets – *Action Set*, *Action Time*, and *Action*

Frequency are presented here to formulate spammers' behaviour on Web 2.0 platforms. Also, a neural network clustering mechanism is used to cluster spammers' behaviour based on the three feature sets.

Chapter 7 proposes the first solution for Spam 2.0 filtering called *early-detection based Spam 2.0 filtering* method (EDSF). EDSF automatically identifies Spam 2.0 submissions from genuine human content contribution inside Web 2.0 platforms. The proposed method studies, investigates and identifies Spam 2.0 in Web 2.0 platforms. EDSF makes use of web usage navigation behaviour to build up a discriminative feature set. This feature set is used later in a machine learning classifier known as a Support Vector Machine (SVM) for classification. EDSF is tested against real-world datasets and its performance is compared against existing Spam 2.0 filtering methods.

Chapter 8 proposes *One-the-Fly early-detection based Spam 2.0 filtering* called (OFSF). OFSF automatically separates Spam 2.0 submissions from genuine human content contribution inside Web 2.0 platforms. The proposed method studies, investigates and identifies Spam 2.0 in the Web 2.0 platform. Similar to the EDSF that is discussed in Chapter 7, OFSF makes use of web usage navigation behaviour to create a discriminative feature set. However, OFSF considers the sequence and order of user web navigation in order to build up the feature set for classification. Unlike EDSF, OFSF uses a novel rule-based classifier to store, retrieve, and separate spam users from genuine human ones. OFSF makes it possible to spot spam users on-the-fly. In other words, while users are interacting with the Web 2.0 platform, OFSF is able to track and identify possible spam navigation patterns. OFSF is tested against real-world dataset and its performance is compared against existing Spam 2.0 filtering method.

Chapter 9 concludes the whole thesis work by summarising what has been achieved and its major benefits, identifying the remaining work that needs to be done, and envisioning future work under this research direction.

1.11 Conclusion

Spam is a major problem for today's web. Spam 2.0 - a new generation of spam that targets Web 2.0 platforms - has spread widely into legitimate websites using automated tools. Current literature has not covered this type of spam in depth and the outcome of efforts to date are inadequate. The aim of this research is to formalise a definition for Spam 2.0 and provide Spam 2.0 filtering solutions. Early-

detection, extendibility, robustness and adaptability are key factors in the design of the proposed method.

Chapter 2

Literature Review

This chapter covers

- Survey and evaluation of current web spam filtering methods.
- Survey and evaluation of current Spam 2.0 filtering methods.
- Discussion of current research gaps in spam filtering domain.

2.1 Introduction

This chapter provides a detailed overview of the literature pertaining to spam filtering as well as giving an evaluation of recent anti-spam methods designed for web and Web 2.0 platforms. As discussed in Section 1.6.4, these methods can be broadly categorised as:

- detection-based (i.e. content-based, statistics-based, and rule-based),
- linked-based (i.e. mainly web spam filtering methods)
- prevention-based, and
- early-detection based

This chapter serves as a review of related works and current literature in the area of web spam and Spam 2.0. This dissertation covers the following studies in the literature:

Web spam – Web spam is referred to as those tactics employed by spammers to increase the visibility of unsolicited websites in search engine results. It involves deception and degrades users' search experiences (Z. Gyongyi & H. Garcia-Molina 2005). Current tactics that are used in web spam along with web spam filtering solutions are covered in detail in this chapter. This chapter also outlines existing open issues.

Spam 2.0 – Spam 2.0 is an infiltration of spam content inside legitimate Web 2.0 platforms. This is a new generation of spam and is the focus of this dissertation (Hayati et al. 2010). A total of 22 Spam 2.0 filtering studies are evaluated and open issues are then outlined.

For each type of spam, we present a preliminary list of terms relevant to the literature survey, an explanation of spam filtering approaches, and a critical evaluation of proposed methods are presented.

2.2 Web Spam

This section provides:

- a definition of web spam,
- an overview of web spam preliminary knowledge,
- tactics that are being used in web spam,
- a survey of current web spam filtering methods, and
- an evaluation of existing web spam filtering methods.

2.2.1 Definition

Search engines are widely used to extract information from the World Wide Web. By querying various keywords, search engines are able to rank web pages containing the keywords. Search engines employ indexing algorithms to provide web page rankings. As the use of search engines increases in popularity and more traffic is derived from search engines to websites, website operators strive for higher ranking in search engine results, creating an opportunity for both legal and illicit efforts (known as web spam) to increase website popularity.

Web spam refers to those features that website operators or developers would not add to their website if search engines did not exist (Perkins 2001). However, this definition might involve some other legitimate features e.g. *Search Engines Optimisation (SEO)* (ref. Section 2.1.2) used by website developers. In another definition, web spam is defined as the techniques used to improve unsolicited website visibility in search engine results which involve deception as well as degrading user-experience of search results (Z. Gyongyi & H. Garcia-Molina 2005). This form of spam is also known as *Spamdexing* or *Black hat SEO* since they are not appropriate SEO algorithms.

As a rough estimation of the total amount of spam web pages, Alexandros *et al*. (2006) conducted a research on the amount of web spam. The result of their study of 105, 484, 446 downloaded web pages shows that 95% top-eight-level domains are spam. .Biz and .us domains contain almost 70% and 35% of spam web pages respectively.

There have been studies to investigate the motivation behind web spam (Chellapilla & Chickering 2006a)(Yi-Min et al. 2007). The outcome of such studies shows that monetary gain is the main motivation for web spam. Malicious users can earn money by subscribing to *pay-per-click affiliate* programs, selling search ranking for specific keywords etc (Gyöngyi et al. 2004).

Web spam formal definition

Suppose *G=(W,E)* is a model for web where *W* (nodes) is a set of web pages and *E* (edges) a set of direct links among web pages (Gyöngyi et al. 2004). If a web page *p* has more than one link to web page *q*, it would be considered as a single link. *ι(p)* is the number of web pages that have links to p. ω*(p)* is the number of web pages that link from *p*. The former is also referred to as *indegree* or *inlink*, while the latter is referred to as *outdegree* or *outlink*. A web page without inlinks is called an *unreferenced page*. Similarly, a web page without outlinks is called a *non-referencing page*. A web page without either inlink or outlink is an *isolated page*.

$$W = \{w_1, w_2, \ldots, w_{|W|}\} \quad 2.1$$

where w_i is the i^{th} web page in graph *G*.

$$C = \{c_l, c_s\} \quad 2.2$$

where *C* is a set of overall web page class, c_l refers to legitimate web page and c_s refers to spam web page. The decision function φ can be defined as Eq. 2.1.

$$\varphi(w_i, c_j) : W \times C \rightarrow \{0, 1\} \quad 2.3$$

φ is a binary classification function where,

$$\varphi(w_i, c_j) = \begin{cases} 1 & w_i \in c_s \\ 0 & w_i \notin c_s \end{cases} \quad 2.4$$

2.2.2 Preliminary Concepts

This section outlines several preliminary concepts pertaining to web spam and web spam filtering that assist in the understanding of the literature review.

Hypertext Transfer Protocol (HTTP) header

HTTP is a foundation protocol for data transmission in the Internet (R. Fielding et al. 1996). The HTTP header contains information about HTTP request, response, sender IP address, time of request, sender browser identity (user-agent), referrer URL (URL of web page that links to current requested web page) etc.

Search Engine Optimising (SEO)

SEO are techniques that are designed to improve the ranking of a particular website or a web page in search engine indexing results (Perkins 2001). SEO techniques are considered to be legitimate while conforming to search engines guidelines (i.e. creating the content for users and considering their accessibility rather than creating content for search engines). Other attempts to increase visibility of web pages are considered illegal. Such attempts are known as *Spamdexing* or *web spam tactics* and are penalised by search engines.

Search Engine Crawlers

Search engine crawlers, search-bots, and spiders are automated agents that can traverse web pages to collect information (Koster 1993). They are used mainly to create a copy of web pages for search engines for future indexing as well as extracting up-to-date versions of the contents in the websites.

Search Engine Indexing Algorithms or Search Engine Ranking

Search engine indexing algorithms or web indexing refer to scoring techniques utilised by search engines to rank web pages (Perkins 2001). The intention of an indexing algorithm is to create a ranked list of relevant documents for various queries (e.g. keywords). Hyperlink Induced Topic Search (HITS) and PageRank are examples of the most common search engine indexing algorithms that have been used in practice.

Hyperlink Induced Topic Search (HITS)

HITS is a web indexing algorithm that assigns two scores – *Authority* and *Hub* – to each web page (Kleinberg 1999). A web page that is linked to many other web pages is known as a *hub* and comparatively a web page that obtains many links from other web pages is known as an *authority*. The original HITS algorithm has been changed due to its limitation and increase in web spam. The HITS algorithm can operate only on a small subset of data and produces two scores instead of one for ranking web pages.

PageRank

PageRank is a well-known indexing algorithm used by the Google search engine. PageRank is based on the notion that a web page with a high rank obtains many incoming links from other high ranking web pages (Page et al. 1998). This is a recursive idea which substantiates the PageRank of a particular web page, by first considering the PageRank of the other referring web pages.

HITS and PageRank are the most common indexing algorithms. So far, the majority of search engines have not published a specification of many other features that they consider when ranking a web page. Search engines conceal the specifications about their actual indexing algorithm to prevent tampering, specifically through web spam tactics.

True positive, true negative, false positive, and false negative

True positive (TP), true negative (TN), false positive (FP), and false negative (FN) are four measures used to calculate the output performance of any classification method (Alpaydin 2004).

In the spam filtering domain, true positive is the number of correctly classified items that belong to the positive class (e.g. genuine email in user's inbox folder), true negative is the number of correctly classified items that belong to the negative class (e.g. spam email in user's *junk* folder), false positive is the number of items incorrectly classified as belonging to the positive class (e.g. genuine email in user's *junk* folder), and false negative is the number of items incorrectly classified as belonging to the negative class (e.g. spam email in user's inbox folder).

C4.5 Decision tree

C4.5 is a decision tree commonly used for classification. Input to the C4.5 is in the form of vector S_i = (f_1,f_2, …, f_n) where f_i s are features or attributes that describe the sample (Quinlan 1992). The already classified sample is known as training data. At each tree node, C4.5 starts by choosing an attribute that is the most effective in splitting the sample into different classes. This process is followed recursively in order to classify each sample.

In web spam filtering, C4.5 is trained with both spam and genuine data to distinguish spam from real genuine content.

Naïve Bayes classifier

Naïve Bayes is a simple and common machine learning classifier which is used for spam classification (Alpaydin 2004). In spam filtering, the training of naïve bayes with fairly small amounts of sample data results in the probability that a message belongs to a class *C* (i.e. spam or non-spam) knowing that it includes F_1,…F_n features (Eq. 2.5).

$$P(C|F_1,\ldots,F_n) = \frac{P(F_1,\ldots,F_n|C)P(C)}{P(F_1,\ldots,F_n)} \qquad 2.5$$

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a machine learning algorithm designed to be robust for classification, especially binary classification (Chang & Lin 2001). SVM trains by n data points or features {(x_1,y_1),(x_2,y_2),…,(x_n,y_n)} and each feature comes with a class label (y_i). SVM then tries to find an optimum *hyperplane* to separate two classes (e.g. spam and genuine) and maximise the margin between each class. A decision function on new data point *x* is defined as $\phi(x) = \mathrm{sgn}(w, x + b)$ where *w* is the weight vector and *b* is the bias term.

In the spam filtering domain, SVM can be trained with spam and genuine data to distinguish spam content from genuine one.

Language Model and N-gram(s)

A language model is a statistical estimate that models the distribution of natural language (Rijsbergen 1979). The most common model in a language model is *N-gram*. N-gram extracts subsequence length *n* from a given sequence; for sentence *S, the* n-gram model is defined as:

$$P(s) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\dots P(w_l|w_1\dots w_{l-1}) \quad 2.6$$

In the spam filtering domain, language models can be used to model content of spam web pages/articles. Later on, this model is used in input vectors to train classifiers.

Zipf's law

Zipf's law is a mathematical discrete distribution equation that can model the occurrence of some data types (Rijsbergen 1979). Zipf's law is formally defined as follows:

$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)}. \quad 2.7$$

where *k* is the rank, *N* is number of elements and *s* is used to define the characteristics of distribution. Zipf's law can predict the frequency of *k.*

In the spam filtering domain, Zipf's law is used to model some particular behaviour or is employed to measure a threshold for discriminative features.

Precision, Recall and F1-measure

Precision and Recall are two measurement techniques to evaluate the accuracy or correctness of the classification algorithm (Alpaydin 2004). Precision presents the exactness, while recall shows completeness of the result. Precision and recall are defined as follows:

$$P = \frac{tp}{tp+fp} \quad R = \frac{tp}{tp+fn} \quad 2.8$$

It is common to combine precision and recall in order to generate one value (known as *F-Measure*) to compare or evaluate different classification results (Alpaydin 2004). F-measure or *F1-measure* is defined below where P and R are precision and recall values accordingly:

$$F_1 = 2\frac{P \times R}{P + R} \quad 2.9$$

The higher the F1-value, the better is the performance. If the F1-value is close to 0, the result is similar to that of a random prediction. A result value closer to -1 shows the strong inverse ability of the classifier.

Receiver Operating Characteristic (ROC) curve and Area Under the ROC Curve (AUC)

The ROC curve is commonly a plot of true positive against false positive to present the trade-off between these two values for various rates (Alpaydin 2004). If the curve skews to the top left, the result accuracy is better (Figure 2.1). However, if the curve is closer to the diagonal, the accuracy of the classification is closer to random.

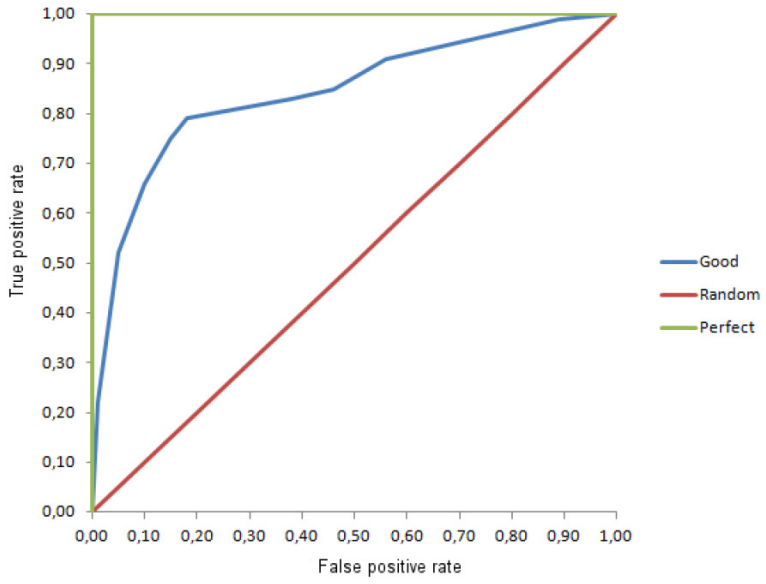


Figure 2.1: Sample ROC curve. The closer the curve is to the left top boundaries, the more accurate is the result.

It is common to measure the *Area Under the ROC Curve* (AUC) to compare different ROCs for different classification tests (Alpaydin 2004). AUC can be measured by simply computing the area under the ROC curve. ROC and AUC are used in the spam filtering domain to measure the performance of classifiers.

2.2.3 Web Spam Tactics

Web spam tactics can be grouped into two classes – content-based tactics (CB) and link-based tactics (LB). The former alter the content or the view of web pages, while the latter takes advantage of link-

based algorithms to give websites higher rankings in search engine results. These tactics are listed below.

2.2.3.1 Content Based

Content-based web spam tactics alter the *look and feel* of web pages in order to obtain higher rankings in search engines results. Content-based tactics can be grouped into 4 different categories: Keyword stuffing, Scraper website, Hidden content, and Cloaking.

Keyword Stuffing

Keyword stuffing refers to the placement of irrelevant keywords on the web page, in order to increase keyword counts of a particular web page (Alexandros et al. 2006). A search engine indexing algorithm counts the number of keywords in a web page to find the level of relevance. Keywords can be placed in the web page body, header, images description attribute etc.

Scraper Websites

The content of scraper websites is stolen from other sources such as search engine results and other legitimate web pages (J. Caverlee et al. 2009). The purpose is to make a quality website gain popularity and later on pass such authority to spam web pages. Occasionally, recently expired domain names that have an already-established search engine ranking are being used for this purpose. Another common technique in scraper websites is *article spinning*. The content that is scraped from other sources is being rewritten manually or automatically in order to avoid content duplication penalties imposed by search engines.

Hidden Content

Hidden content refers to the placement of unrelated text or keywords on a web page with the same colour as the web page background; hence, it can be read by search engine crawlers but is not visible to human users (Alexandros et al. 2006). Similarly, a website can be populated with invisible hyperlinks to other websites. This technique increases the links' reputation.

Cloaking

Cloaking which is the most common web spam tactic, delivers content to search engine crawlers different from human users (Z. Gyongyi & H. Garcia-Molina 2005). The decision to cloak can be

based on visitors' IP addresses, type of Internet browser, users' navigation behaviour etc. With the cloaking technique, a server-side script shows search engine friendly content to a search engine crawler that is different from content visible to the human user. *Mosaic cloaking* and *page hijacking* are the most recent types of cloaking technique (J. Caverlee et al. 2009). The former replaces selected parts of a web page – not the whole web page, with regard to search engine crawler or human users and keeps rest of the web page unmodified. The latter copies popular websites' content. This copy is shown to the search engine crawler while the human is redirected to a different web location or content.

2.2.3.2 Link-based

Link-based web spam tactics involve strategically modifying the way web pages or websites are linked together in order to gain higher rankings in search engine results. Link-based tactics can be grouped into five categories: Redirection, Scraper websites, Linkfarms, and Hijacking.

Redirection

Redirection refers to entrance web pages to other spam web page (Baoning Wu & Brian D. Davison 2005). Redirection web pages are stuffed with relevant keywords and phrases to obtain a higher rank in search engines. Search engines typically do not follow the redirection chain and rank the web page based on keywords in the redirected web page, as opposed to the actual destination web page. This technique is also known as *Doorway web page* which typically has a mechanism which sends visitors to different web pages (Wu & Davison 2005).

Linkfarms

Linkfarm is a group of websites that are highly hyperlinked to each other (J. Caverlee et al. 2009). The more inbound links a website obtains, the higher is the rank that it gains. Although new search engine indexing algorithms employ other ranking features, linkfarm still influences search engine results (J. Caverlee et al. 2009).

Splog

Splog refers to the creation of multiple blogs usually on legitimate blog hosting (e.g. *blogger.com*) to generate hyperlink to spam web pages (P. Kolari et al. 2006). Occasionally, the content of splog is stolen from other sources. The more inlinks a website obtains, the higher will be its rank.

Hijacking

Hijacking refers to linking from legitimate website to spam web pages by distributing links inside public and editable Web 2.0 platforms such as forums, wiki, comment etc (J. Caverlee et al. 2009). Hijacking can also be regarded as a Spam 2.0 tactic where links to spammers' campaigns are propagated in Web 2.0 platforms.

In this chapter, web spam tactics are the criteria against which each web spam filtering method is evaluated. After discussing the preliminary concepts and providing background information, a comprehensive study of the existing web spam filtering methods in the literature is presented in the following sections. A summary of each method along with a comparative analysis of the methods is then discussed.

2.2.4 Web Spam Filtering Methods

This section provides a comprehensive study of state of the art web spam filtering methods based on four different categories: link-based, detection-based, and early-detection based.

2.2.4.1 Link-based Web Spam Filtering Methods

As mentioned in Section 2.2.3.2 , in the link-based approach, web spam filters investigate various link-analysis, graph-analysis or network analysis techniques to separate spam web pages from genuine ones (J. Caverlee et al. 2009). By constructing a graph and looking for different parameters, such methods are capable of detecting spam websites. There has been a significant amount of attention on the link-based approach in web spam filtering (ref. Section 2.2.5). This section covers recent works on link-based web spam filters.

TrustRank is one of the earliest and most cited works for filtering spam web pages in search ranking results (Z Gyongyi et al. 2004). TrustRank assists search engine indexing algorithms to separate reputable and good websites from spam infested ones. It is based on the notion of semi-automatism since an expert initially needs to rank a small subset of websites. This set is considered as an initial *reliable seed* to rank other good pages. Here, the TrustRank of spam websites that are not hyperlinked from good websites is diminished. The result of authors' works shows that by using small seed (200 websites) their method can effectively filter out spam web pages. Eq. 2.10 defines TrustRank score (*TR*).

$$TR = \alpha.TR.\Gamma.(1-\alpha).d \quad 2.10$$

where α is a decay factor (e.g. the score equal for all web pages in the set) , d is a *static score distribution vector* (i.e. this score manually assigned by expert as initial good seed.), Γ is a matrix as defined in Eq. 2.11 and p and q are web pages and E is set of edges. TrustRank suffers from drawbacks as follow:

$$\Gamma = \begin{cases} 1 & (p,q) \in E \\ 0 & (p,q) \notin E \end{cases} \quad 2.11$$

- Similar to PageRank, TrustRank is not effective on the small scale datasets.
- TrustRank is unable to assign adequate scores to web pages without in-links.
- TrustRank fails to detect spam web pages if there is a link from a legitimate web page to the spam one (Gyöngyi et al. 2004) (Asano et al. 2007).
- TrustRank requires manual effort in order to construct the initial trusted-seed set.

Web spam filtering methods in the literature strive to either enhance web pages rankings or discover spam web pages in the search results (or both). Some such methods are inspired by the idea of the PageRank (e.g. TrustRank), some try to improve the HITS algorithm and the rest are more general approaches toward identification of spam web pages.

Given the above drawbacks, (A. Benczúr et al. 2005) proposed a web spam filtering method that tries to resolve some of the problems in the original PageRank and TrustRank approaches by investigating the link exchanges among web pages. One of the main drawbacks of PageRank is that if a website *A* is linked from many low-ranked web pages, PageRank would increase ranking for *A* by aggregating proportion of rankings from low-ranked web pages. To identify such link exchanges and provide a more automated way to identify spam web pages (i.e. without the need for human experts), authors in (A. Benczúr et al. 2005) proposed a technique called *SpamRank*. The main assumption in their proposed method is that PageRank for a spam web page is mainly from many low-ranked web pages (referred to as *supporters*) that point to it. Overall, the SpamRank has three phases as follows.

Phase 1: Selecting the supporters of a given page *i* by using *personalised PageRank (PPR)*. PPR (Eq. 2.12) is a simulation algorithm described in (Fogaras & B. Rácz 2004) that can be interpreted as the probability of a random walk terminating at *i* in the web graph (*G=(W,E)*).

$$\mathrm{PPR}(r) = \epsilon \cdot \sum_{t=0}^{\infty} (1-\epsilon)^t r \cdot A^t \qquad 2.12$$

where *r* is the teleportation vector and is not uniform, ε is the probability of teleportation and *A* is a random walk matrix over a given web graph.

Phase 2: Penalising web pages that originate a suspicious PageRank by measuring the similarity of the PageRank histogram and their power law distribution. As stated in (Adamic et al. 2000) web graph is believed to follow power law distribution.

Phase 3: Applying the penalties on PageRank of the suspicious web pages and their supporters iteratively.

In order to evaluate the proposed method, the authors used a web-graph dataset consisting of *.de* domains which contains 31.2 M web pages. They manually classified 910 sample web pages and divided them equally and randomly to 20 buckets. Each bucket contained 5% of the total PageRank, but the first bucket contained the highest PageRank.

The authors kept the performance measurement of their method as an open question; however, they found that the proportion of spam web pages is higher in the bucket with the highest PageRank value. Therefore, it is observable how much spam impacts on PageRank and the average number of reputable pages.

The main drawbacks of PageRank is that if a website A is linked from many low-ranked web pages, PageRank would increase ranking for A by aggregating a proportion of rankings from low-ranked web pages.

The other drawback of PageRank-based methods is their reliance on a significant amount of data in order to initiate the ranking process. Authors in (Asano et al. 2007) addressed this problem by proposing methods for improving the HITS algorithm that can be run on proportionally smaller scale

datasets. The aim of their proposed methods was to remove *linkfarms* and spam web pages from search engine results. The authors proposed three new factors – *Nameserver*, *Domain name*, and *IP address* to be used with a modified version of the HITS algorithm known as *BHITS*.

BHITS (Zhou & Dang 2007) is an enhanced version of the HITS algorithm designed to handle *mutually reinforcing* problems. Mutually reinforcing problems can be simply stated as *supportive link exchange behaviour between two websites or web pages designed to influence the web pages' rankings.* For example, if a web page is linked from a number of other web pages on the same host, the authority score of the web page unfairly increases. In BHITS, authors incorporated *hostname* when calculating the authority and the hub score for each incoming and outgoing link to overcome the *mutual reinforcing problem*.

'Mutual reinforcing problem' is a term used to describe supportive link exchanges behaviour among the susceptible web pages. Linkfarms, splogs, and occasionally hijacking tactics are real-world examples of mutually reinforcing problems that occur in today's web.

Authors in (Asano et al. 2007) proposed to use the Nameserver, Domain name, or IP address instead of the hostname in the BHITS algorithm and they named their method *N+BHITS*, *D+BHITS*, and *I+BIHTS*. For example, web pages that are linked from the same *Nameserver*, *Domain name*, or *IP address* are counted only once when measuring the web page score. The authors also proposed the *trust-score* algorithm based on the HITS algorithm to score the *reliability* of web pages. Their method is similar to *TrustRank* (Gyöngyi et al. 2004) but with the application for the HITS algorithm. A *reliable hub web page* is defined as a web page that has outgoing links to a *reliable seed* (web page) while a *reliable authority web page* is a web page that is linked from a number of *reliable hubs*. The authors combined the *true-score* algorithm with N+BHITS, D+BHITS, and I+BIHTS and proposed two weighted measures to rank a web page. In their experiment, the authors extracted the top 200 web pages from Google by querying 14 keywords. Their algorithm showed good result for *linkfarms* detection and web pages ranking without the need for a significant amount of resources (TrustRank comparatively requires a significant amount of data for processing.).

However, the proposed method can be executed only at the system run time; hence, it cannot be utilised to pre-process web pages rankings when the system is not online. This is the major drawback of most HITS-based algorithms.

Wu & Davison (2005) conducted another study to provide more relevant search results and enhance the HITS algorithm. The authors proposed a method to remove linkfarms from search engine results. Overall, the idea behind this method was that if a web page is a part of the linkfarm, its popularity would be damaged after using this algorithm; otherwise, it would retain a reasonable ranking.

The method consists of three steps as follows:

- Generate a seed set based on the number of common inlinks and outlinks for each web page. If this number exceeds the given threshold (here 3), it would be inserted in the *seed set* and considered as a spam or bad web page.

- *Badness* value of web pages outside the seed set is evaluated based on the number of outgoing links they linked to the web pages inside the seed set. For a specific web page, if the number of links exceeds the given threshold (here 3) it would be included in the seed set.

- In order to incorporate above value in the web page ranking, the values of the bad web pages in the *adjacency matrix* are deleted or down-weighted. A final matrix is used for ranking web pages.

In order to examine the effectiveness of the method, the authors used two data sets. The first dataset was gathered by means of the HITS algorithm, while the second dataset was obtained from *search.ch* search engine. Manual evaluation was used to measure the accuracy of the algorithm. The result of the algorithm has been compared with both the HITS and the BHITS methods. The relevance of the search results to the queries were 23.6% and 42.8% for the HITS and the BHITS methods. However, this value was 72.6% for the proposed algorithm based on the same datasets.

Although the proposed algorithm performed well in diminishing the effect of linkfarms inside search results, it still suffers from the *mutual enforcement* problem that comes with the original HITS algorithm.

Another approach to web spam filtering that attempts to overcome the mutual reinforcement problem has been studied by (Carvalho et al. 2006). The authors proposed a site-level method for detecting

noisy links (e.g. spam and nepotistic links). The proposed method has been employed to detect various linkfarm tactics as follows.

1. *Mutual reinforcement:* Two algorithms have been proposed here to discover this link exchange behaviour. (1) *Bi-Directional Mutual Site Reinforcement (BMSR)*: this algorithm is based on the number of link exchanges between the web pages of two target websites. If the number of link exchanges exceeds a given threshold (here 2), all links between these two websites would be considered as noisy and would be removed. (2) *Uni-Directional Mutual Site Reinforcement (UMSR)*: this algorithm is based on link density which includes all uni-directional links between web pages of two websites. If the UMSR value exceeds a given threshold (here 250), all links between web pages of two websites would be removed.

2. *Abnormal support in which majority of the links from one website point to another website* by building a chain of websites in between. For detection, the number of links from website *A* to website *B* is divided by the total number of links which point to *B*. If the ratio is above a given threshold (here 2%), all links between A and B (which point to either A or B) would be considered as noisy links and would be removed.

3. *Link alliances in which by hosting wide range of websites and linking them together, website rankings would be influenced.* To discover this type of tactic, all web pages that have originated from a specific web page are investigated. If these web pages are highly connected together, this is an indication that the link structure aims to manipulate the overall ranking. Here, instead of removing links, a susceptivity value is calculated.

The authors studied the impact of their method on the link database of *TodoBR* search engine which includes 12,020,513 pages extracted from Brazilian websites and connected by 139,402,245 links. They also extracted 11,246,351 queries as the test queries. The authors investigated the performance of each single algorithm or combination of them which resulted in 16.7% noisy links.

Until 2008, the web spam filtering methods have strived to detect spam web pages by analysing the hyperlink structure of the web-graph (ref. Section 2.2.1). However, other logical relationships amongst web pages that can be exploited to construct the web-graph have not been investigated (e.g. User Browsing Graph). The methods discussed next utilise such relationships for spam web pages identification.

(Yuting Liu et al. 2008) proposed a web spam filtering method by studying the *User Browsing Graph* instead of hyperlink graph to rank the web pages. The proposed method (known as *BrowseRank)* utilises a User Browsing Graph as a means of measuring web page rank. A user browsing graph is a graph that is extracted from the users' *click-through* navigations on the web; the vertices are the visited web pages and the edges are constructed based on the user navigations among web pages. The authors believe that a User Browsing Graph is more reliable than a hyperlink graph since a User Browsing Graph:

- is not as dynamic as a hyperlink graph,
- contains additional factors that can be used for better web page-scoring (e.g. the length of time users spend on the web pages.).

The authors accumulate user browsing data through *toolbars* that are installed on the users' *Internet browsers*. This data consists of user visited web page, time of visit and type of visit. Type of visit can be a click through or direct input of the web page URL in the Internet browser's *address bar*. A similar approach to PageRank is applied to the User Browsing Graph to discover the importance of each web page.

As a result, BrowseRank demonstrated better results than the PageRank and TrustRank algorithms. Additionally, BrowseRank outperformed each of the others in regards to web spam since BrowseRank is based on a user navigational behaviour graph rather than a hyperlink graph.

Although authors in (Yuting Liu et al. 2008) **mentioned that BrowseRank outperforms PageRank and TrustRank,** (Yu et al. 2009) **conducted a comprehensive study comparing TrustRank, PageRank and BrowseRank on different user browsing graphs, hyperlink graphs and a combination of two datasets. Their findings proved that TrustRank outperforms the others.**

(Yu et al. 2009) exploited User Browsing Graph, Hyperlink Graph and a combination of the two in PageRank, BrowseRank and TrustRank. This is a novel approach because it uses a combination of user browsing and hyperlink graphs in order to rank web pages.

TrustRank algorithm has been applied to the improved hyperlink graph with user access information. The result shows that a combination of user browsing and hyperlink graph gives better accuracy on spam web page detection rather than using each graph individually.

(Yuting Liu et al. 2008) and (Yu et al. 2009) have proposed a method to provide more accurate and relevant ranking for the web pages rather than the elimination of spam web pages from the search results. To cover this issue, (Yiqun et al. 2008) conducted research on the effectiveness of a User Browsing Graph on the spam web pages identification. The authors proposed a statistical classification method based on three features as follows.

- *Search Engine Oriented Visit Rate*: the number of visits originated from search engines divided by the total number of visits for a given web page. Spam web pages are more common in search engine results rather than in the users' bookmarks.

- *Source Page Rate*: the number of times a given web page has appeared as the source web pages divided by the number of times the web page was actually visited by users. Spam web pages are more likely to be destination web pages rather that source web pages.

- *Short-time Navigation Rate*: the number of web pages that users visit in a given website divided by the number of times users visit the website. The frequency of the visits for spam web pages are proportionally less than the legitimate ones since the spam web pages do not provide any useful information for the users.

The authors ran their Bayesian method over 800 million Chinese web pages to evaluate the effectiveness of their proposed method. Results showed 79.26% accuracy in classifying spam web pages.

The proposed methods discussed so far, consider only the link structure of the web-graph in order to rank the web pages or eliminate the spam web pages. However, recent methods have incorporated both the link structure and the content features of the web pages for spam web page identification.

(Abernethy et al. 2010) proposed the first learning algorithm called *WITCH* (Webspam Identification through Content and Hyperlinks) which uses both the link structure and the content features for classifying spam hosts (i.e. the set of web pages that share the same domain name). The authors employ a *graph regularisation* linear classifier on the hyperlink graph and SVM-based classifier on content features. There are two graph regularisation functions which indicate how locality of host's *spaminess* is applied in this method. The first one penalises the square of deviation between predicted values for two specific nodes. In the second one, predicted spam scores are penalised if a node related to the source of a link has lower spam value than the destination node. Although the second method is better, the best choice is a combination of these two approaches, since focusing only on the second one will fail nodes which have spam incoming links and non-spam outgoing links.

To examine the performance of WITCH, the authors utilised *Webspam-UK 2006* spam collection and their graph included 11,402 hosts, 7,473 of which were labelled. The hyperlink graph also included 730,774 triples which showed the number of links between two hosts. In addition, 236 features were extracted and normalized. The proposed method used *transductive* setting whereby the test set is known in the process of training. For all results, the test set was stable with 1,851 hosts.

As a result, the authors showed that a combination of web page content and hyperlink graph gave the best results. The authors also compared their results with *Webspam Challenge*, *Stacked Graphical Learning* and *Transductive Link Spam Detection* (which is a link-based method) results. Results showed that WITCH had the highest AUC in comparison. WITCH increased the AUC from 0.95 to 0.96 which means 20% reduction in ranking error.

The main assumption in the proposed method is that there is no link from the genuine host to the spam host. This assumption can be false in most hijacking and splog tactics where spam hosts are linked from the genuine ones. The method proposed by (Araujo & Martinez-Romo 2010) can overcome this problem by analysing the Qualified Link (QL) and Language Model (LM) of the web pages.

The main idea of their method is that two linked web pages or websites should be semantically related. The authors used LM analysis methods for feature selection based on the content of web pages and used them as the classifier criteria.

In the LM approach, a certain triple set of features is extracted from each two linked web pages - source page and target page. The first set drawn from the source page includes anchor text, URL terms, and surrounding anchor text. The second one drawn from the target page includes title, content of the page, and *meta-tags*. Moreover, two combined features are added: (1) anchor text and URL terms (AU), and (2) surrounding anchor text and URL terms (SU). Finally, there are 14 *cross-referred* features extracted from each web page including anchor text and web page content, surrounding anchor text and title, or anchor text and meta-tags.

In the QL approach, indicators of the spam web pages include an excessive number of *recovered* links rather than the non-recovered links (the recover link refers to the links that can be retrieved in the top ten search engine results), high amount of empty anchor text, high number of incoming links with respect to the number of outgoing links, negative difference between the internal and the external links, and finally, the approximate number of broken links.

In order to evaluate the method, authors used two sets of data *WEBSPAM-UK2006* and *WEBSPAM-UK2007* which contain 77.9 million and 105.9 million web pages respectively. The result of their work based on the SVM classifier achieved an F1-measure of 0.86 in WEBSPAM- UK2006 dataset and 0.40 in WEBSPAM-UK2007 dataset.

The users' judgment must be taken into consideration when classifying spam web pages since it is a subjective issue. A website might be spam for one user but not for other users. This issue has not been considered in the majority of automated (link or content based) web spam filtering methods. The following methods strive to rectify this oversight.

(Metaxas & Destefano 2005) proposed a method for finding untrustworthy websites in the search engine results by utilising social networks. The authors highlight the similarity of the web spam tactics and propagandistic techniques in society. They proposed to use anti-propagandistic techniques to detect spam web pages. The proposed anti-propagandistic method is user-initiated. Users themselves should determine whether or not a web page is trustworthy. By finding other mutually connected websites in the web graph of the particular untrustworthy website, other non-trustworthy websites or spam web pages can be discovered.

The proposed method starts from a specific website which is manually marked as an untrustworthy website. By conducting a *breadth first search* (BFS) for a defined depth *s*, the websites which point to *s* are extracted and a directed graph known as *trust neighbourhood* of *s* is created. After ignoring websites such as the educational institutions, the popular directories and blog sites, bi-connected components (BCCs) which point to s can be found. The BCC component contains websites that strongly support s; hence, they can be considered to be untrustworthy as well.

For experimental purposes, the authors used 8 websites including 6 untrustworthy and 2 trustworthy. The authors investigated BCC's trustworthiness for each site by considering 20% of BCC websites as the sample. The results showed that trustworthiness or untrustworthiness of a BCC highly depends on the starting website. If the starting website is untrustworthy, the majority of the BCC websites are also untrustworthy and vice versa. The proposed method is limited to user-level and authors limit their work to the users' decision for trustworthy and untrustworthy of a particular website. However, the authors did not consider the existence of malicious users within the system who might misleadingly mark an initial spam website as non-spam or vice versa.

Table 2.1 presents a comparative summary of link-based web spam filtering methods that have been discussed in this section.

Table 2.1: A comparative summary of link-based web spam filtering methods.

Method	Vertex type	Edge type	Graph	HITS / PageRank (PR) based	Dataset	Use content features?	Perfor-mance
(Z Gyongyi et al. 2004)	Website	Link	Hyperlink	PR	1000 samples 31M websites	No	Avg. 0.85 Precision / Recall
(Benczúr et al. 2005)	Website	Link	Hyperlink	PR	910 samples 32M websites	No	Future Work
(Wu et al. 2005)	Web page	Link	Hyperlink	HITS	200 samples 20M+2.1M web pages	No	Avg. 0.72 Accuracy
(Metaxas et al. 2005)	Website	Link	Social	-	8 websites	No	-
(Carvalho et al. 2006)	Website	Link	Hyperlink	-	200 samples 12M web pages		Imp. 0.59+ Mean Average Precision.
(Zhou et al. 2007)	Hostname	Link	Hyperlink	HITS	-	No	-
(Asano et al. 2007)	Nameserver, Domain, IP	Link between Nameserver, Domain, IP	Hyperlink	HITS	200 web pages	No	Avg. 0.87 Accuracy
(Yuting Liu et al. 2008)	Website	Users' navigation	User Browsing	PR	5.6M websites	No	Best. 0.87 Normalised Discount Cumulative Gain
(Yiqun et al. 2008)	Web page	Users' navigation	User Browsing	-	800M websites	No	Avg. 0.79 Accuracy
(Yu et al. 2009)	Web page	Link & User navigation	Hyper-link & User Browsing	PR	2646 samples 3B web pages	No	Avg. 0.77 AUC
(Abernethy et al. 2010)	Domain	Link	Hyperlink	-	Webspam-UK 2006	Yes	Avg. 0.96 AUC

Method	Vertex type	Edge type	Graph	HITS / PageRank (PR) based	Dataset	Use content features?	Perfor-mance
(Araujo et al. 2010)	Web pages	Link	Hyperlink	-	Webspam-UK 2006 & 2007	Yes	Avg. 0.63 F1-measure

Technically, link-based approaches have the following drawbacks:

Methods that strive to enhance the PageRank algorithm need significant amounts of resources. This makes them usable only for large scale implementations (e.g. major search engines); they are not efficient on a smaller scale.

Methods that strive to enhance the HITS algorithm cannot be used in offline mode. This might cause system performance issues during the system run time.

In the next section, detection-based (content) web spam filtering methods are discussed.

2.2.4.2 Detection-based Web Spam Filtering Methods

In the detection-based web spam filtering method, the focus is more on the content of the web pages rather than their position in the web graph. By extracting discriminative features from content/header of web pages or HTTP request, filtering methods are capable of detecting web spam tactics (ref. Section 1.6.4.1).

(Fetterly et al. 2004) proposed a method using a statistical approach to identify machine-generated spam web pages. They utilised some heuristics to differentiate spam web pages from genuine ones, which include:

- various features of the website host,
- IP addresses referred to by an excessive number of symbolic host names,
- outliers in the distribution of in-degrees and out-degrees of web pages,
- the rate of structural change of web pages, and
- excessive replication of content.

Both manually and statistically (using *Zipf' law distribution*) they measured the threshold for the extracted features.

In this study, two datasets were used. The first data set consists of 150 million URLs, and the second covered about 429 million HTML pages and 38 million HTTP redirects. The result indicated that web pages which are generated by machines differ from those created by humans with regards to the abovementioned heuristics.

A similar study was conducted by (Urvoy et al. 2008) who examined the non-textual content (HTML) in the web pages rather than content features to identify auto-generated spam web pages. The authors argue that several spam web pages share a similar *look-and-feel* since they have been built using the same generation process. Six different *parsers* have been used to extract textual content from web pages and eliminate structural and styling information of each web page. These parsers parse the HTML page into five different versions: (1) filters the alphanumeric characters, (2) same as previous but also replaces multiple spaces with a single one, (3) extracts only the HTML styling tags, (4) same as previous one but also retains non-alphanumeric characters, (5) retains only the alphanumeric characters, and (6) retains full web page content.

Since the amount of data is large, the authors utilised hashing algorithms (i.e. *locality sensitive hash*: *Broder MinHashing* and *Chikar Fingerprints*) to make fingerprints of each web page and later used them for similarity comparison. By clustering similar web pages, authors also extracted features such as the number of URLs, hosts and domains in each cluster to detect new spam web pages. A 130 GB dataset was used for the clustering. The authors used Webspam-UK2006 dataset to evaluate their proposed method. The best F1-measure value (0.57) was achieved when both the HTML tags and alphanumeric data were incorporated for the clustering.

The identification of cloaked web pages is one of the major challenges in the web spam filtering area, which has not been addressed in the previous two studies. In (Wu & Davison 2005), authors investigated cloaking and redirection web pages.

Their content-based detection method is based on the comparison of three copies of the same web page. In their proposed method, terms and links are extracted from the copies of the web pages and the repeated content is added to the separated lists. If the difference among lists is higher than a given threshold, the web page is considered as a cloaking web page.

The proposed method has been evaluated based on two datasets containing a total of 297,170 URLs along with their copies. The authors identified that about 3% of the first dataset and 9% of the second data set contain cloaking web pages that have already manipulated search engine ranking result.

A similar study has been undertaken in (Chellapilla & Chickering 2006a) by mimicking user-agent field in the HTTP header to obtain different copies of the websites. Their method downloads multiple copies (two) of the websites and then compares the content and structure of each copy. If the difference among copies is bigger than a given threshold, website would be marked as a cloaked website. They used a dataset consisting 3 million URLs out of 10,000 search queries for the experiment. The result of their work shows that the ratio of cloaking depends on the popularity and *monetisability* of search queries - i.e. as the query gains popularity, the number of cloaked attempts increases accordingly.

Various techniques have been utilised to fool search crawlers into generating cloaked web pages. For example, the early search crawlers could not execute embedded JavaScript inside HTML web pages. Hence, JavaScript cloaking techniques could be utilised to show two different versions of a web page to users and search crawlers. The methods described next all try to investigate even newer sophisticated cloaking techniques.

The *Strider Search Ranger System* described in (Yi-Min Wang & Ming Ma 2007) is a system to discover cloaking (e.g. redirection and door way) web pages in the search results. The detection mechanism is based on similarity-based grouping. The process in the system starts by selecting a list of keywords and querying them against the popular search engines to extract top-N URLs. Keywords have been extracted from spam forums, spam URLs, popular keywords etc. A fully-fledged crawler (referred to as *SearchMonkeys*) has been used to visit each URL and bypass possible complex cloaking efforts (e.g. Javascript cloaking) that may be embedded in the spam web pages. Initial URLs along with other URLs in the redirection chain have been stored for similarity analysis. Five types of similarity-based grouping are undertaken for each URL:

- Redirection domains: grouping based on the source URLs.

- Final user-seen web page content: grouping based on the content of destination web page.
- Domain *whois* and IP Address: grouping based on the same IP address.
- Link-query results: grouping based on the overlaps among *inlinks* similarity.
- Click-through analysis: grouping based on the similar redirection chain.

The proposed system was tested on the initial list of 4803 URLs that was extracted from search engines. This list was later expanded to about 6 million web pages.

The main aim of the system is to detect cloaked web pages by following their redirection chain. Some of the heuristics such as "grouping based on similar host IP address" might wrongly classify good websites as spam or vice versa since spam websites can be hosted on multiple web servers with different IP addresses. The authors made use of the proposed system in another work to investigate the nature of redirection chains in more detail (Yi-Min et al. 2007).

A novel study has been done in (Wu & Davison 2006) to identify *semantically* cloaked web pages. The authors utilised a machine learning approach to propose the first automatic method to detect the semantically cloaked web pages. Semantic cloaking is a cloaking attempt in which the difference between each cloaked version is in the meaning of the content rather than the syntax of the content.

The proposed method consists of two steps:

1. A copy of the web page from crawler and a copy of the web page from Internet browser are given to the filtering algorithm. After applying term (the number of differences between dictionary terms) and link rules (the number of unique links for two copies), if the value is above a given threshold (here 3), the web page would be passed to step 2.
2. One more copy of the web page is downloaded and used in the SVM classifier. These copies (4 copies) are classified by SVM as either cloaking or non-cloaking web pages.

For the evaluation, authors accumulated 257 unique queries from, *Google Zeitgeist*, *Lycos* and *Ask Jeeves*. For each query, 200 top web pages were obtained from Google search results. For training, a labelled list of 1,285 URLs along with 539 and 746 positive and negative examples of semantic cloaking were used in the classifier. The result shows 93% and 85% for precision and recall metrics respectively.

To examine the performance of the proposed method, the authors used 4,378,870 URLs of 2004 Open Directory Project (ODP) RDF file. It was concluded that over 1% of ODP web pages contain semantic cloaking (91.5% precision and 82.7% recall).

Apart from the above methods which focus only on cloaking detection, (Alexandros et al. 2006) addressed the problem of classifying spam web pages based on the content of the web pages. The authors investigated 10 content-based features such as the number of words, the words' average length, the portion of anchor text, the amount of visible content, the compression ratio, the percentage of popular texts in the web page, the amount of widespread words, and the probability of independent and conditional n-grams.

The authors constructed a *C4.5 Decision tree* for the classification. The decision-tree classifier detected 82.1% and 97.5% of all spam and all non-spam web pages respectively. They also utilised bagging and boosting techniques, to improve their classifier which result in 86.2% accuracy of spam web pages detection. Their evaluation dataset contained 17,168 web pages with the proportion of 2,364 spam web pages.

A similar study has been undertaken in (Pranam Kolari et al. 2006) to identify splogs (P. Kolari et al. 2006). The authors trained SVM classifier with 3 features – *bag-of-anchors*, *bach-of-urls*, *bag-of-n-gram*. Bag-of-anchors are extracted from the anchors text inside blog pages. Bag-of-urls are split (tokenized) based on the special characters inside URLs such as "/", "?". Bag-of-n-gram is a subsequence of *n* character from the given words (here 4-grams).

The authors made use of SVM. The dataset contained 5 million blog homepages with 2,600 blogs as samples. The result of their work showed 88% accuracy (F1-measure) in detecting splogs. However, the result has a considerable number of false-positives.

Another content-based approach has been studied in (A. A. Benczúr et al. 2008) to classify spam websites inside *web archivist*. This work adopted a novel approach in that it applied a content-based detection method to the web archivists. Web archivists create and store snapshots of web pages. The ensemble classifier in the proposed method was trained through content (e.g. terms inside web page) and link (e.g. web page outlinks) features.

Two datasets were used – Webspam-UK 2006 and 2007. The authors achieved the AUC value of 0.84 for the result of their classification.

Table 2.2 presents a comparative summary of detection-based web spam filtering methods that have been discussed in this section.

Table 2.2: A comparative summary of link-based web spam filtering methods.

Method	Content features	Non-content features	Usage of Link features?	Dataset	Classifier	Performance
(Fetterly et al. 2004)	Web page	Host features	Yes	150 + 429M web pages	-	Avg. 0.14 False-Positive
(Baoning et al. 2005)	Terms and links inside the web page	-	No	297K web pages	-	Avg. 0.74 F1-measure
(Wu et al. 2006)	Terms and links inside the web page	-	No	47K web pages	-	0.93 & 0.85 Precision & recall
(Chellapilla et al. 2006)	Web page	-	No	3M web pages	-	-
(Alexandros et al. 2006)	Web page content	-	No	17K web pages	C4.5	Avg. 0.82 Accuracy
(Pranam et al. 2006)	Blog page	-	No	2K samples 5M blogs	SVM	Avg. 0.88 F1-measure
(Yi-Min et al. 2007)	Web page	IP and host	Yes	6M web pages	-	-
(Urvoy et al. 2008)	Web page	HTML structure	No	150GB URLs + Webspam-UK 2006	-	Avg. 0.57 F1-measure
(Benczúr et al. 2008)	Content and links inside the web page	Link graph, Robot meta data, etc.	Yes	Webspam-UK 2007	SVM	Avg. 0.84 F1-measure

Overall, detection-based approaches are afterthoughts. The methods deal with the web spam problem after spam web pages have already infiltrated the web. Hence, these methods cannot prevent the distribution of web spam from wasting resources on the web.

The next section offers an evaluation of early detection-based web spam filtering methods.

2.2.4.3 Early Detection-based Web Spam Filtering Methods

Early detection-based web spam filtering methods investigate specific features such as meta-data, and HTTP header information in websites to detect spam web pages. This section discusses early detection-based web spam filtering methods.

A real-time spam web page detection technique was proposed in (Steve Webb et al. 2008). The authors proposed a classification method based on HTTP session (header) information which can be implemented inside client application (e.g. web browsers) to detect spam web pages before the actual content is transferred.

HTTP sessions contain information such as the request and the response messages from and to the web server. After the client (web browser) sends a HTTP request message to the web server, the web server responds to that request with a HTTP response message (i.e. status, timestamp, content length, web server IP address etc) along with the web page content. The authors proposed a method to investigate the HTTP response message to identify spam web page. Features such as timestamp, content-length, status etc. in HTTP response are used for classification.

The authors utilised 40 different machine learning algorithms and select the best result for classification purposes. Their data set consisted of over 750,000 web page samples. The result showed F1-measure value of 93.5% with accuracy value of 93.9%. The authors also investigated the resource usage and computational costs of their proposed method and claim that their method can save an average of 15.4 KB of bandwidth and storage resources for each spam web page. It adds an average of 101μs for each web page retrieval.

Research in the area of early detection-based web spam filtering methods has not received much attention from the community. The method discussed above is the most recent work in this field. The next section provides detailed evaluation of all the methods for web spam filtering.

2.2.5 Evaluation

In the previous section, a comprehensive survey of web spam filtering methods was presented. In this section a critical evaluation of those methods is discussed in detail.

The majority of the proposed web spam filtering methods attempt to counteract specific web spam tactics. Therefore, it is interesting to see the success of state of the art methods in combating current web spam tactics. This provides a good benchmark to evaluate current literature and discover the research gaps. Table 2.3 presents an evaluation of web spam filtering methods based on various web spam tactics that were discussed in Section 2.2.3 . The hijacking method has been removed from the evaluation list since none of the current web spam filtering methods is specifically designed to address this tactic. Mainly, hijacking is the focus of Spam 2.0 filtering methods (Section 2.3.4). The general column is reserved for the methods that aim to detect spam web pages in general.

Link-based web spam filtering methods have mainly addressed linkfarm tactics. However, splogs can also be employed to generate linkfarms; hence such methods might be applicable for identification of splogs. The link-based method presented by (Yiqun et al. 2008) might be capable of detecting redirection attacks as well as linkfarms and splogs. The other method presented by (Metaxas et al. 2005) utilises users' knowledge into classification of spam web pages. This method can be employed to detect any type of spam web page regardless of tactic.

In the detection-based web spam filtering methods, the majority of the techniques focus on identification of cloaking and redirection tactics. Some of the methods such as (Fetterly et al. 2004) or (Alexandros et al. 2006) are not focused on specific web spam tactics and hence they can be used to detect spam web pages in general.

The state of the art prevention-based and early detection-based methods have not focused on specific type of web spam tactic, but propose to detect spam web pages in general.

Table 2.3: Evaluation of state of the art web spam filtering methods *.

Method	Cloaking	Linkfarms	Splog	Keyword Stuffing	Hidden Content	Redirection	Scraper Websites	General
(Fetterly et al. 2004)	-	-	?	?	?	-	?	×
(Gyongyi et al. 2004)	-	×	?	-	-	-	-	-
(Benczúr et al. 2005)	-	×	?	-	-	-	-	-
(Metaxas et al. 2005)	-	-	-	-	-	-	-	×
(Wu et al. 2005)	-	×	?	-	-	-	-	-
(B. Wu et al. 2005)	×	-	-	-	-	×	-	-
(Carvalho et al. 2006)	-	×	?	-	-	-	-	-
(Chellapilla & Chickering 2006b)	×	-	-	-	-	-	-	-
(Alexandros et al. 2006)	-	-	-	-	-	-	-	×
(Wu et al. 2006)	×	-	-	-	-	-	-	-
(Pranam et al. 2006)	-	-	×	-	-	-	-	-
(Zhou et al. 2007)	-	×	?	-	-	-	-	-
(Asano et al. 2007)	-	×	?	-	-	-	-	-
(Yi-Min Wang & Ming Ma 2007)	×	-	-	-	-	×	-	-
(Goodstein et al. 2007)	-	-	-	-	-	-	-	×

Method	Cloaking	Linkfarms	Splog	Keyword Stuffing	Hidden Content	Redirection	Scraper Websites	General
(Yuting et al. 2008)	-	×	?	-	-	-	-	-
(Webb et al. 2008)	-	-	-	-	-	-	-	×
(Benczúr et al. 2008)	-	-	-	-	-	-	-	×
(Urvoy et al. 2008)	-	-	?	?	?	-	?	×
(Yiqun et al. 2008)	-	×	?	-	-	-	-	-
(Yu et al. 2009)	-	×	?	-	-	-	-	-
(Abernethy et al. 2010)	-	×	?	-	-	-	-	-
(Araujo et al. 2010)	-	×	?	-	-	-	-	-

* *Cross* sign: the method can counteract the web spam tactic (the authors have specifically mentioned this in their study). *Question* mark: the method might counteract the web spam tactic (the authors have not mentioned this in their study).

Figure 2.2 illustrates the coverage of web spam filtering methods in terms of web spam tactics. It is clear from the figure that research attention in this survey has been directed towards splog, linkfarm, general websites and cloaking tactics.

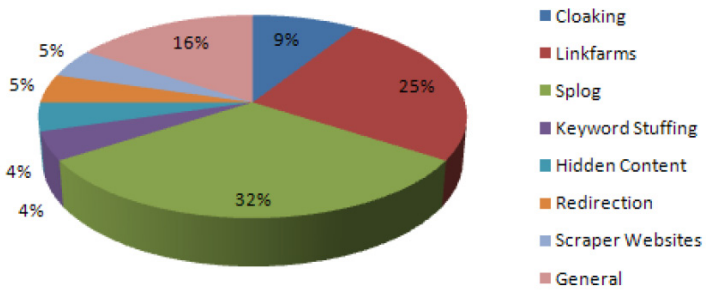


Figure 2.2: Coverage of state of the art web spam filtering methods on web spam tactics.

Figure 2.3 presents the number of web spam filtering methods under each technical category in this survey. It is clear that most of the efforts in the web spam filtering domain have focused on link-based web spam filtering. There is scant literature on the prevention and early detection-based methods.

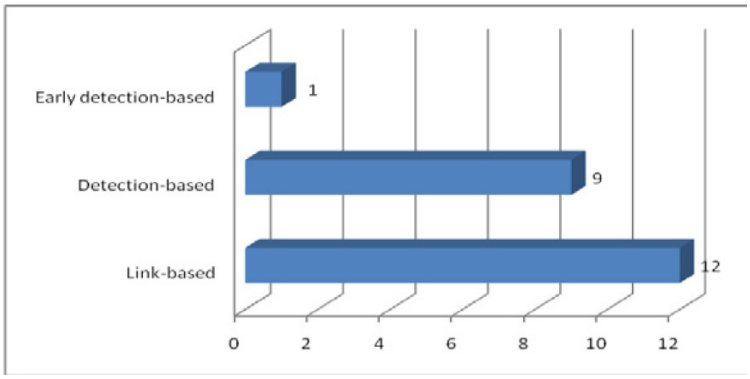


Figure 2.3: The number of web spam filtering methods per technical approach.

Technically, the methods focus mainly on link- and detection-based approaches, and they raise the following concerns:

1. Methods that strive to enhance the PageRank algorithm need significant amounts of resources. This makes them usable only for large scale implementations (e.g. major search engines); are not efficient on a smaller scale.

2. Methods that strive to enhance the HITS algorithm cannot be used in offline mode. This might cause system performance issues during the system run time.

3. Both link- and detection-based approaches are afterthoughts. The methods deal with the web spam problem after spam web pages have already infiltrated the web. Hence, it cannot prevent the distribution of web spam and the wasting of web resources.

4. Early detection-based methods are not effectively designed to prevent web spam distribution. Moreover, there is no study on prevention-based web spam filtering methods.

5. Hijacking is one of the major tactics that has recently been used by spammers. This tactic has not specifically been covered by state of the art web spam filtering methods.

6. The majority of web spam filtering methods have addressed link-based tactics; however, content-based tactics have not received much attention.

It should be noted that the focus of this dissertation is on Spam 2.0 filtering rather than web spam filtering. The issues that have been discussed so far are mainly associated with web spam filtering methods. Spam 2.0 produces some of the abovementioned issues as well as a set of new problems. The following sections provide an in-depth evaluation of Spam 2.0 filtering methods.

2.3 Spam 2.0

This section provides:

- a definition of Spam 2.0,
- an overview of Spam 2.0 preliminary knowledge,
- tactics that are being used in Spam 2.0,
- survey of current Spam 2.0 filtering methods,
- and finally, an evaluation of existing Spam 2.0 filtering methods.

2.3.1 Definition

Web 2.0 is a term associated with read and write web where users can contribute to content generation. Facilities created with Web 2.0 give users a place to collaborate, share information and generate content. Some examples of Web 2.0 include blogs, wikis, forums, online communities (Figure 2.4). Any type of web application which supports user-content generation can be included in the Web 2.0 platform.

Figure 2.4: Examples of Web 2.0 platforms.

However, Web 2.0 platforms have not been immune from malicious usage i.e. spam. A malicious user can sign in to such platforms in order to distribute spam content. Later, the spam content is seen by both genuine users and search crawlers. The spam content on a legitimate website is more trusted than an isolated spam web page on a different website. Examples of Spam 2.0 include: a fake eye-catching profile in the social networking websites, a promotional review, an unsolicited response to a thread in an online forum and a manipulated wiki page etc.

Spam 2.0 is different from previous types of spam such as web spam. Spam 2.0 infiltrates legitimate web pages. Spam 2.0 is more trustworthy and highly influential for users. It is also more challenging for search crawlers to prune spam from genuine content in legitimate websites (Hayati et al. 2010).

As mentioned in Section 1.5.1, Spam 2.0 is defined as the “propagation of unsolicited, anonymous, mass content to infiltrate legitimate Web 2.0 applications” (Hayati, Potdar, Sarenche, et al. 2010).

Similar to other spam filtering methods, Spam 2.0 filtering methods search for spam patterns inside the spam content or the spam distribution behaviour. However, some efforts have been made to prevent Spam 2.0 distribution as well.

2.3.2 Preliminary Concepts

In this section, some preliminary knowledge and concepts are discussed that would be useful in understanding the relevant literature in the area of Spam 2.0.

Web usage data

Web users navigate through web objects such as web pages, image, files, and documents in the websites. Such navigation can be recorded and later mined to extract valuable information (Cooley et al. 1999). This tracking data is referred to as ‘web usage data’. Web usage data has been widely employed in many studies to understand visitor browsing patterns, make personalised websites, improve web performance, and to implement decision support and recommendation system.

In Spam 2.0 filtering, web usage data has been used to identify web crawlers and automated bots.

Social networking websites

Literature on Spam 2.0 filtering methods has paid a lot of attention to spam filtering in social networking websites. Social networking websites provide a platform for people to share their thoughts, exchange messages, photos etc. An example of a social networking website is *Twitter* (www.twitter.com). *Twitter* is a popular micro-blogging service that allows users to share a short message (called a *tweet*). Tweets must contain no more than 140 characters. Twitter allows users to create their own public profile where their tweets appear. Users can subscribe (called *followers*) to other users’ profiles to receive updated tweets.

Social networking websites such as Twitter have received much attention from the Spam 2.0 filtering community and many filtering methods have been proposed for this service.

Blog

Blog or web-log is a type of Web 2.0 platform that individuals maintain for various purposes. For example, blogs can be an online diary, photo gallery, and music sharing platform. As most blogs provide a read/write platform where blog operators can publish their desired content and blog visitors can submit their comments, blogs receive attention from both genuine and malicious users. For example, spam users comment spam content on individuals' blogs.

Public wikis (e.g. Wikipedia), public forums, social networks (e.g. Facebook, delicious etc), organisations' websites etc. are some examples of legitimate hosts that can be infiltrated by Spam 2.0. Almost any website that provides a Web 2.0 platform can be infiltrated by Spam 2.0.

Honeypot

Honeypot is a technique in computer security to trap the adversary by appearing to be a real computer, file, record, and entity in a computer network (Spitzner 2002). In the spam filtering domain, honeypots are used to appear as an email server or mimic a real user in online communities to track spammers' behaviour.

Information gain and Chi-Squared (X^2)

Information gain is frequently used in machine learning for term selection (Michalski et al. 1985). Information gain considers both the existence and non-existence of a term in the content. In the spam filtering domain specifically, detection-based methods information gain is used to reduce the dimension of terms used in an input vector.

Similar to information gain, in the spam filtering domain, X^2 is used in statistical language modelling of word associations and term selection. However, X^2 measures the lack of independence between a term and its class (i.e. term belong to spam or genuine class).

Cosine similarity

Cosine similarity is used to calculate the similarity between two vectors by measuring the cosine angle between them. Cosine similarity can show whether or not two angles are pointing in the same direction (Steinbach et al. 2000).

In spam filtering, cosine similarity can be used to measure the similarity between two content vectors.

Linear regression

Linear regression is a method used to model the relationship between two scalar values. It is used to estimate the linear function to generate data based on the scalar values (Yan & Su 2009).

In spam filtering, linear regression is used as a machine learning tool to predict the linear function based on the training set of scalar values. This linear function later is used to separate spam values from legitimate ones.

K-nearest neighbours

K-nearest neighbour (k-NN) is a machine learning algorithm to discover the K closest object to training samples. Training samples are vectors that are labelled. Using k-NN, unlabelled vectors are classified based on the training samples classes (Alpaydin 2004).

k-NN in spam filtering is used to distinguish spam objects (e.g. content, behaviour) from genuine ones.

Back propagation neural networks

A back propagation neural network is a multi-layered feed forward neural net that adapts the network based on a known set of inputs and output. This algorithm is trained based on known input and desired output. Once the algorithm learns *the problem*, it then estimates outputs for the unpredicted inputs (Michalski et al. 1985).

In spam filtering, the back propagation neural network is used as a machine learning tool for the classification.

Similar concepts that have been discussed in Section 2.2.2 include Precision, Recall, F1-measure, SVM, Naïve Bayes classifier, True positive, true negative, false positive, and false negatives – all of which have been utilised in the Spam 2.0 domain.

2.3.3 Spam 2.0 Tactics

Spam 2.0 is employed by a spam user to distribute content via Web 2.0 platforms. Depending on the structure of the Web 2.0 platform, spam users can propagate spam content in different sections of the website. The following are examples of common strategies that spam users employ to infiltrate web applications with spam content.

Spam online profile (spam profile)

Profiles are a type of *About me* web page that summarise information about particular users. A typical profile contains information about the user's contact details, photo, and latest activities in a Web 2.0 platform.

With this tactic, spam users register/sign up in the Web 2.0 platform (e.g. forums, wikis, social networking website etc.) and fill the profile fields with irrelevant/spam content. For example, in most of the profiles, there is a field for the user's homepage link. A spammer uses that field to direct a genuine user to his/her spam campaign.

Occasionally, spam profiles are filled with appealing content (e.g. celebrities' photos) to attract or mislead genuine users.

Spam comment (comment spam)

With this tactic, comment systems (e.g. blog comments) are targeted by spam content. Spam users fill up comment forms with their spam content. Spam content typically contains URLs to spam campaigns. Spam comments might be easy to discover or quite misleading where normal human users cannot identify them. Figure 2.5 illustrates a misleading comment spam on the author's homepage. This comment has a link to a spam website in the comment's starter name i.e. *maladireta*.

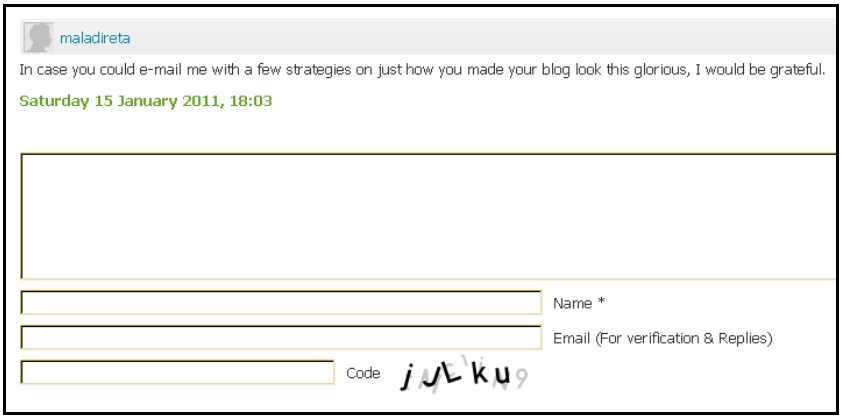


Figure 2.5: Example of misleading comment spam on author's homepage.

Spam wiki (wiki spam)

Wikis provide a platform where users can create a number of web pages and provide links among them. In this tactic, wiki pages are manipulated by spam users to present spam content or a link to spam campaigns. Figure 2.6 presents an example of spam in wiki. A link to the "Arena51" website is modified to a spam website.

Figure 2.6: Example of spam in wiki platform.

Spam personal message

Web 2.0 platforms such as forums, online communities etc. provide a messaging system whose users can exchange personal messages amongst themselves. This tactic allows a personal messaging system to be infiltrated by spam users to send spam content to other genuine users.

Spam online community (social spam)

With this tactic, online communities or social networks are infiltrated by spam users to create spam profiles, sending spam messages to real human users, spam comments etc. Depending on the facilities provided by an online community, they can be misused by a spam user to propagate spam content.

Spam specific website/web application (targeted spam)

With this tactic, spam users target a Web 2.0 platform on a particular website. For example, Twitter is targeted by many spam users. Spam users distribute spam tweets or create spam profiles on Twitter.

Additionally, some popular web applications specifically, *open-source* or free web applications, are also targeted by spam. Web operators use these web applications packages to rapidly set up online collaborating platforms, thereby receiving much attention. Once common features and facilities have been targeted by spam in these web applications, spam users can distribute spam content across all websites that use the same web application. Simple Machine Forum (SMF) (http://www.simplemachines.org), phpBB (http://www.phpbb.com), and Wordpress (http://www.wordpress.org) are examples of common open source web applications.

Spam 2.0 tactics would be used as the criteria to evaluate each Spam 2.0 filtering method. After discussing the preliminary concepts and background information, a comprehensive study of the existing Spam 2.0 filtering methods in the literature is presented in the following sections. A summary of each method, along with a comparative analysis of the methods, is then discussed.

2.3.4 Spam 2.0 Filtering Methods

This section provides a comprehensive study on state of the art Spam 2.0 filtering method based on three different categories detection-based, prevention-based, and early-detection based.

2.3.4.1 Detection-based Spam 2.0 Filtering Methods

In detection-based Spam 2.0 filtering methods, the focus is on identifying spam patterns inside the content of Spam 2.0. The techniques employed here are similar to those described for web spam.

A content-based method is proposed by (Narisawa et al. 2007) to detect spam messages in forums. The authors believed that spam messages are designed to distribute advertisement messages in forums.

The authors started by creating *equivalence classes* for the content of the message. An equivalence class consists of the same occurrences of substrings within a message. By employing three features – length (length of representative of spam equivalence class is higher), size (size of representative of spam equivalence class is higher) and *Maximin* (length of the representative and the length of the longest minimal element of the equivalence class), authors could distinguish spam messages from the legitimate ones.

They evaluated their proposed method based on four Japanese forums (24,591 messages) and used the F1-measure to evaluate the performance of their proposed method. The result of their work varied from 68% to 80% in all four datasets.

Similar to the work mentioned above, authors in (Uemura et al. 2008) proposed a content-based method to detect spam in forums and blogs. They claim that the document complexity of normal messages is much higher than that of spam since spam messages are auto-generated and their generation costs are lower. The authors focused on detecting *Mutation* (swapping random letter in the document), *Crossover* (exchanging a random part of a pair of documents) and *Word Salads* (replacing a random word in the document).

The authors used the same four datasets that had been used in (Narisawa et al. 2007). The result showed that the proposed method works well on word salad detection.

Each and every Web 2.0 platform provides different facilities for users. For example, in forums, users can reply to other users' topics while in wikis, users can edit others' contributions. The variety of such facilities has made Spam 2.0 filtering more challenging since it requires adaptable solutions.

Apart from the abovementioned studies that target Spam 2.0 in specific web applications (e.g. blog and forum), The majority of recent work on Spam 2.0 filtering focuses on Spam 2.0 filtering in particular websites (e.g. Twitter, Facebook, MySpace etc). Among such websites, Twitter has received considerable attention from the research community. The methods which will be discussed

next have focused on various features of the Twitter platform in an attempt to identify spam messages or spam profiles.

(Benevenuto et al. 2010) focus on detecting malicious users who post spam tweets inside Twitter. Their content-based method is based on extracting discrimination features from the content of tweets and spam profiles. After analysing the content of tweets, 39 features are extracted as describers of the tweets. Usage of more URLs in each tweet, bigger proportion of spam words, and *hash tags* (keywords that refer to other topics in Twitter) were the three best features for identifying spam profiles. For example, if spam profiles intend to have more *followees*, they need to become *followers* of an excessive number of members. As they cannot create a balance between number of followees and followers, they end up having a greater number of followers than followees compared to those of genuine users.

For the dataset, the authors crawled 54 million Twitter profiles. They manually labelled the dataset that resulted in 8,207 profiles of which 355 were spam profiles and 7,852 were genuine profiles.

SVM is applied for classification tasks. Chi-Squared (X^2) and Information Gain methods are used for feature selection. Apart from the above features, some attributes that are related to the tweet content are also extracted. For instance, the number of hash tags, number of URLs, number of characters, etc are considered. In total, the classifier managed to identify 70% spam users and 96% genuine ones.

The above study was extended by (Irani et al. 2010) who proposed a method for detecting unrelated tweets in *trending topics* whereby they considered both content of tweets and their associated website. Twitter lists popular topics named *trending topics* in its homepage. Since trending lists are growing in popularity, they have attracted malicious users and spam content. The proposed method has the following three steps.

- Tweets classification based on content of each tweet to find out whether or not it is related to their trends topics.
- Tweet classification based on the content of the web page to which they are referring.
- The results of both the abovementioned steps are combined using ‘AND’ and ‘OR’ operators. This step is incorporated in order to detect spam tweets which look legitimate in content or have a few characters for classification.

For simplicity, tweets created by suspended accounts are assumed to be spam. The authors examined the performance of three classifiers: Naïve Bayes, C4.5 decision trees and *Decision Stumps*. For the dataset, they collected 1.3 million tweets from 2,067 trends along with 1.3 million web pages related to tweets. For tweet text, Naïve Bayes and C4.5 decision trees performed well. Overall, the OR operator had better performance for the combined classifiers.

Instead of finding spam tweets in Twitter, work has been done to identify spam profiles which are used to distribute spam tweets. By eliminating spam profiles, spam tweets can be ignored as well.

Similar work has been done in (A. H. Wang 2010) (A. Wang 2010) to detect spam profiles. However, the authors created a social graph to model users and spam users in Twitter. A directed social graph is constructed to model four relationships: follower, friend, mutual friend, and stranger in users' profiles. Spam detection is done based on (1) graph-based and (2) content-based features. Former features are derived from a social graph that has been constructed for each user account e.g. frequency of friends (indegree), followers (out degree) and reputation (ratio between indegree and outdegree). Later, features are focused on the content of tweets e.g. black-listed URLs, duplicate tweets, number of reply tweets, and unrelated posts having same trending topic. Results showed that the reputation of spam profiles is low since they have excessive numbers of followers compared with the number of friends, and a high number of duplicate tweets is an important factor in detecting spam profiles.

The author first created a collection containing 25,847 users, 500,000 tweets, and almost 49M relationships by Twitter's API methods in three weeks. Evaluation was carried out on 500 manually labelled dataset. The author chose to use the Naive Bayesian technique as a classifier. The classifier was able to correctly identify 89% spam profiles in the test collection.

The abovementioned studies focus only on the Twitter platform. However, in the study of (K. Lee et al. 2010), the authors proposed an automatic honeypot-based method for detecting spam profiles in the online social communities. The goal of this method is to use honeypots in order to decrease the human participation in spam profile classification, create a statistical user model to distinguish spam profiles from legitimate ones and finally detect spammers in a large number of unknown profiles. This method proposes the following solution.

If a honeypot receives a friend request from a user (in the online communities users can add others as their friend), it would consider the sender of the request as a potentially suspicious user and starts to

extract information from the user's profile. Next, these features are coupled with some legitimate profiles and considered as the training set of a classifier. By using a human inspector who verifies the predicted spam profiles, new instances are added to the training set in order to have a more powerful classifier and subsequently better predictions.

The effectiveness of this method was studied over two social communities, Twitter and *MySpace* (http://www.myspace.com). The authors randomly selected 388 legitimate profiles from MySpace and 627 spam profiles from the set of profiles detected by honeypot as being spam. After extracting features such as the number of friends, marital status, etc., it was shown that all top ten classifiers for MySpace had good performance with accuracy over 98.4%, F1-measure over 0.98 and false positive rate less than 1.6%, which was robust against different training sets. They also investigated the discrimination power of extracted features using an ROC curve. It was shown that marital status and gender are the least discriminative features, while bag-of-words and features related to the "about me" section are the most discriminative ones.

To evaluate spam classification in Twitter, authors randomly selected 104 legitimate users. They also considered two classes for this step: spammers and promoters which had 61 and 17 users respectively. Information about user profile, tweets, following and followers were gathered for all these users. Features such as longevity of account in Twitter, average number of tweets per day etc. were also extracted. Content similarity between pairs of tweets was also calculated using standard *cosine similarity*. When investigating discrimination power, average posted tweets per day, percentage of bidirectional friends, ratio of the number of following and followers were found to have the least discrimination power, whereas account age, text-based features extracted from tweets, and number of URLs were found to be the most powerful discriminating features. Furthermore, in the case of content similarity, spam tweets were more similar than others while promoters' tweets had the most number of URLs. Using these features, the top ten classifiers again had good performance for Twitter with accuracy over 82.7%, F1-measure over 0.82 and a false positive rate less than 10.3%.

(Zinman & Donath 2007) is another work that has been done on MySpace to detect spam profiles. However, the result of this work was not satisfactory due to the nature of the features authors utilised. The authors proposed an approach to detect spam users in Social Networking Services (SNS). They utilised features of senders' profiles as well as network features. In SNS, not all the unsolicited messages are spam, so the authors tried to identify the sender instead of the sent messages. In

addition, making a decision about requests is extremely subjective. In other words, the desirability or undesirability of a sender depends on receivers' opinions.

MySpace profiles were examined in this work based on two dimensions: sociability and promotion. Sociability refers to the number of social activities in MySpace, while promotion is related to amount of information that has effect on users. Based on these two dimensions, four prototypes have been created as follows.

- Prototype 1 (low sociability and low promotion): includes new members of the site as well as approximately inactive spam profiles.
- Prototype 2 (low sociability and high promotion): here profiles are promotional and use SNS as a marketing means.
- Prototype 3 (high sociability and low promotion): this prototype is related to an ordinary user and most of the SNS users belong to this group.
- Prototype 4 (high sociability and high promotion): unlike prototype 2, here entities have individual relationships with their friends.

The authors randomly selected several profiles and assigned values from one to five, to profiles' sociability (s) and promotion (p). They considered only those profiles with s or p values higher than one.

They examined four machine learning algorithms - *linear regression, K-nearest neighbours, back propagation neural networks*, and naïve Bayesian networks - with different sets of features: profile-based, network-based, and mixed. They had 40 feature dimensions as well as 600 training data points and 200 test data. The best performance was obtained using a mixed set of features in neural networks with a single layer. On the other hand, network-based tests had poor performance in comparison with profile-based ones which was 78%-83%.

Apart from MySpace and Twitter which were the focus of the previous studies, (H. Gao et al. 2010) proposed a method to detect spam campaigns in online social networks specifically on Facebook (http://www.facebook.com). The authors proposed to use text-based similarity and URL correlation between different *wall posts* (wall posts in Facebook are spaces on each user's profile, where known users can post information for the other users). Similar posts have been clustered based on the post

template as well as posts with the same destination URLs. Two threshold values are applied to the groups for the minimum number of users in each cluster and a maximum time interval between two consecutive wall posts.

They examined their method using the Facebook community with a dataset of 3.5 million users and 187 million wall posts. The method detected 297 clusters which contain 212,863 wall posts as malicious clusters. The authors did not mention the outcome of their study in terms of performance.

Youtube (http://www.youtube.com) is another popular social video sharing website that is infiltrated by spam users to promote spam video, post spam comment on the other users' videos, etc. (Benevenuto et al. 2009) proposed a content-based method for classification of spam, promoters, and genuine profiles in the social video sharing systems i.e. YouTube. A collection of YouTube users are built and are then manually labelled according to the three mentioned classes. Authors pre-classified 31 users as promoter, 157 as spam and 641 as legitimate profiles. Three sets of attributes were extracted from video, user, and social networks. Features such as view numbers, video's duration, and favourite value of video were considered in the video attributes set. The second set which describes behaviour of users includes number of uploaded and watched videos, number of favourite videos, frequency of uploading, etc. Finally, in the social network set, four attributes (e.g. clustering coefficient, reciprocity) were used. Then, X^2 and information gaining techniques were used to assign priority to the 60 extracted features. SVM was used for classification based on two approaches - flat and hierarchical. In the flat approach, 56.69% spam, 96.13% promoter, and 94.66% genuine profiles were correctly categorised. In the hierarchical approach, first the classifier separated 92.26% promoters and 99.45% non-promoters. For non-promoter, the classifier was used again to separate spam profiles from genuine ones. The best F1-measure value achieved was 0.8.

Opinion gathering websites such as *Amazon*, *CNET*, and *Epinions* are places where users can share reviews of products, services and events. They are useful resources enabling product consumers to discover quality products and manufacturers to receiver users' feedback. However, such open platforms are not immune from malicious use, i.e spam reviews.

(Nitin & Bing 2008) proposed the first content-based spam detection system inside opinion gathering websites. Opinions are given as reviews or comments from customers on services, events, products etc. Currently, customers can post their opinions on merchant websites. Some websites such as Amazon provide an interface to accumulate users' reviews. Since there is no quality control over the context of the reviews, this results in the generation of unsolicited or spam reviews. The authors classify spam reviews into three types: untruthful reviews, brand-only reviews and non-reviews, and employ a machine learning algorithm to distinguish spam reviews from legitimate ones.

The authors started by finding duplicate and near-duplicate reviews and used them as a training set to detect the brand-only and untruthful reviews. Logistic regression (Alpaydin 2004) has been used for classification. The authors extracted 36 features that were review, reviewer and product specific (e.g. number of feedbacks, ratio of number of reviews written as first reviews, price of product etc.).

Their investigation is based on 5.8 million reviews crawled from the *Amazon* website (http://www.amazon.com), 470 of which they manually labelled for classification purposes. For the classification of brand-only and non-reviews, the average performance using AUC was 98.7%. However, the results for classification of untruthful reviews were not promising. The authors argue that detection of untruthful reviews is more difficult even for a human inspector. Hence, automatic classification of those reviews might not produce effective results.

The authors also found that:

- biased reviewers who give a negative ranking to a product are very likely to be spam,
- the products with just one review are very likely to be spam,
- top-ranked reviews are more likely to write spam reviews,
- feedback on reviews is not a good indicator of spam vs. non-spam reviews, and
- spam reviews are more likely to be given for low selling products.

The other way to detect opinion spam is to detect spam users who spread opinion spam. Lim et al. (2010) proposed a method for detecting spam profiles inside opinion gathering websites based on behavioural features (mainly content-based). The authors extracted behavioural features from the content of reviews and used them inside machine learning classifiers. The intention is to detect spam

profiles or users who distribute spam reviews in order to eliminate spam reviews inside opinion gathering websites.

The authors used Amazon API to extract 48,894 reviews and 11,038 user profiles from Amazon websites. The following features have been measured for classification purposes: (1) multiple ratings of the same product (spam users tend to post multiple high rates for the same product), (2) multiple reviews for the same product (similar to the previous feature but considers review content), high/low rating of similar products that share the same attribute within a short time span (spam users assign high/low rating for similar products – products with the same brand – within a few hours), (3) deviation from ratings by other reviewers for the same product (spam users try to demote or promote a product; hence, their ratings are different from those of others), and (4) similar to the previous feature but considers early deviation (spam users attempt to post early reviews as soon as a product is made available for review; this prejudices others' opinions and attracts more attention).

For evaluation, they randomly selected 62 user profiles and recruited three human evaluators to validate the result. The authors discovered that first and second features have better accuracy for the detection of spam profiles. By applying a regression model to the remaining 11,038 reviews, they found that over 4.65% of reviewers are spam.

The social bookmarking or social tagging websites have recently have gained attention in the Spam 2.0 filtering domain. In the social bookmarking website, users can share links to other web pages, make comments on other users' links and tag (a short descriptive annotation for the resources) the links. However, on the dark side, they can be a potential campaign for distribution of spam links.

The following studies focused on the detection of spam content inside social bookmarking websites.

(Paul et al. 2007) and (Koutrika et al. 2008) provided a comprehensive study of tag spam in social tagging systems, i.e. *del.icio.us* (*Delicious*). The authors studied the effect of different spam attacks on tagging systems and tagging search results. They investigated the effects on tagging systems of the following criteria: number of spam users, number of good users, limited number of tag per users,

usage of trusted moderators to manually block spam tags. The authors also proposed a coincide-search algorithm that considers users' reliability when generating search results. The result of the coincide-search algorithm was better than those of the state of the art search mechanisms in the tagging system.

The authors developed a simulator of a social tagging system and used a base set containing over 380,923 documents from Delicious. Their result showed that social countermeasures have a significant effect on the relevance of search results. That is, if the number of good users in the system is lower than the number of spam users, the system substantially suffers from tag spam. The moderators help to reduce the tag spam although this requires considerable amount of effort and time, and the coincide-based search showed better results compared with those of other search algorithms.

Another work on Delicious has been done in K. Liu et al. (K. Liu et al. 2009). However, the authors evaluated their method based on real data rather than simulated data. The authors made use of social countermeasures and collaborative knowledge to identify spam tags in the social bookmarking systems (K. Liu et al. 2009). The collaborative knowledge was implicitly gathered from the way users utilise the system and annotate the content. The authors measured two features: post- and user-oriented. The former is used to calculate the value of each tag compared with other tags for a specific resource. The latter is the inverse value of a user's tag on the system.

Their dataset contained 82,541 users and about 1 million tags. The authors manually classified 2,000 posts and 500 users to evaluate their proposed method. They achieved an accuracy of 93.75% for spam posts detection and 96.20% for spam user detection.

The content of the tags reveals some characteristics about the tags' nature. Therefore, it can be exploited to distinguish spam tags from the genuine ones in the social bookmarking systems. (Bogers & Bosch 2008) proposed an approach for detecting spam in the social bookmarking system (i.e. BibSonomy). Their approach is based on the assumption that the spam language model is different from the genuine ones. In the proposed method, a similarity comparison among genuine language models and spam are done at two levels: post-level and user-level. In the post-level, language models are built for each post in the system. Later, language models are matched against the language model of the new posts in order to find the similarity. Using the spam status of the best-matching posts in the training set, it is possible to predict the spam status of new posts. In the user-level, the same procedure is applied to the users' profile.

For experiment and evaluation, the authors gathered resources posted to BibSonomy in two types – bookmarks and BibTeX records. They manually labelled users as spam or non-spam (2,400 genuine users and more than 29,000 spam profiles).

The results showed that better performance is achieved by using the user-level data for building language models rather than using the post-level. In the user-level, the AUC score and F-measure of 0.9784 and 0.9767 was achieved.

Another work on BibSonomy was conducted by (Krause et al. 2008). However in this study, the authors employed a machine learning approach instead of a language model to identify spam tags. Twenty-five features were extracted to describe the systems according to four categories: (1) user's profile (e.g. the amount of digits in email or name, and length of real name, (2) user's location e.g. the amount of users from different domains, (3) user's integration e.g. number of tags in each post, (4) semantic features e.g. the number of "group=public", ratio of the spam tag to all the tags in the system.

In this paper, BibSonomy is used as the social bookmarking web site to build the data set. The data set consists of 18,681 spam users, and 1,411 non-spammers which were labelled manually.

Four machine learning mechanisms were applied including Naïve Bayes, SVM, Logistic Regression, and C4.5. The best performance achieved was for the SVM classifier and logistic regression. Since the misclassification for the false-positive rate was considerable, a cost-sensitive learning approach was used for training purposes. Similarly, the four machine learning algorithms were applied on all the features, and the highest F1-measure 0.927 was achieved for logistic regression classifier.

Apart from utilising social knowledge, a language model and statistical approach (Dellschaft & Staab 2010) investigated tagging behaviour in both Delicious and BibSonomy. The authors studied tagging behaviour in two models as well as two properties. They used the notion of *co-occurrence streams* in tagging systems to describe all tags that have occurred simultaneously. Fifteen different co-occurrence streams were extracted from two specific datasets – Delicious and BibSonomy (10 streams from the first one and 5 streams from the second one) - and they removed spam profiles from their dataset in order to minimize their impact on the results. Two properties of co-occurrence streams used in tagging models were *sub-linear vocabulary growth* and *tag frequency distributions*. The former

presents the tag's vocabulary drift, while the latter presents the distribution of the tags and the users in the system.

The authors investigated two models - *Epistemic* model and *Yule-Simon* model with *Memory* model. These models aim to simulate how micro-level behaviours of users lead to the two properties of the co-occurrence streams. In the Epistemic model, two factors impact on users' tagging behaviours: (1) the content of resources which is related to background knowledge of users about a specific resource, (2) other users' tag assignments which persuade assigners to assign specific tags. The Yule-Simon model simulates only the imitation process and ignores the background knowledge of users. Unlike the Epistemic model, the Yule-Simon model cannot explain sub-linear vocabulary growth and is also unable to consider the background knowledge of users although it can faithfully reproduce tag frequency distributions.

The results showed that in order to simulate spam tag assignments, a completely different model should be used from the ones used for regular users. It is also possible to use the same model but with a different set of parameters. In addition, the results of comparing these two models showed that for both tag frequency and vocabulary growth properties, the Epistemic model performs better than the Yule-Simon model which can be attributed to its consideration of both the imitation process and the background knowledge of users.

Apart from the above studies that have focused on specific websites, the following work targets Spam 2.0 filtering in blog comments. A content-based method to detect spam comments in blogs has been proposed in (Huang et al. 2010). Based on four features, the authors made use of a machine learning algorithm to distinguish spam comments from legitimate ones. The features are: (1) the length of comment, (2) the similarity between the comment and the blog post, (3) the divergence between the comment and the blog post, and (4) the ratio of propaganda and popular words in the comments. The authors argue that none of these individual features is a good discriminator for classification purposes and hence a combination of them needs to be considered.

SVM, Naïve Bayes and C4.5 on 2,646 manually labelled data were used for the classification task. The best result achieved was through last classifier with 84% accuracy. The authors continued their work by building a rule-based classifier based on statistical measurement of each feature. They discovered that the ratio of propaganda, the similarity and the divergence of the comments, and the length of comments were the best features. By manually obtaining 500 samples from data, the authors

evaluated their rule-based method. The results obtained were 90.4% and 84.5% average precision and recall.

Table 2.4 presents an overall overview and comparison summary of detection-based Spam 2.0 filtering methods.

Table 2.4: Comparative summary of detection-based Spam 2.0 filtering methods.

Method	Classifier	Dataset	Result	Method	Classifier	Dataset	Result
(Zinman et al. 2007) on MySpace	Linear regression, K-NN, back propagation NN, Naïve baye	800 profiles	Best 0.83 Accuracy	**(Dellschaft et al. 2010) on Delicious & BibSonomy**	-	15 co-occurrence streams	-
(Narisawa et al. 2007) on forum	-	24591 forum messages	Avg. 0.80 F1-measure	**(Lim et al. 2010) on Amazon**	-	11K profiles	-
(Nitin et al. 2008) on Amazon	Logistic regression	5.8M reviews 470 samples	Avg. 0.98 AUC	**(Irani et al. 2010) on Trending topics in Twitter**	Naïve bayes, C4.5 decision trees & Decision Stumps	1.3M tweets + web pages	Best 0.8+ F1-measure
(Uemura et al. 2008) On forums and blogs	-	24591 forum messages	Best 0.73 F1-measure	**(Benevenuto et al. 2010) on Twitter profiles**	SVM	54M twitter profiles 8207 samples	Best 0.94 Accuracy
(Bogers et al. 2008) on BibSonomy	-	31K user profiles with posts	Best 0.97 F1-measure & AUC	**(Wang 2010) on Twitter profiles**	Decision Tree, Neural Networks, SVM, Native Bayes	25K users 500 samples	Best 0.91 F1-measure
(Krause et al. 2008) on BibSonomy	Naïve Bayes, SVM, Logistic Regressuib, C4.5	20K user profiles	Best 0.93 & 0.91	**(Lee et al. 2010) on Twitter & MySpace**	10 classifiers	1015 MySpace 104 Twitter profiles	Avg. 0.98 & 0.82 F1-measure
(Koutrika et al. 2008) on Delicious	-	38K documents	-	**(Gao et al. 2010) On Facebook**	-	187M posts	-

Method	Classifier	Dataset	Result	Method	Classifier	Dataset	Result
(Benevenuto et al. 2009) on Youtube	SVM	829 profiles	Best 0.8+ F1-measure	**(Huang et al. 2010) on blog comments**	SVM, Naïve Bayes, C4.5	500 comments	Avg. 0.84 Accuracy
(Liu et al. 2009) on Delicious	-	1M tags 2000 samples	Best 0.93 and 0.92 Accuracy				

Overall, the detection-based Spam 2.0 filtering methods can be summarised as follows:

- Methods are platform-dependent. They cannot be extended to other platforms.
- They are mainly content-based and can be bypassed by spammers.
- Content-based methods are language-dependent. Hence, a solution might not be applicable to different languages.

The next section provides a discussion of state of the art prevention-based Spam 2.0 filtering methods.

2.3.4.2 Prevention-based Spam 2.0 Filtering Methods

This section discusses the current prevention-based Spam 2.0 filtering methods. In the prevention-based Spam 2.0 filtering approach, methods strive for blocking or slowing down spam distribution inside the system. By posing economical or computational challenges, Spam 2.0 filtering methods can prevent spam distribution.

One of the widely-used methods to block spam in different systems is blacklisting or whitelisting. Blacklists typically contain information about forbidden entities/sender (e.g. email address, IP address, hostname etc) that are denied to transmit information. On the other hand, whitelists contain information about trusted entities/sender (e.g. friends, known email addresses, etc).

Using blacklist to detect tweets inside is a method that has been proposed in (Grier et al. 2010). The authors analysed more than 2 million tweet URLs which can fall into three categories of spam -

scams, phishing sites and malwares. This work also includes the study of the success of spam at enticing users to click on spam URLs (click-through analysis) as well as an investigation of different spam campaigns which deceive users by taking advantage of Twitter's features.

Three blacklists were studied in this work: Google Safebrowsing which includes phishing and malware data, and *URIBL* and *Joewein* which include both spam and non-spam domains for email spam. By using twitter streaming API, the authors gathered 200 million tweets. The use of a custom web crawler to follow URLs in each tweet resulted in 25 million URLs. The web crawler was designed to find URLs' final landing web pages. As a result, 2 million URLs (8% unique URLs, 5% malware and phishing while the remaining 95%, 3% were related to scams websites) were recognised as spam.

With regard to click-through analysis, authors used *bit.ly* API (a shortening URL service) to find out how frequently URLs in tweet are clicked. Out of 245,000 spam links that belong to *bit.ly* , 97% did not obtain any hits, while the amount of hits for the other 3% were 1.6 million clicks. The authors claim that features such as the number of spam profiles that spread spam tweets, the number of followers who receive spam links, *hashtags*, and *retweet* with hashtags have increased the ratio of click-through. They found out that the click-through rate is 0.13% (number of tweets sent vs. number of clicks) which is two orders of magnitude higher than for that of spam emails. This increase is due to the existence of little information for users to decide (only 140 characters), lack of spam filter in twitter and low time complexity for broadcasting tweets in twitter.

The authors also conducted a study of the blacklists' performance based on three features - blacklist delay, evasion and domain blacklist limitation. They found that most of the spam messages are blacklisted 2-40 days after their appearance on twitter. Since 90% of clicks occur in the first two days of tweeting, it can be concluded that blacklists are too slow to be a powerful protection mechanism. 39% of malware and phishing URLs evade detection using shorteners, once they have been detected.

The use of automated tools such as spambots is quite effective for fast distribution of spam messages to many platforms. By providing a mechanism to prevent automated distribution of spam messages, Spam 2.0 filtering method can reduce this problem significantly. The majority of prevention-based methods try to identify automated attempts in order to counteract Spam 2.0.

One of the most popular prevention-based methods to stop automated requests and distribution of automate spam content is CAPTCHA (von Ahn et al. 2003). As discussed previously, CAPTCHA is a challenge and response test to make human users request and automate generated requests separately. CAPTCHA tests need to be automatically generated so that it is easy for human users to solve but hard for machines to bypass. Such tests have been widely used in email and web. Most of the current web applications include CAPTCHA as a part of their content submission and registration process.

CAPTCHA currently is the most common solution for slowing and preventing Spam 2.0 distribution.

Apart from CAPTCHA which is designed mainly to prevent automated requests, there have been some Proof-of-Works (POW) based methods (ref. Section 1.6.4.3) focused on slowing down or stopping automated requests.

Hashcash is an example of the POW-based method used for slowing down automated form submissions (Mertz 2004). The idea behind hashcash is to increase the cost of web form submission thereby increasing computational, time and economical costs of the spam distribution. With this method, the sender needs to calculate a stamp for submitted content. The calculation of the stamp is difficult and time-consuming but comparatively cheap and fast for the receiver to verify.

However, by employing a pool of computational resources, an adversary can bypass the time and cost of computation to calculate the stamp. This makes Hashcash an impractical solution for the current trend of spam distribution.

There are some technical solutions aimed at detection of spambots prior to visiting a website. However, most of these approaches are not effective enough to prevent spammers, mainly because

they are out-dated or spammers can easily bypass them. The following is a discussion of these technical solutions along with spammers' tactics to bypass the methods.

Robots.txt: Known as Robot Exclusion Protocol is a text file normally inside a root directory of websites. It contains a list of access restriction places (e.g. web pages, directories, etc) for the website that forbid web robots from accessing them (Koster 1994). According to this protocol, web robots should examine robots.txt file whenever they visit a website. However, pursuing Robot Exclusion Protocol is voluntary and many web robots do not follow this protocol (Koster 1995). As spambots are tasked to distribute promotional and junk content, it is doubtful that they would observe the robot rules stated in the robots.txt file. Therefore, this solution is ineffective in dealing with spambots.

User-agent: User-agent is a field inside a HTTP Request Header to the web server that identifies the client application. A web robot should declare its identity to the web server by specifying this field (Koster 1993). For instance, the user-agent for a Yahoo! web crawler is Slurp (Yahoo! 2009). By simply examining the user-agent field, web administrators are able to restrict access for some specific web robots. However, spambots often hide their identity or rename their user-agent to a name that is not restricted (Hayati et al. 2009). Therefore, this technique is ineffective in restricting spambots.

Head Request: The response of web server on the HTTP Head request is header information without a message body. Web robots use head request to check the validity of hyperlinks, accessibility and recent modifications on the requested web page (Fielding et al. 1999). Therefore, large numbers of head requests in the incoming traffic can indicate web robot activity. However, this heuristic is not effective at detecting spambots as they normally request the message body rather than the header.

Referrer: Referrer is a field inside the HTTP request that contains a link to the web page a client has followed to reach the current requested web page. For instance, if a user reaches *www.example/page2.html* from *www.example.com* page, the referrer field is *www.example.com*. Ethical web robots normally do not assign any value to the referrer field so they can be differentiated from human users. However, spambots often assign a value to the referrer field to mimic a human user and can also provide fake links to conceal their navigation (Park et al. 2006).

Flood Control: Flood control is a technique to limit the number of requests a client can send within a specified time interval. The idea is to restrict the number of automated submission requests to the server (Ogbuji 2008). For example, once a user creates a new thread in a forum, s/he may have to wait

ten minutes before creating another thread. This technique may slow down automatic submissions by spambots but it also causes inconvenience for genuine users. Additionally, such a technique cannot stop spambots but only delay the submission process. Spambots often create multiple user accounts and can therefore easily bypass this technique (Hayati et al. 2009).

Nonce: Nonce is a technique to stop the submission of automated forms. Nonce is a random generated set of characters that are placed on a web page with a form (Ogbuji 2008). When the user submits the form, the nonce is sent to the server. If the nonce does not exist in the form submission, it reveals that the form was not loaded by the client and indicates the possibility of an automated attack by a spambot. This technique only ensures that submitted content originates from a web form loaded by the client, but cannot stop spambots that submit content through the website forms.

Form variation: This technique involves varying objects inside a form upon web page request (Ogbuji 2008). For example, the name of input fields within a form changes for each user web page request. The idea of this technique is similar to nonce, in that it wants to ensure that a server form generated is used during submission. However, like nonce, it cannot block spambots that use the website forms.

Overall, prevention-based methods are about to expire as computers become more powerful. The studies show that state of the art prevention methods such as CAPTCHA are weak against current spam tactics.

Apart from detection and prevention based Spam 2.0 filtering methods, there are some studies in the literature which aim to identify spam behaviour inside the system. The next section presents a survey of these methods known as early-detection based Spam 2.0 filtering methods.

2.3.4.3 Early-Detection based Spam 2.0 Filtering Methods

The early-detection based Spam 2.0 filtering methods try to discover spam behaviour inside the system prior to or just after spam content distribution. These methods investigate the behaviour of spam origin, such as spambots.

(Tan & Vipin Kumar 2002) proposed a rule-based framework to discover search engine crawlers and camouflaged web robots. They utilised navigation patterns such as session length, set of visited web pages, and requested method type (e.g. GET, POST, etc) for classification. Their main assumption

was that navigational patterns of web robots are different from those of human users. For instance, a web crawler intends to maximise coverage of a website; hence, it typically crawls in a breadth-first manner. A link checker robot (to validate whether a link in a web page is still valid or not) normally sends HEAD requests to obtain header information of a web page. This characteristic is quite different from that of human users who typically request the entire web page.

The authors utilised the following properties to distinguish web robots from human users:

- The total number of requested web pages,
- the number of image requests,
- the total time of each session, and
- the number of different file format requests.

Their proposed method initially started by labelling each session in web server log files based on some already known heuristics such as user-agent field, IP address, etc. The task resulted in 4 types of sessions: known web robots, known human users, unknown robots, and unknown human users. The tendency to assign each type is towards a non-robot. The authors utilised a C4.5 decision tree classifier and common precision, recall and F1-measure to evaluate their method.

Their dataset for this study contained 180,602 sessions with 1,639,119 web log entries. For training purposes, the authors chose web robot and non-robot sessions equally. The result of their work showed that after 4 requests, their method was capable of differentiating human users from web robots with 90% accuracy.

The adoption of non-robot type classification to simplify the model can result in misclassification. The authors did not investigate the existence of web spambots that mimic human users' behaviour.

Another early-detection based method that focuses on spambot navigational patterns was implemented by (Park et al. 2006). The authors of this work proposed a malicious robot detection method by employing both rule-based and machine learning algorithms. In the former case, the main assumption is that typically robots do not show any mouse-movement activity while interacting with the websites and they do not request any specific web objects (e.g. style sheet, Flash files etc). With the latter method, authors used machine algorithms with specific training based on 12 features (e.g.

the number of HEAD requests in HTTP, the number of HTML requests, the number of CGI requests etc.).

The authors experimented on 929,922 sessions. They discovered that 24.2% of the total sessions belonged to humans which either have mouse movement activity or requested the style sheet file. However, the percentage of sessions with mouse movement activity was 22.3% which exhibits 1.9% false positives in the result. The authors explained that this might be related to some Internet configuration settings inside users' computers that disallow the execution of specific scripts. The authors labelled 167,246 sessions consisting of 42,975 human and 124,271 robot sessions to use with a machine learning algorithm. The result showed 95% accuracy in classifications after 160 requests.

The proposed method needs a significant number of requests in order to generate reasonable classification accuracy. Hence, it might not be applicable for real-world practices.

Table 2.5 presents a comparative summary early-detection based Spam 2.0 filtering methods.

Table 2.5: Comparative summary of early-detection based Spam 2.0 filtering methods.

Method	Platform	Classifier	Dataset	Accuracy	Extendable to other platforms?
(Tan et al. 2002)	Websites	C4.5	1M web server logs	Best 0.9 Accuracy	Yes
(Park et al. 2006)	Websites	8 classifiers	16K session in the web server logs	Avg. 0.95 Accuracy	Yes

Overall, early detection-based methods did not fully consider spam user (e.g. spambot) identifications and they have been designed to identify web crawlers in general. Additionally, they are platform-dependent and cannot be extended to different platforms.

Given the Spam 2.0 filtering methods discussed in previous sections, the next section provides a comprehensive evaluation of the current state of the art Spam 2.0 filtering methods.

2.3.5 Evaluation

It was clear from the literature on spam filtering that, once a web platform increases in popularity, it attracts malicious users and spam content. The area of Spam 2.0 filtering is very young but recently it has received much attention from the research community. The comprehensive survey of the state of the art Spam 2.0 filtering methods has been discussed in the previous sections. This section gives a critical evaluation of such methods to point out the gaps and problems in the Spam 2.0 literature. Table 2.6 presents an evaluation of current Spam 2.0 filtering methods.

The main drawback of the majority of Spam 2.0 filtering methods is that they depend on one specific platform. For example, many Spam 2.0 filtering methods have been proposed for Twitter. However, they are not extendable to the other platforms such as online discussion boards, blogs, wikis etc. This is due to the nature of the features which are heavily platform-dependent (e.g. the ratio of hashtags and tweet content, the amount of tweets per users etc.).

The other problem associated with the majority of Spam 2.0 filtering methods is that they are mainly content-based. Previously, it has been shown that content-based methods can be easily overcome by spammers. Additionally, the content-based methods are language-dependent and might not be extendable to other languages.

The majority of detection-based methods are all afterthoughts. Once spam content has already infiltrated the platform, such methods are applicable to discover spam patterns. The detection-based methods require more resources with regards to CPU and time.

Prevention-based Spam 2.0 filtering methods, such as CAPTCHA, inconvenience end users and waste their time, cause distraction, are unpleasant and at times intimidating. A number of recent works have reported approaches to defeat CAPTCHAs automatically by using computer programs (Abram et al. 2008). CAPTCHA's drawbacks include the following:

- Decreases user convenience and increases complexity of human computer interaction.
- As programs become better at deciphering CAPTCHA, the image may become increasingly difficult for humans to decipher.
- As computers become more powerful, they will be able to decipher CAPTCHA better than humans can.

Therefore, CAPTCHA is a short-term solution that may not prove effective in the future. Spambots armed with anti-CAPTCHA tools are able to bypass this restriction and spread spam content easily while the futile user inconvenience remains.

Finally, early-detection based Spam 2.0 filtering methods have not comprehensively considered current trends in Spam 2.0 and spambots. To date, spambots can mimic human user behaviour; however, the main intention of the early-detection based methods was to identify web robots in general, not spambots specifically.

Table 2.6: Evaluation of state of the art Spam 2.0 filtering methods*.

Method	Comment	Forum (personal message)	Wiki	Social Net (profile, targeted)	Content-based	Extendable	On-he-fly detection
(Tan et al. 2002)	?	?	?	?	-	×	×
(von Ahn et al. 2003)	?	?	?	?	-	×	×
(Mertz 2004)	?	?	?	?	×	×	-
(Park et al. 2006)	?	?	?	?	-	×	×
(Zinman et al. 2007)	-	-	-	×	×	-	-
(Narisawa et al. 2007)	-	×	-	-	×	?	-
(Nitin et al. 2008)	-	-	-	×	×	-	-
(Uemura et al. 2008)	×	×	-	-	×	?	-
(Bogers et al. 2008)	-	-	-	×	×	-	-
(Krause et al. 2008)	-	-	-	×	×	-	-
(Koutrika et al. 2008)	-	-	-	×	×	-	-

Method	Comment	Forum (personal message)	Wiki	Social Net (profile, targeted)	Content-based	Extendable	On-he-fly detection
(Benevenuto et al. 2009)	-	-	-	×	×	-	-
(Liu et al. 2009)	-	-	-	×	×	-	-
(Dellschaft et al. 2010)	-	-	-	×	×	-	-
(Lim et al. 2010)	×	-	-	-	×	-	-
(Irani et al. 2010)	-	-	-	×	×	-	-
(Benevenuto et al. 2010)	-	-	-	×	×	-	-
(Wang 2010)	-	-	-	×	×	-	-
(Lee et al. 2010)	-	-	-	×	×	-	-
(Gao et al. 2010)	-	-	-	×	×	-	-
(Huang et al. 2010)	×	-	-	-	×	-	-
(Grier et al. 2010)	?	?	?	×	-	?	?

* *Cross* sign: the method can countermeasure the web spam tactics (the authors have specifically mentioned this in their study). *Question mark*: the method might counteract the web spam tactics (The authors have not mentioned this in their study.)

Figure 2.7 presents the coverage of Spam 2.0 filtering methods based on different evaluation criteria in this survey. It is clear from the figure that most of the efforts are on Spam 2.0 detection in social networking systems and the majority of the methods are content-based.

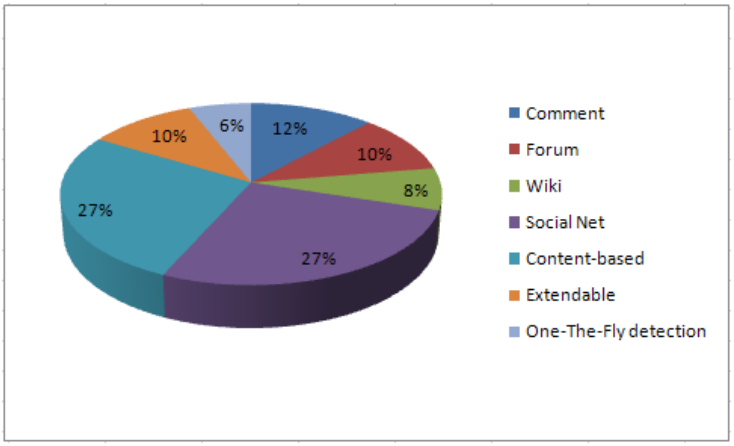


Figure 2.7: The coverage of the Spam 2.0 filtering method based on different evaluation criteria

Figure 2.8 illustrates the number of Spam 2.0 filtering methods based on each technical approach in this survey. It is clear from the figure that the majority of the studies use detection-based approaches.

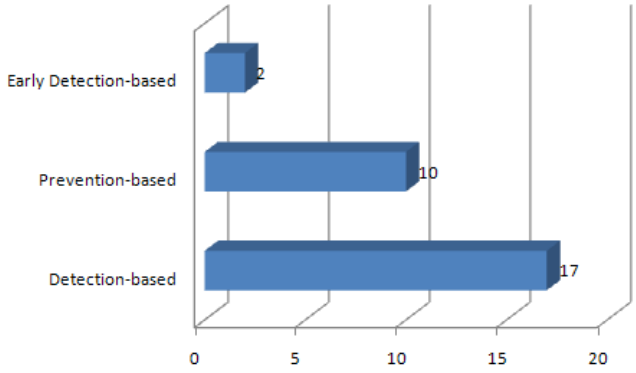


Figure 2.8: The number of Spam 2.0 filtering methods based on different technical approaches.

The overall issues that exist in the current Spam 2.0 filtering methods are as follows:

1. Methods are platform-dependent. They cannot be extended to other platforms.
2. Detection-based methods are mainly content-based and can be bypassed by spammers.
3. Content-based methods are language-dependent. Hence, a solution might not be applicable to different languages.
4. Prevention-based methods are about to expire as computers become more powerful. The studies show that state of the art prevention methods such as CAPTCHA are weak against current spam tactics.

5. Early detection-based methods did not fully consider spambot identifications; rather, they have been designed to identify web crawlers in general.

Spam 2.0 filtering is the main focus of this research which proposes a new approach for the elimination of Spam 2.0 which is discussed in the following chapters. The next section concludes this chapter.

2.4 Conclusion

In this chapter an overview of the relevant literature and the current efforts in the spam filtering field has been presented, together with an investigation and evaluation of the most successful anti-spam methods. A survey was conducted of spam filtering efforts on the web and Web 2.0 platforms including detection-based (i.e. content-based, linked based and rule-based), prevention-based and early-detection based approaches. Although this research presents a novel Spam 2.0 concept that has not been investigated previously, attempts that have been utilised in other related types of spam (i.e. web spam) are also investigated for the purposes of contrast and to promote this approach.

Through study and evaluation of existing spam filtering approaches in the web spam area, it has become clear that state of the art methods focus on providing a more robust web page ranking algorithm or identification of web spam tactics in the search results. Technically, the methods are mainly based on link- and detection-based approaches.

This dissertation focuses on the comprehensive study of Spam 2.0 in terms of several web spam concerns as well as new problems. A comprehensive survey of the state of the art web spam and Spam 2.0 filtering methods that have been examined in this chapter is one of the main contributions of this study to highlight the unresolved issues and at the same time effectively capture the knowledge in the domain of spam filtering.

Chapter 3

Problem Definition

This chapter covers

- A formal definition for spam filtering categories.
- Problems associated with Spam 2.0 filtering approaches.
- The research issues that need to be addressed.
- The research methodology that will be adopted in this research to systematically address the identified research issues.

3.1 Introduction

Spam filtering has been a challenging problem on the web. Search indexing algorithms need to prune spam web pages from their indexes, website operators needs to eliminate spam content (e.g. comment spam) from their websites, and web developers need to be aware of the latest spamming tactics in order to ensure their software is robust against spam attacks.

A comprehensive survey and detailed evaluation of the state-of-the-art spam filtering methods in web spam and Spam 2.0 were discussed in Chapter 2. Although there has been considerable research in the spam filtering area, no single approach has addressed the problems produced by Spam 2.0.

This chapter outlines the main problems and challenges in Spam 2.0 detection, prevention and early detection approaches. It provides the problem definition for Spam 2.0 filtering categories. It then follows with a formal description of the research issues that will be addressed in this dissertation. The chapter concludes with a brief discussion of research methodologies and a summary of the problem definition.

3.2 Problem Definition

The problems and challenges that are associated with Spam 2.0 filtering can be categorised according to three categories as follows:

- Problems with detection-based Spam 2.0 filtering methods.
- Problems with prevention-based Spam 2.0 filtering methods.
- Problems with early-detection based Spam 2.0 filtering methods

For each category, a formal definition is given and the technical concerns are discussed. The formal definition is used to define the technical problem. The problems associated with the technical concerns will form the research issues for the development of the new solution.

3.2.1 Problems with detection-based Spam 2.0 filtering methods

3.2.1.1 Formal definition

The detection-based approach strives to discover spam patterns inside the content of user-submitted data. In Spam 2.0, the content can be in the form of a comment, a forum post, a wiki page, a social profile etc. A *content* is represented as a set of *characteristic values* or *features* (*F*) E.q. 3.1.

$$F = \{f_1, f_2, \ldots, f_{|F|}\} \quad 3.1$$

where f_i is a value of the features associated with *content*. For example, in online discussion boards, f_i can represent the existence of the given terms inside the forum post. f_i is 1 if the i^{th} term exists in the forum post, otherwise $f_i = 0$. Therefore, *F* is a Boolean *content* vector. The process of representing *content* in the form of *F* is also known as *feature extraction*.

A binary decision function φ is defined over *F* for classification as follows.

$$\varphi(f_i, c_j) : F \times C \rightarrow \{0, 1\} \quad 3.2$$

C is a set of *content* class where

$$C = \{c_l, c_s\} \quad 3.3$$

c_l refers to legitimate *content* and c_s refers to spam.

The overall process in the detection-based Spam 2.0 filtering methods is illustrated in Figure 3.1.

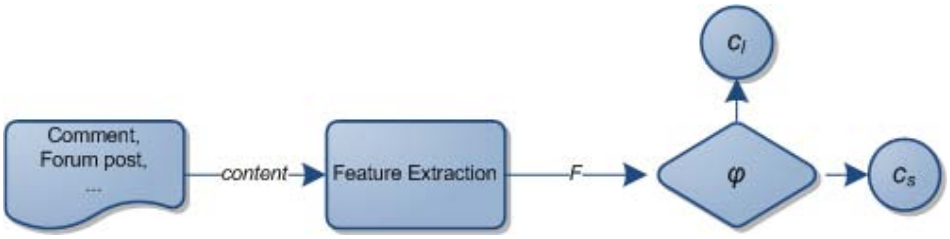


Figure 3.1: Overall process in the detection-based Spam 2.0 filtering methods.

3.2.1.2 Technical concerns

The detection-based Spam 2.0 filtering methods seek spam patterns with used-submitted content in Web 2.0 platforms. The main technical drawbacks with these approaches are:

1. **Passive solutions**. Detection-based Spam 2.0 filtering methods are afterthought approaches. That is, once the spam content has already infiltrated Web 2.0 platforms, a detection-based method can be implemented to distinguish spam from genuine content i.e. c_l from c_s.

2. **Consume network and storage resources.** As spam uses the network bandwidth to distribute content on Web 2.0 platforms, this may waste the limited bandwidth available for websites. Additionally, storage of spam content can waste the storage space and incurs monetary costs.

3. **Inability to automatically adapt to new spam patterns**. The majority of detection-based approaches are supervised (ref. Section 2.3.5). There is a need to constantly update spam pattern datasets to address new spam patterns. Maintenance and storage of such datasets is expensive in terms of time and money.

4. **Inability to adapt with different platforms and languages.** Most of the methods are designed to identify spam inside specific Web 2.0 platforms (ref. Section 2.3.5). Hence, the features (*F*) are platform-dependent and currently there is no single solution that can be applied to multiple Web 2.0 platforms. Moreover, the majority of detection-based methods are content-dependent. Hence, they might be useful for a particular language (e.g. English) but cannot be extended to identify spam in other languages.

The detection-based Spam 2.0 filtering methods are still burdened by the above issues and the weak classification results of such methods shows that no method to date has addressed all of the problems.

3.2.2 Problem of prevention-based Spam 2.0 filtering methods

3.2.2.1 Formal Definition

The prevention-based Spam 2.0 filtering methods impede various functionalities of the system in order to keep malicious users out. The majority of the-state-of-art prevention-based methods produce time and computational challenges. Some may also produce economic challenges through online payments for accomplishing various tasks in the system.

A user (client) who wants to perform an action in the system (e.g. register an account) needs to overcome a challenge that is issued by the server. The server can quickly and easily verify the client's solution to the challenge and may grant permission to the client to perform the action.

The server uses function *CHAL()* to issue the challenge (*C*). The client solution for *C* is *S* using a function *SOL()*. The server validates *S* through function *VER()* and if the output has required value (*V*), the server allows the client to complete the action. E.q. 3.4 illustrates the formal procedure of the prevention-based methods.

$$\begin{cases} C & \leftarrow CHAL() \\ S & \leftarrow SOL(C) \\ V & \leftarrow VER(S) \end{cases} \quad 3.4$$

In the prevention-based Spam 2.0 filtering methods, *C* is either a human solvable challenge that can be met only by humans (e.g. CAPTCHA), or a computational challenge which requires computational resources in order to meet the challenge (e.g. POW).

The process followed by the prevention-based Spam 2.0 filtering methods is presented in Figure 3.2.

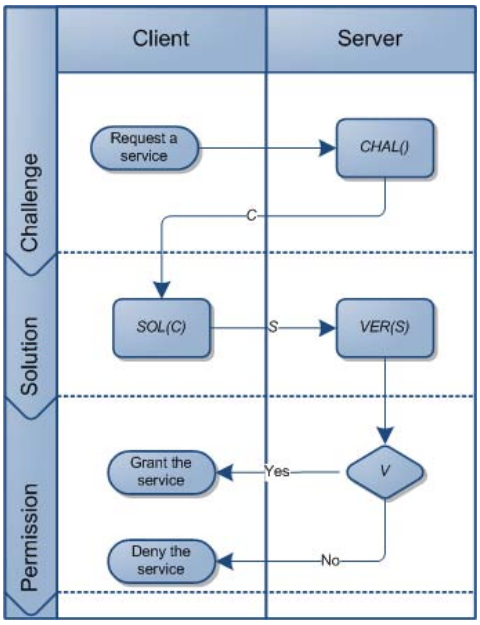


Figure 3.2: The process followed by prevention-based Spam 2.0 filtering methods.

3.2.2.2 Technical concerns

The main technical problems associated with prevention-based Spam 2.0 filtering method are:

1. **Weak/complex challenges result in ineffective Spam 2.0 filtering**. The goal of CAPTCHA is to allow humans to pass while preventing automated systems from gaining access. It is important to consider the types of challenges posed by CAPTCHA. Complex challenges inconvenience the users as they waste their time, cause distraction, are unpleasant and, at times, intimidating. Visual and textual CAPTCHA challenges pose major problems for human users with specific disabilities such as vision impairment (May 2005). Some challenges might also restrict them from accessing the system.

2. **Computer programs are becoming better at processing CAPTCHA**. As computers become more powerful, they will be able to decipher CAPTCHA better than humans can. A number of recent works have reported approaches to defeat CAPTCHAs automatically by using computer programs (Hindle et al 2008)(Chellapilla and Simard 2004). The latest version of CAPTCHA known as *reCAPTCHA* (von Ahn et al. 2008) has also been by-passed by computer programs (Houck 2010).

3. **Computer programs are becoming more proficient at overcoming POW challenges.** By accessing a network of compromised computers (i.e. BotNet), an adversary can have access to

a significant pool of computational resources to compute various POW challenges. Hence, the current computationally expensive challenges might not be effective for spam filtering in the near future (ref. Section 2.3.4.2).

4. **Current prevention-based methods are vulnerable to relay attacks.** In a relay attack, adversaries use humans to overcome a challenge. For example, a spammer can hire someone to defeat CAPTCHA and bypass the restrictions (Workathome 2009).

5. **Blacklists are too slow to protect Web 2.0 platforms.** Studies (Grier et al. 2010) show that most of the spam hosts are blacklisted 2-40 days after their appearance. Since 90% of clicks occur in the first two days of spam distribution, it can be concluded that blacklists are too slow to offer effective protection against spam.

6. **Blacklists are not effective at Spam 2.0 filtering**. Spam URLs evade detection by using URL shorteners. Additionally, the maintenance of blacklists is time-consuming and labour intensive. Hence, blacklists are not effective enough to protect Web 2.0 platforms against Spam 2.0.

The state of the art prevention-based Spam 2.0 filtering methods are still subject to the abovementioned technical concerns and to date no study has been able to resolve all of these issues.

3.2.3 Problem of early-detection based Spam 2.0 filtering methods.

3.2.3.1 Formal Definition

Similar to the detection-based Spam 2.0 filtering methods, early-detection based Spam 2.0 filtering identifies spam generation behaviour. Methods in this category are not detection-based, i.e. searching for spam patterns inside the content; nor are they prevention-based that generate a challenge for clients.

Typically, by investigating the behavioural aspects of spam content generation, such early-detection based spam filtering methods are capable of detecting spam. This *behaviour* is represented as a set of features (*F*) E.q. 3.5.

$$F = \{f_1, f_2, \ldots, f_{|F|}\} \quad 3.5$$

where f_i is a value of the features associated with *behaviour*. For example, in an online discussion board, each f_i can represent the total amount of time a user has spent to generate a reply to a thread or the existence of mouse activity while the user was navigating through the web platform.

Similar to the detection-based methods, a binary decision function φ is defined over F.

$$\varphi(f_i, c_j) : F \times C \rightarrow \{0, 1\} \quad 3.6$$

C is a set of the *behaviour* class where

$$C = \{c_l, c_s\} \quad 3.7$$

c_l refers to legitimate *behaviour* and c_s refers to spam *behaviour*.

The process used by early-detection-based Spam 2.0 filtering methods is illustrated in Figure 3.3.

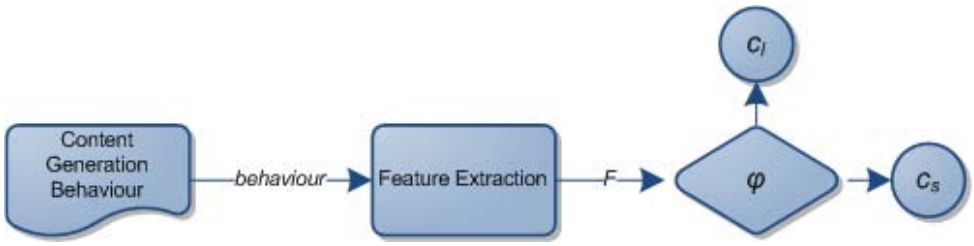


Figure 3.3: The process used by early-detection-based Spam 2.0 filtering methods.

3.2.3.2 Technical concerns

Technical concerns associated with early-detection based Spam 2.0 filtering methods are as follow.

1. **Inability to identify Web 2.0 spam users**. Existing early-detection methods are not capable of identifying web spam users (e.g. spambots) and there is no study in the literature that addresses the detection and prevention of web spambots.

2. **Machine learning based methods are not effective for real-time detection**. Machine learning based solutions require a considerable amount of time and computational resources (Sculley and Gabriel 2007). Therefore, they might not be suitable for spontaneous spam user identification on the web.

3. **The nature of Spam 2.0 is not well understood.** As discussed in Chapter 1 and 2, no study as yet has focused on characterising the new generation of spam in Web 2.0 platforms. Hence, Spam 2.0 characteristics are not well defined for effective filtering approaches.

The above list presents issues and weaknesses of the current early-detection based Spam 2.0 filtering methods. To date, no study has been conducted to resolve all of these issues.

3.3 Research Issues

Given the problem definition presented in above sections and the evaluation of the-state-of-art Spam 2.0 filtering methods discussed in Chapter 2, this section discusses the research issues that are addressed in this study. From Chapter 4 onwards, the solutions that attempt to address these research issues and research objectives (Chapter 1) will be pursued.

3.3.1 Research Issue 1: The nature of Spam 2.0 is not well understood

Based on the survey of current Spam 2.0 filtering methods that have been discussed in Chapter 2 and problem definitions for Spam 2.0 filtering methods, it is clear that Spam 2.0 is not well understood. The literature contains no general definitions of such spam. The majority of the methods are directed at applications of existing techniques as used in other spam filtering domains (e.g. email/webspam). They are not effective enough to identify and combat Spam 2.0.

In order to address this problem, a number of research questions need to be investigated:

1. What are the differences between Spam 2.0 and other types of spam? What are the aims of Spam 2.0 distribution? How is Spam 2.0 created?

2. What is the nature of Spam 2.0 generation? What tools do spammers use to generate and distribute Spam 2.0 content? Is Spam 2.0 distributed automatically or manually?

3. What tactics do spammers employ to find and choose their targets (Web 2.0 platform)?

4. How effective are current Spam 2.0 filtering methods in identifying and blocking spammers? How do spammers bypass current detection, prevention and early-detection based Spam 2.0 filtering methods?

5. What are the main topics of Spam 2.0-related content used by spam users?

6. What are the main characteristics of Spam 2.0 distribution? How are these different from real human user behaviour?

7. What possible strategies or approaches can be used to block Spam 2.0?

By creating a platform to monitor Spam 2.0 generation and archive Spam 2.0 content, this research addresses all the above questions. This platform is discussed in Chapters 5 and 6.

3.3.2 Research Issue 2: Existing Spam 2.0 filtering solutions are afterthoughts

As discussed in Chapter 2, current Spam 2.0 filtering methods are mainly detection-based. The majority of detection-based approaches are not capable of eliminating spam content before its submission. Therefore, the main technical problem associated with current methods is that they are passive and afterthought solutions.

In order to address this issue, the following questions need to be addressed.

1. How can a spam propagator be identified before content submission?

2. How can an appropriate approach be selected to detect Spam 2.0 in a more proactive way?

In this research, all the above questions have been addressed in the design of the proposed Spam 2.0 filtering method. The conceptual framework for this design is presented in Chapter 7.

3.3.3 Research Issue 3: Platform and language-dependent approaches

Existing Spam 2.0 filtering approaches are either platform-dependent or language-dependent. They might be effective only in particular platforms or languages. Therefore, the main technical problem here is that current methods are platform and language dependent.

In order to address this issue following questions need to be addressed.

1. What approaches are required to make spam filters language-independent?

2. Which features are language-dependent?

3. How can a spam filter be extended to multiple platforms?

4. What discriminative features need to be investigated that are common among the majority of Web 2.0 platforms for Spam 2.0 filtering?

In this research, all the above questions have been addressed in the design of the proposed Spam 2.0 filtering method. The conceptual framework for this design is presented in Chapter 7.

3.3.4 Research Issue 4: Current methods waste network and storage resources

As mentioned in Research Issue 2, existing methods are not capable of eliminating spam content before its submission. Therefore, spam content is transmitted and hosted on a web platform. This results in network and storage wastage. To address this wastage issue, the following questions need to be considered.

1. How can spam content be identified before its submission on a web platform?

2. What other features can assist the minimisation of resource wastage?

In this research, all the above questions have been addressed in design of the proposed Spam 2.0 filtering method. The conceptual framework for this design is presented in Chapter 7.

3.3.5 Research Issue 5: Prevention-based methods pose inconvenience to human users

A survey of prevention-based Spam 2.0 filtering methods has been presented in Chapter 2. Based on the problems definition associated with such methods, the following main technical problems have been addressed in this research.

- Weak/complex challenges result in ineffective Spam 2.0 filtering.

- Computer programs are becoming better at processing CAPTCHA.

- Computer programs are becoming proficient at overcoming POW challenges.

- Current prevention-based methods are susceptible to relay attacks.

The above technical problems are resulted in a number of research questions. In order to overcome the problems, the following research questions need to be considered.

1. What factors are important in slowing/blocking spam distribution? Do these factors limit real user experience as well? How can spam be kept out of the system without limiting the genuine user experience?

2. How can spammers be explicitly blocked out of the system while letting genuine users enter to the system?

3. In order to avoid the limitations of prevention-based methods (e.g. CAPTCHA) for disabled people, what approach needs to be investigated?

Above research questions need to be addressed in order to make an effective Spam 2.0 filter. The conceptual framework for this design is presented in Chapters 7 and 8.

3.3.6 Research Issue 6: Inability to identify Web 2.0 spam users

Existing early-detection based methods that have been discussed in Chapter 2 aim to identify automated requests. For example, such methods have been designed to detect web robots (e.g. link checkers) activities on the web platform. Although malicious web robots are employed by spammers to distribute Spam 2.0 content, existing early-detection based methods are not specific and effective enough to identify web spam users.

In order to address this issue, the following research questions need to be considered.

1. What are spam users? What do spammers use as tools (e.g. spambots)? How do spam users navigate the Web 2.0 platforms?

2. What do spam users aim to achieve?

3. What are the key features of spam users' behaviour? What are the differences between spam and genuine users’ behaviour?

In this research, all the above questions have been addressed in the design of the proposed Spam 2.0 filtering method. The conceptual framework for this design is presented in Chapters 7 and 8.

3.3.7 Research Issue 7: Machine learning based methods are not effective for real time detection

Machine learning algorithms need a considerable amount of time and computational resources (Sculley and Gabriel 2007). Therefore, they are not effective for spontaneous classification tasks in the spam filtering process due to the rapid rate of data generation on Web 2.0 platforms and the need to identify spam content quickly. In order to address this issue, the following research questions need to be considered.

1. How can the generated data be modelled so that it is suitable for online scenarios where the incidence of data generation is high?

2. What classifier can perform well in on-the-fly situations?

To address the problem of existing early-detection methods for the identification of spambots, the above research questions need to be addressed. The conceptual framework for this design is presented in Chapters 7 and 8.

The next section provides the research methodology that would be adopted for this research.

3.4 Research Methodology

A science and engineering-based research approach will be adopted for this research. Science and engineering research leads to the development of new techniques, architecture, methodologies, devices or a set of concepts, which can be combined to form a new theoretical framework. This research approach commonly identifies problems and proposes solutions to these problems.

The science and engineering approach that has been utilised in this research consists of: problem definition, conceptual solution, implementation, experimentation, testing and validation of prototypes against existing solutions. It consists of three main stages:

- Problem definition

- Conceptual solution

- Implementation, testing and evaluation

3.4.1 Problem definition

In the problem definition stage, the aim is to highlight the significance of the research questions. This problem definition stage has been covered in this chapter. Problems for spam filtering method have been grouped into three different Spam 2.0 filtering categories: detection-based, prevention-based, and early-detection based. For each category, the discussion is carried out from three perspectives: a formal definition of the category, the socio-economic, and the technical concerns. The problems associated with the technical concerns led to the research issues for the new solution development.

3.4.2 Conceptual solution

The conceptual solution focuses on formulating a new method and approach through the design and building of tools, an environment or system through implementation. In this stage, a conceptual framework is designed for the proposed solution. A conceptual framework is an abstract model of the practical solution. It provides a road map for building the actual solution for the system and can be regarded as a blueprint for the whole system.

3.4.3 Implementation, test and evaluation

In this stage, testing and validation are carried out through experimentation with real-world examples and field testing. The process of testing and validating a working system provides unique insights into the benefits of the proposed concepts, frameworks and alternatives.

By building a prototype system, implementing, testing and evaluating, a better insight into the feasibility and functionality of the conceptual framework as well as the whole solution, is provided.

In this dissertation, Chapter 4 provides a conceptual framework for the proposed solution, and Chapter 5 to Chapter 8 carry out implementation and experimentation of the conceptual framework along with testing of the proposed solutions against real-world practices and comparisons of the proposed methods against existing practical solutions

3.5 Conclusion

This chapter provides a problem definition for existing Spam 2.0 filtering methods and approaches. Based on the socio-economic and technical problems of existing solutions, four research issues have been defined. For each research issue, a number of research questions have been proposed. These

research questions need to be addressed in the development of any new Spam 2.0 filtering method. To address each research issue, three research aims have been proposed. Furthermore, the research methodology for this research has been discussed.

In the next chapter, an overview of the proposed solution along with its conceptual framework will be provided. The conceptual framework is designed to address all the issues that have been discussed in this chapter.

Chapter 4

Overview of Solution and Conceptual Process

This chapter presents

- an overview of the proposed solutions
- a conceptual framework of the proposed solutions
- a conceptual process adopted in the development of the proposed solutions

4.1 Introduction

As outlined in Chapter 2, a number of works have addressed the issue of Spam 2.0 filtering. Existing Spam 2.0 filtering methods strive to identify spam content inside Web 2.0 platforms. However, this area of research is quite young and it is evident that to date no research has resolved the Spam 2.0 problem entirely. Chapter 3 presented seven major research issues pertaining to the existing solutions that are addressed in this research.

This chapter provides an overview of the solutions to each of the research issues discussed in Section 3.3.

4.2 Overview of the Solution

In this section, three solutions are briefly introduced to address the research issues discussed in Section 3.3. These three solutions are:

1. HoneySpam 2.0.
2. Early detection-based Spam 2.0 filtering method (EDSF)
3. On-the-Fly Spam 2.0 filtering method (OFSF)

Each of these proposed methods is discussed below.

A framework named *HoneySpam 2.0* is proposed in Chapter 5 to monitor and store spam users' activity on Web 2.0 platforms. This solution is proposed to address the following:

- Research Issue 1: The nature of Spam 2.0 is not well understood

HoneySpam 2.0 is capable of tracking and archiving current Spam 2.0 web navigational behaviour and its associated submitted content on active Web 2.0 platforms. The outcome of this investigation provides a better insight into the nature of Spam 2.0 for future Spam 2.0 filtering approaches.

Based on initial tracking results from HoneySpam 2.0, Chapter 6 provides a framework to investigate, formulate and cluster a spam user behaviour model. This is achieved by discovering uncovered spammers' behaviour on Web 2.0 platforms, understanding spammers' targets, formulating spammers' behaviour in order to develop discriminative feature sets, and cluster spam users' activities based on similarities.

The Spam 2.0 filtering scheme named *Early-Detection based Spam 2.0 Filtering* (EDSF) is proposed in Chapter 7 to identify and eliminate spam users from Web 2.0 platforms. This solution is proposed to address the following issues:

- Research Issue 2: Existing Spam 2.0 filtering solutions are afterthoughts
- Research Issue 3: Platform and language-dependent approaches
- Research Issue 4: Current methods waste network and storage resources
- Research Issue 5: Prevention-based methods pose inconvenience for human users
- Research Issue 6: Inability to identify Web 2.0 spam users

EDSF implicitly investigates Spam 2.0 content generation and spam users on any Web 2.0 platform; hence, it does not pose inconvenience for human users. The proposed solution minimises resource wastage by preventing submission of spam content and it is language and platform independent.

The EDSF scheme is then extended in Chapter 8 where an On-the-Fly Spam 2.0 Filtering (OFSF) scheme is presented. OFSF improves the practicality, effectiveness and robustness of the EDSF. This solution is proposed to address the following issue:

- Research Issue 7: Machine learning-based methods are not effective for real-time detection

The idea is to propose a Spam 2.0 filtering method that can spontaneously detect spam users. ESDF uses a machine learning classifier which is not effective for spontaneous detection of spam behaviour. The OFSF is capable of on-the-fly detection of spam behaviours. The idea here is to monitor users' behaviour as they use the system by using a novel rule-based classifier to store, retrieve, and classify spam users from genuine human ones.

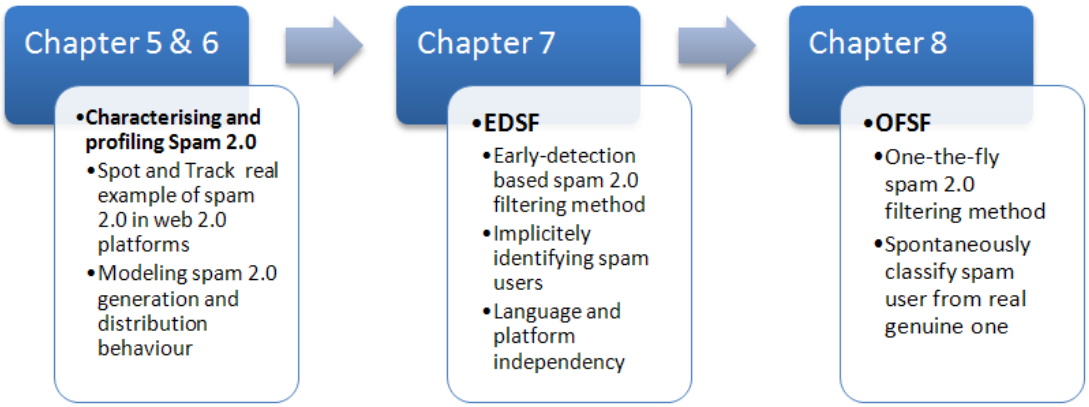


Figure 4.1: Overview of the conceptual solution.

Figure 4.1 shows how the dissertation is organised from this chapter onwards. The conceptual solution is proposed to address the four main problems concerning Spam 2.0 filtering, which were outlined in Section 3.2. The figure also shows how each of these problems is addressed in the respective chapters. In the next section, the details of each proposed solution are discussed.

4.3 Solution Description

This section provides an overview of the solutions which are developed to address the issues highlighted earlier in this chapter. The three solutions presented in this dissertation are:

1. Characterising Spam 2.0 and Spam 2.0 distribution model.

2. Early-Detection based Spam 2.0 Filtering (EDSF) method to implicitly detect spam users, be language and platform independent and minimise user inconvenience.

3. One-the-Fly Spam 2.0 Filtering (OFSF) method to spontaneously identify and prevent spam users.

Each of the above solutions is explained in the following sections.

4.3.1 Characterising Spam 2.0 and Spam 2.0 distribution model

HoneySpam 2.0 is a monitoring tool to track, log and archive spam user activities on Web 2.0 platforms. It can be embedded inside any web application to monitor Spam 2.0 activities.

HoneySpam 2.0 is capable of storing and logging all information related to spam and non-spam users on Web 2.0 platforms such as web usage data, user-profile related data, submitted content, etc. Each part of this information provides a better insight into the way Spam 2.0 works. Web usage data can be employed to find out the origin of Spam 2.0, demographic information, timestamp of different web requests, etc. Web usage data can also be used to analyse the behaviour of Spam 2.0 distribution. Additionally, it provides a model for human content contribution as well.

Submitted content contains information related to the contributed content on the Web 2.0 platform. Examples of such submitted content include: a comment, a reply to a discussion or a wiki page. This information can be used to analyse the nature of submitted content for Spam 2.0. Chapter 5 is focused on HoneySpam 2.0 and its conceptual model.

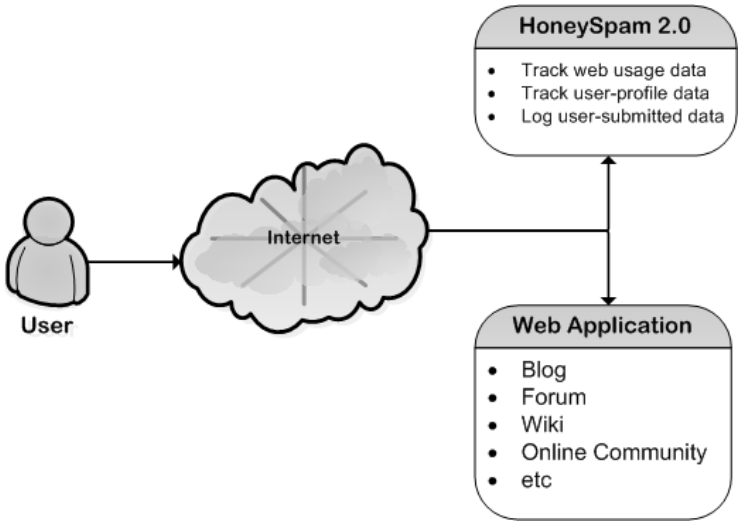


Figure 4.2: HoneySpam 2.0 simple deployment network diagram.

Figure 4.2 illustrates a simple deployment network diagram of HoneySpam 2.0.

Based on the initial result of HoneySpam 2.0, a more in-depth analysis of Spam 2.0 is presented in Chapter 6 including:

- Discovering uncovered spammers' behaviour on Web 2.0 platforms

- Understanding spammers' intentions on Web 2.0 platforms
- Formulating spammers' behaviour in order to develop discriminative feature sets
- Clustering spammers' activities into groups based on features sets
- Characterising spammers' behaviour in each cluster
- Developing a strategy to utilise feature sets for Spam 2.0 filtering

Chapter 6 presents a framework to investigate, formulate, and cluster Spam 2.0 behaviours. Three feature sets – Action Set, Action Time, and Action Frequency are presented accordingly to formulate spammers' behaviour on Web 2.0 platforms. A neural network clustering mechanism is used to cluster spammers' behaviour based on these three feature sets.

HoneySpam 2.0 along with more in-depth analysis of Spam 2.0 in Chapter 6 would address the following research issue:

- Research Issue 1: The nature of Spam 2.0 is not well understood

---

Action is a user set of requested web objects used to perform a certain task or purpose. For instance, a user can navigate to the registration page in an online discussion board, fill in the required fields and press on the 'submit' button in order to register a new user account. This procedure can be modelled as "Registering a user account" Action. Action is explained in Section 6.2.2.

4.3.2 Early-Detection based Spam 2.0 Filtering (EDSF)

A novel Early-Detection based Spam 2.0 Filtering (EDSF) method is proposed to filter out Spam 2.0 from Web 2.0 platforms. The EDSF is language-independent since it does not use any language-dependent features (e.g. content features) and can be extended to multiple platforms. The EDSF can be used inside multiple platforms since it exploits behavioural features that are common in Web 2.0 platforms. Additionally, it minimise network and storage resource by preventing submission of Spam 2.0 and blocking spammers activity inside the system.

The proposed method addresses the following issues.

- Research Issue 2: Existing Spam 2.0 filtering solutions are afterthoughts
- Research Issue 3: Platform and language-dependent approaches
- Research Issue 4: Current methods waste network and storage resources
- Research Issue 5: Prevention-based methods pose inconvenience for human users
- Research Issue 6: Inability to identify Web 2.0 spam users

The fundamental assumption in the EDSF is that spammers behave differently compared to genuine users within Web 2.0 platforms. Spam users' web usage behaviours are different from that of genuine users. Spammers develop or make use of technologies such as auto-submitters, spambots etc. (Hayati et al. 2010) to assist them in fast distribution of spam content. Therefore, spam users' web usage data can be different from that of genuine users.

EDSF makes use of web usage navigation behaviour to build up a discriminative set of features including Action Set, Action Time, and Action Frequency. Such a feature set is used later in a machine learning classifier known as Support Vector Machine (SVM) for classification.

Given that there is no work in the literature on Spam 2.0 behaviour classification, the development of EDSF are the initial steps toward identification of Spam 2.0 by utilising web usage behaviour.

Figure 4.3 presents the overview process of the EDSF:

- Step 1: Web usage data for the incoming traffic (e.g. user navigation through website) is implicitly gathered.
- Step 2: Based on gathered web usage data, Actions are formulated i.e. Action Set, Action Time and Action Frequency.
- Step 3: Actions are grouped into two sets: Spam Actions and Genuine Actions. The former is a representation of spam behaviour while the later is a representation of genuine human user behaviour.

- Step 4: A machine learning classifier (SVM) is trained based on set of known Spam and Genuine Actions. The classifier will be used to identify spam Actions in new incoming traffic. Users who perform spam Actions are marked as spammers and blocked from the system.

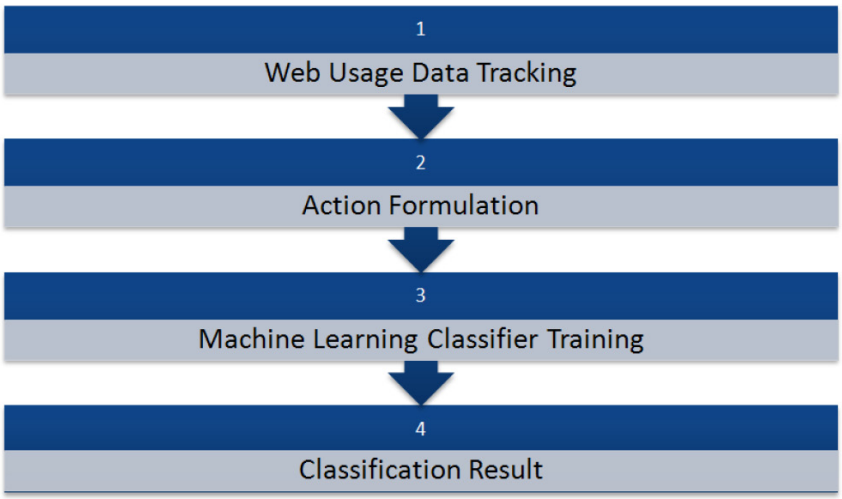


Figure 4.3: The overview process of the EDSF.

The detail of the EDSF is discussed in Chapter 7.

4.3.3 On-the-Fly Spam 2.0 Filtering (OFSF)

The EDSF can detect Spam 2.0 in Web 2.0 platforms. However, due to its design constraints and use of a machine learning classifier, it ca not be used for fast and on-the-fly identification of spam behaviour. Compared to ESDF, the OFSF method is capable of on-the-fly detection of spam behaviour in Web 2.0 platforms. Hence, OFSF can satisfy the following research issue that could not be address in the design of EDSF

- Research Issue 7: Machine learning based methods are not effective for real-time detection

A novel data structure and decision tree known as Trie has been utilised to provide on-the-fly functionality.

The OFSF tracks and records users' activities inside the system. After a certain threshold, the method can provide a decision as to whether or not a particular user is considered to be spamming. Based on previously observed behaviour of spam and genuine users, OFSF can provide a degree of certainty for the decision at each stage of execution. For example, while the user is filling out web forms and

navigating through web pages, the OFSF measures and investigates specific behavioural features (e.g. the amount of time taken to navigate between web pages) to determine the probability of spam behaviour.

The OFSF uses an extended version of Action to formulate web usage data. Here, each user Action is assigned an *Action Index Key*. An Action Index Key for each user is placed together to create an *Action String*. Action Strings are representations of user behaviour for a given period. Action Strings contain the sequence order of user Actions inside the system. The order of performed Actions provides a better insight into users' behaviour and intentions.

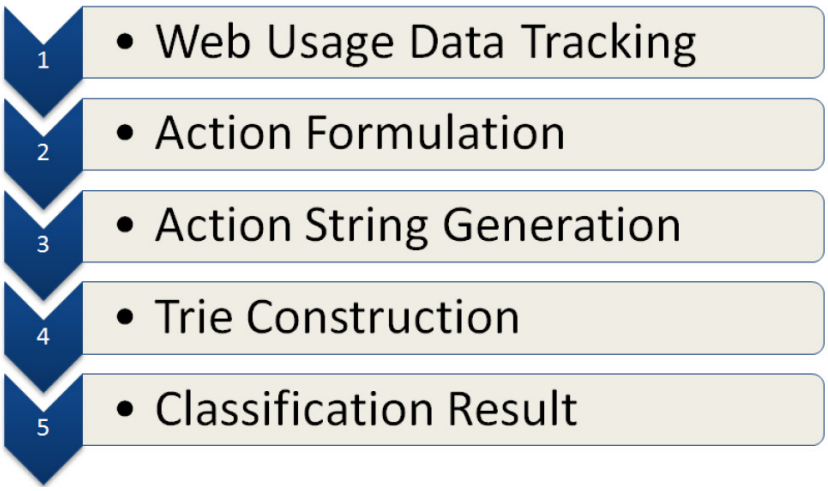


Figure 4.4: The overview process of the OFSF.

Figure 4.4 illustrates the overview process of the OFDF.

- Step1: Web usage data for the incoming traffic (e.g. user navigation through website) is implicitly gathered.
- Step 2: Based on gathered web usage data, Actions are formulated.
- Step 3: Each Action is assigned an Action Index Key. The Action Index Key is placed in order to generate Action Strings.
- Step 4: Action Strings are used to construct the classifier based on Trie. Spam and non-spam probability value is associated with each length of Action String and added to the Trie.

- Step 5: Each new Action String is fed into the classifier and is assigned a spam behaviour probability value. If this value is greater than a given threshold, the activity is marked as spam.

The detail description of the OFSF is discussed in Chapter 8.

This concludes the overview of the proposed solutions presented in this dissertation. A comparison of the proposed solutions is discussed in the next section highlighting the main differences.

4.4 Comparative view of the proposed solutions

As mentioned earlier in this chapter, EDSF and OFSF are two Spam 2.0 filtering methods which are proposed in this dissertation to combat Spam 2.0. The similarities between these two methods are as follows.

- Early-detection based: both methods are early-detection based and they investigate behavioural data.
- Usage of Web usage data: both methods gather web usage data to formulate Actions.
- Supervised: both methods require a pre-set of spam and non-spam behaviour for classification.
- Behavioural based: both methods investigate web usage behavioural data and Spam 2.0 generation models to identify spam behaviour.

The following is a list of differences between the EDSF and OFDF methods.

- Machine learning classifier vs. a novel classifier based on Trie: EDSF uses a machine learning classifier to identify spam behaviour while the OFDF uses a novel classifier based on Trie to represent Action Strings and compare new Action Strings based on constructed Trie. Each Trie node contains the probability of a specific Action String being either genuine or spam.
- Offline vs. Spontaneous detection: EDSF requires complete feature vectors to use inside the machine learning classifier and it might not be applicable in online scenarios. OFSF can

classify an incoming Action String while a user is using the system (i.e. with incomplete feature vector).

- Set vs. Sequence: An EDSF feature vector is an Action set while OFSF is an ordered sequence of Action Strings. The order of Action can provide better insight into user's behaviour on the Web 2.0 platform.

This concludes the comparison of the proposed solutions. The following section describes the research approach that is adapted in this research. A science and engineering-based research approach is followed in this research and is discussed in the next section.

4.5 Conceptual Process

The overview of the proposed solution has been discussed in Section 4.2 . In order to develop these solutions, a conceptual process needs to be followed. The conceptual process has the following 7 steps.

1. Requirement Elicitation
2. Design Rationale
3. Theoretical Foundation
4. Prototype Implementation
5. Experimental Setting
6. Results and Observation
7. Validation and Comparative Analysis

The above steps are consistent with a science and engineering approach which is adopted here. This section discusses the conceptual process which is applied throughout all the three research projects covered in this dissertation.

4.5.1 Requirements Elicitation and Prioritisation

This stage starts with a survey of existing solutions and identification of main issues relevant to these solutions. The issues are then prioritised according to their importance. The prioritised issues act as a framework for the proposed solution.

4.5.2 Design Rationale

The second stage in the conceptual process is design rational. This stage describes the basic design decisions to meet the requirement elicited in Stage 1. All the requirements need to be considered and contribute to the design.

4.5.3 Theoretical Foundation

The third stage in the conceptual process is theoretical foundation. This stage is based on the design rationale which comes from Stage 2. This stage provides a solution to address all the requirements elicited in Stage 1 by analysing the design rational in Stage 2. A detailed algorithm is proposed in this stage to cover all the needs.

The feasibility, computational complexity, scalability, real time deployment and reliability of each design decision are investigated. The final algorithm is the accumulation of all the design decisions that meet these criteria.

The final algorithm needs to be tested for the functionality purposes. Later on, the results of the tests can be used as insights to optimise the initial algorithm. To test the algorithm, it needs to be implemented which is described in the next stage.

4.5.4 Prototype Implementation

The fourth stage in the conceptual process is prototype implementation. This stage provides an implementation of the theoretical foundation and prototype algorithm. The prototype is used to test and experiment and validate the theoretical foundation.

4.5.5 Experimental Setting

The fifth stage in the conceptual process is experimental setting. This stage provides some test cases to examine the prototype before conducting further experiments. The test cases analyse performance of the prototype in different conditions and real world scenarios. Once a reliable result is obtained from the prototype, the experiment can be conducted to investigate the prototype.

4.5.6 Results and Observation

The sixth stage in the conceptual process is concerned with results and observation. This stage discusses the experiments of running the prototype. It also analyses the experiments and describes the observations. For example, a spam filter examines real-world spam data contents to identify the

performance of spam filter classification. The observations are then cross-related with the theoretical foundations to validate the results.

4.5.7 Validation and Comparative Analysis

The final stage in the conceptual process is validation and comparative analysis. This stage compares the result from the experiments (actual results) with the theoretical model (expected results). The expected results might be similar to or quite different from actual results. In the case of difference, the theoretical model needs to be adjusted and all the steps from stages1to 6 might need to be amended. After validation of the expected results, the proposed solution is compared against existing solutions to discover its strong and weak points. Comparative analysis provides new avenues for future research.

4.6 Conclusion

This chapter provides an overview to the solutions that are proposed in this dissertation. These solutions are proposed to address research issues that have been discussed in Chapter 3.

Three solutions that have been discussed in this chapter are as follows.

- Characterising Spam 2.0 and Spam 2.0 distribution model: Current literature lacks a comprehensive understanding of Spam 2.0 and the way it is propagated on Web 2.0 platforms. In this dissertation, these problems are addressed by providing a mechanism named HoneySpam 2.0 to monitor, track and store Spam 2.0. This information is later analysed and investigated to discover the characteristics of Spam 2.0 and its distribution model.
- Early-Detection based Spam 2.0 Filtering (EDSF) method to implicitly detect spam users, be language and platform independent and minimise user inconvenience: The early-detection based Spam 2.0 filtering methods attempt to identify spam behaviour prior to or just after spam content submission. Such methods investigate the behaviour of a spam submitter, e.g. automated tools, spambots etc.
- One-the-Fly Spam 2.0 Filtering (OFSF) method to spontaneously identify and prevent spam users: OFSF makes use of web usage navigation behaviour to establish discriminative feature sets for spam users detection. OFSF considers the sequence and order of user web navigation in order to build up the feature sets for classification. Unlike EDSF, OFSF uses a novel rule-based classifier to store, retrieve, and classify spam users separately from genuine human. OFSF makes it possible to spot spam users on-the-fly. While users are interacting with the Web 2.0 platform, OFSF is able to track and identify possible spam navigation patterns.

The conceptual process used to develop these solutions has also been discussed in this chapter.

A detailed discussion of these solutions is the focus of Chapter 5 to Chapter 8.

Chapter 5

Tracking Spam 2.0

This chapter covers

- Introduction to HoneySpam 2.0 as a mechanism to understand Spam 2.0.
- Characterising of Spam 2.0.
- Experimentation and testing of HoneySpam 2.0 on real data.
- Validation and comparative study of HoneySpam 2.0 against state-of-the-art solutions.

5.1 Introduction

As mentioned earlier in this dissertation, current literature lacks a comprehensive understanding of Spam 2.0 and the way it is propagated on the Web 2.0 platforms. This chapter addresses these problems by providing a mechanism named **HoneySpam 2.0** to monitor, track and store Spam 2.0. This information is later analysed and investigated to discover the characteristics of Spam 2.0. HoneySpam 2.0 is tested against real-world Spam 2.0 data and its performance is compared against existing solutions.

The theoretical foundation for HoneySpam 2.0, along with its algorithm design to address all the requirements laid out in Chapter 4, is also discussed here. Finally, the details of tests, experiments and observations of the results along with their validation are presented out in this chapter.

In this chapter, *Unified Modelling Language* (UML) is used to visualise and present the artefacts of HoneySpam 2.0. UML is a *de-facto* modelling language used in software engineering to visualise, document and model *object-oriented* software (Pilone & Pitman 2005). UML has been used widely to model software.

This chapter is organised as follows. A general overview of the HoneySpam 2.0 framework is presented in Section 5.2. The requirements and design rationale of the HoneySpam 2.0 framework

development is presented in Sections 5.3 and 5.4 respectively. The theoretical foundation of the proposed framework along with its algorithms is covered in Section 5.5. Section 5.6 presents the prototype of the HoneySpam 2.0. Experimental settings are discussed in Section 5.7 Finally, results of experiments along with validation and a comparative analysis are presented in Section 5.8 and 5.9 respectively. Chapter 5 is concluded in Section 5.10.

The following section describes the main requirements and design rationale for the HoneySpam 2.0 that represents stage 1 and 2 of the conceptual process.

5.2 General Overview of HoneySpam 2.0

Spam content is prolific on the web as spammers no longer need to purchase, host and promote their own domains. This is due to the development and widespread adoption of Web 2.0 platforms. Spammers are now able to use legitimate websites to host spam content. Examples include fake eye-catching profiles in social networking websites, fake promotional reviews, responses to threads in online forums with unsolicited content and manipulated wiki pages. This new generation of spam is referred to as Spam 2.0 as defined in Section 1.5.1.

Little is known about the nature of Spam 2.0 and its distribution model. As discussed in Chapter 2, existing Spam 2.0 filtering solutions are not effective in detecting and preventing Spam 2.0 and there is no clear understanding of Spam 2.0 characteristics. Therefore, this research initially proposes a solution to investigate and study Spam 2.0. The proposed solution includes two parts – tracking and analysing. The former is done through a framework named HoneySpam 2.0, the latter through statistical analysis.

HoneySpam 2.0 draws on the idea of honeypots – a technique used to attract and track cyber attackers (Spitzner 2002). It implicitly tracks web usage data which includes detailed monitoring of click streams, page navigation, forms, mouse movements, keyboard actions and page scrolling. HoneySpam 2.0 is a tracking framework that can be integrated in any web application ranging from blogging tools to social networking applications. The tracking data is later analysed to generate initial reports on Spam 2.0 characteristics. Extensive profiling and characterising of Spam 2.0 is the focus of Chapter 6.

The main objective of this chapter is to:

- Present a novel framework to monitor, track and store Spam 2.0.
- Demonstrate the potential of HoneySpam 2.0 to monitor and track Spam 2.0.
- Statistically analyse Spam 2.0 and its generation model.
- Provide a study of the strengths and limitations of the proposed solution.

Given the brief insight into the research conducted in this chapter, the following section discusses the requirements for this research.

5.3 Requirements

This section represents the *first* stage of the conceptual process where requirements are elicited and prioritised. The following requirements are laid down for the proposed solution.

1. **Openness**: The solution should gather real-world examples of Spam 2.0 and be further extended to include different types of Spam 2.0 activities.
2. **Implicitly**: The solution should implicitly accumulate data from spam users and their interactions.
3. **Analysability**: The captured data needs to be complete and analysable for future investigation. The captured data needs to be traceable, and it should be easy to weed out useless data.
4. **Security**: The solution should be deployed in a secure environment. It should capture and store spammers' data in a safe place and deny any unauthorised access to the other parts of the system. Otherwise, the system could be used for malicious purposes such as hosting malicious executable programs, illegal content, etc.
5. **Integrity**: The captured data should be safe from malicious and genuine human tampering.
6. **Accessibility**: The solution should be accessible and discoverable for capture spam tactics.

This concludes stage 1 of the conceptual process. The design rationale which is discussed in the following section is based on the abovementioned requirements. A design decision for each

requirement is made in the design rationale accompanied by a discussion of how the requirement is fulfilled.

5.4 Design Rationale

This section provides a design rationale to fulfil the requirements that is outlined in Section 5.3. This is the *second* stage of the conceptual process. The following design decisions are proposed in order to address each requirement.

1. Develop a plug-in based solution using real-world best practices that can be integrated to multiple (live and operating) Web 2.0 platforms. This would satisfy the requirement for gathering real-world examples of Spam 2.0 and different types of Spam 2.0 activities (Req. 1).

2. Capture server-side data (e.g. web/form usage data) rather than executing scripts on the client-side to capture data. This would satisfy the requirement for implicit tracking for of data (Req. 2).

3. Assign identifiers to incoming traffic in order to trace users operating on the Web 2.0 platform (Req. 3).

4. Host the framework in a secure environment with an isolated network connection in order to decrease the risk of security attacks on the local network (Req. 4).

5. Manually monitor the execution of the framework in order to eliminate malicious or genuine human data from spam users' data (Req. 5)

6. Develop marketing/promotional strategies to advertise the URL of the hosted framework, i.e. HoneySpam 2.0 through different channels in order to attract spam users (Req. 6). A detailed discussion of this design decision is provided in Section 5.7.3 .

In the development of any solution, the above design decisions should be considered. All these design decisions are included in the proposed solution to track and characterise Spam 2.0. This concludes stage two of the conceptual process. The following stage is the theoretical foundation for the proposed solution.

5.5 Theoretical Foundation

The theoretical foundation for HoneySpam 2.0 discussed in this section represents the *third* stage of the conceptual process. The theoretical foundation of HoneySpam 2.0 has three main parts (Figure 5.1): (1) tracking, (2) pre-processing and (3) knowledge discovery.

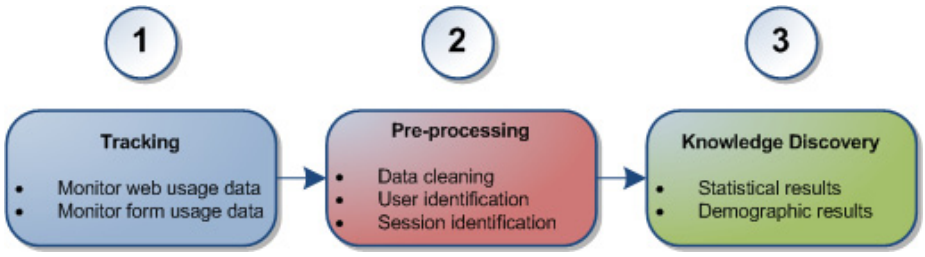


Figure 5.1: Theoretical foundation of HoneySpam 2.0

The first part of the framework is designed for the *tracking* mechanism. The tracking part provides input data for other parts of the system. The input data consists of navigation usage data (i.e. IP address, referrer URL, browser identity etc) and form usage data (i.e. mouse movement, keyboard actions, form load/unload, form's field focus/un-focus etc.).

The second part is pre-processing which includes tasks such as *data cleansing*, *user identification*, and *session identification*. In the data cleansing task, irrelevant records would be removed. User identification is the task of associating each usage data with a specific user. Finally, session identification refers to splitting usage data into different sessions. Each session contains information about usage data for each user's visit.

Knowledge discovery is the last part of the proposed solution. The output of the previous part of the system would be used here for mining and characterising Spam 2.0. General data mining techniques including statistical and demographic analysis are used here.

The proposed solution explained above addresses all of the requirements described in Section 5.3 . The next sections provide a detailed description of each part of HoneySpam 2.0.

5.5.1 Tracking

The tracking part of HoneySpam 2.0 is capable of tracking spam users on the Web 2.0 platforms, and monitoring Spam 2.0 distribution behaviour. The tracking part consists of two main components:

Navigation Tracking Component (NTC) and *Form Tracking Component* (FTC). Figure 5.2 illustrates the architecture of the tracking part.

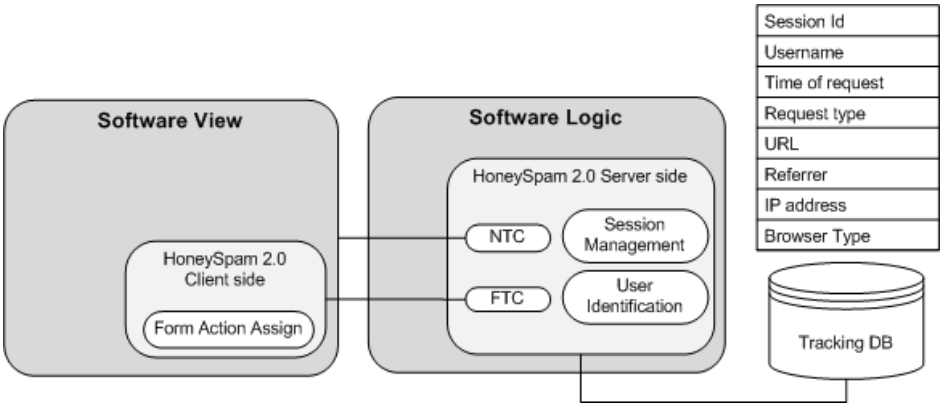


Figure 5.2: HoneySpam 2.0 tracking part architecture.

5.5.1.1 Navigation Tracking Component

This component is designed to track web page navigation patterns and header information of incoming traffic. The following information is captured by this component:

- **Requested URL**: The requested URL is the web page address that is requested by the client. This contains the domain name address and the path and filename of the requested web page. Occasionally, the requested URL contains information about the parameters that are passed via the web page (e.g. *http://www.example.com/dir/file.php?parameter=value*).

- **Type of request method**: NTC captures two most common types of methods for every HTTP request (i.e. GET and POST method). A GET method is used to retrieve or send data to the web server. To retrieve data, the URL of the web object is sent to the web server and in return the content of the web object or the data-producing process would be passed to the client. To send the data, the form data is encoded and attached to the URL. This URL is sent to the web server. A POST method is used for submitting data to the web server. The data is included in the body of the request. The requested URL here is not used for retrieving; it is a program that can handle the submitted data. The POST method is usually used for submitting form data.

- **Time of request** (timestamp): The time of the request is stored as a *UNIX timestamp*. UNIX time is a well-known time format to record time. It is in the format of seconds elapsed since midnight *Coordinated Universal Time of January 1, 1970*.

- **IP address**: Internet Protocol (IP) address is a unique numerical label assigned to each device connected to the Internet. NTC records the IP addresses to identify each user.

- **User Agent**: The User Agent contains information about the software program used by the client. It contains the software name, its version and some other optional information such as operating system name, operating system version etc.

- **Referrer URL**: Suppose a web page A has a link to a web page B. If a user navigates from A to B, web page A is the referrer URL of B. NTC stores this information to find the origin of each request, determining whether it comes through internal web pages or other websites. For example, the referrer URL is very useful in discovering whether a user was directed to the website through the search engines. The search query can be extracted from the referrer URL.

Figure 5.3 explains sample data that has been captured by NTC.

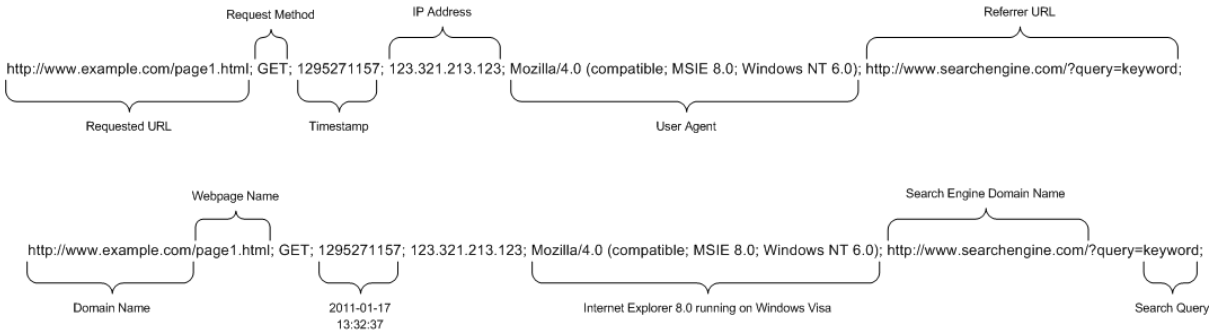


Figure 5.3: Sample data captured by NTC.

5.5.1.2 Form Tracking Component

Other than the NTC, HoneySpam 2.0 has additional functionality that can provide insights into how spam users use online forms. It is a common belief that spambots do not interact with forms (Hayati et al. 2009); however, this research aims to examine this aspect here. The form tracking component is designed to investigate form input behaviour. The form Tracking Component (FTC) captures and stores the following form related data:

- **Mouse clicks and movements**: The mouse clicks and movements capture whether the client is using a mouse when using forms. The timestamp for mouse movements on forms filed is transmitted to the server.

- **Keyboard action**: The keyboard action checks whether the client is using a keyboard when filling up fields in a form. The timestamp when the user presses any key along with key code is captured by the FTC and is then transmitted to the server.

- **Form field focus and un-focus**: An online form consists of various fields. Once the user clicks or selects a field in the form, the *Field Focus* action would be triggered. When the user un-selects or moves to other fields, the *Un-focus* action would be triggered. The Form Field Focus and Un-focus can show users' form field interaction as well as the amount of time taken to fill each field.

- **Form load and submission**: Similar to the Form field focus and un-focus, the *Form load* and *submission* can show users form interaction as well as the amount of time that is taken to fill the whole form.

FTC uses *Asynchronous JavaScript and XML (AJAX)* technology to capture the above information and transmit it to the web server (Zakas et al. 2006). AJAX provides data exchange with a server without reloading the whole web page. For reasons of practicality and implicit data collection, AJAX has been implemented to capture user-form interaction and transmit such data to the server behind the scenes. Other technologies (e.g. traditional HTML-based loading) interfere with user and system interactions; hence, they do not meet the framework's requirements (Holdener III 2008).

There are many other form events that can be captured, such as selection of elements, release of a key, occurrence of error etc. However, the above form usage data satisfies the requirements. Moreover, for simplicity and extendibility of the proposed solution, only the abovementioned form actions are captured.

Figure 5.4 presents the sequence diagram for the tracking part, the process of which is as follows:

Step1: User requests a web page that contains a form.

Step2: NTC first captures the web usage data associated with initial request.

Step3: NTC forwards the request to the web application to be processed.

Step4: Web application returns the produced web page.

Step5: HoneySpam 2.0 attaches its FTC client code (AJAX) and sends it back to the user.

Step6: Once the form has been loaded in the users' browser and the user starts to interact with the form, the FTC client code starts to aggregate form usage data and implicitly transmits it to HoneySpam 2.0.

Step7: Once the user submits the form, NTC captures the web usage data of the incoming requests.

Step8: HoneySpam 2.0 passes the form submitted data to the web application for future processing.

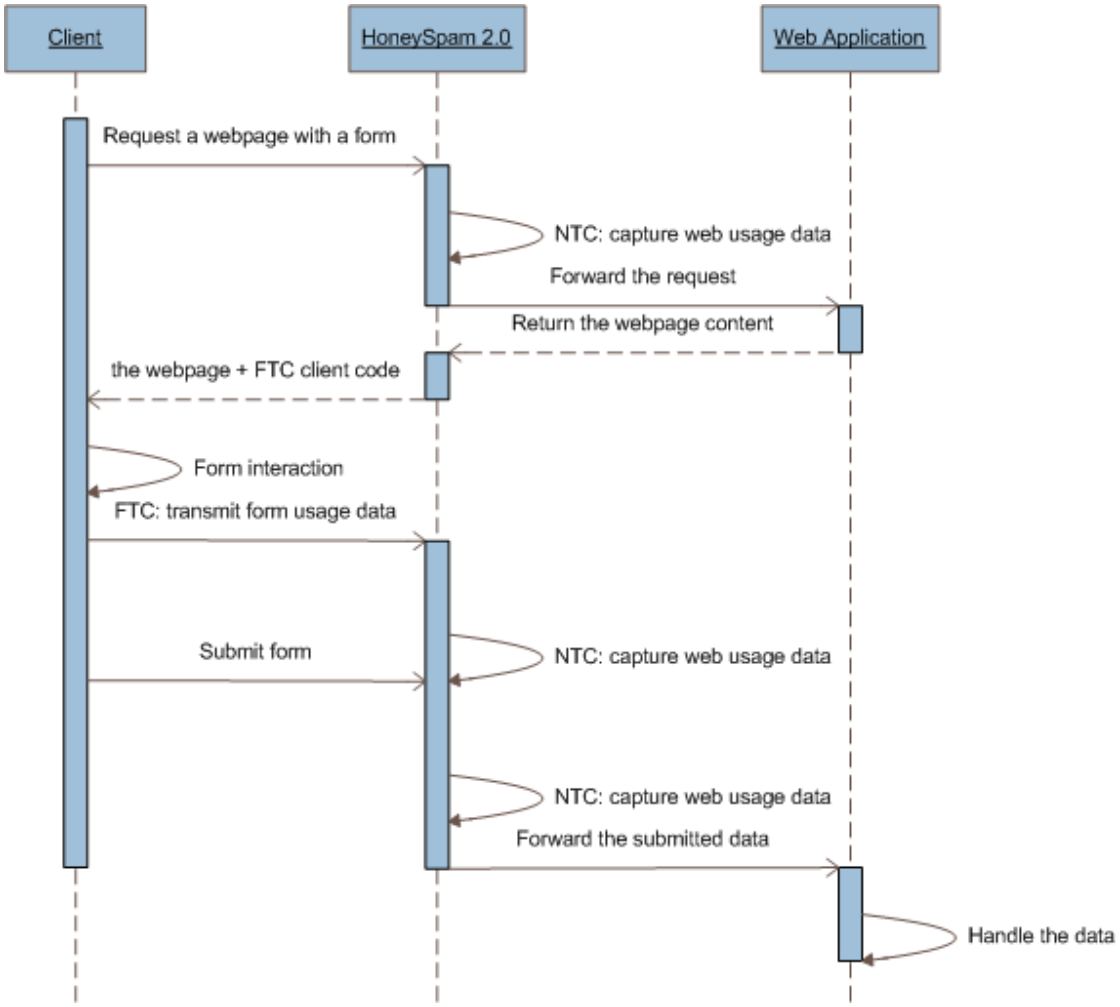


Figure 5.4: Sequence diagram for HoneySpam 2.0 tracking process.

5.5.2 Pre-processing

Pre-processing the captured data is the second part of HoneySpam 2.0.

Input: Raw form and navigation usage data

Output: Session and user identified usage data

There are three main tasks: data cleansing, session and user identification.

5.5.2.1 Data cleansing

Data cleansing is the process of removing any irrelevant records from the data. The data stored by HoneySpam 2.0 needs to be accurate, precise, and clean from noise.

As stated before, NTC captures only the web page requests. However, in real practice, each HTTP request contains separate requests for other entities inside the web page such as graphics, style and script files. This is done automatically by the user's browser and not explicitly by the user. NTC ignores other requests since the main purpose is to study users' navigation. Other automatic requests do not provide valuable information for this study since the user does not explicitly request them. Therefore, all the requests that contain URL suffixes as shown in Table 5.1 are removed.

Table 5.1: URL suffixes that need to be removed from the capture data.

Graphic files	Script files	Animation files	Style files
.jpg, .jpeg, .gif, .png, .tiff, .bmp, .ico	.js	.flv, .swf	.css

Additionally, the data related to researchers monitoring the forum and the data related to known web crawlers and robots is also removed from the results.

5.5.2.2 User Identification

User identification is the task of identifying users from the web usage data (Robert Cooley et al. 1999). HoneySpam 2.0 is designed to study spammers' behaviour on a Web 2.0 platform. Hence, it is important to identify unique spam users on web applications in order to discover different spam behaviours. Previous studies show that user identification is a complicated task (Robert Cooley et al. 1999), since web usage data does not provide enough information to differentiate requests between different users. User identification becomes even more complicated with the existence of:

- **Proxy and cache servers**: These servers are designed to speed up access to a specific web resource. A proxy server caches previous requests and returns subsequent requests without connecting to the remote server. Therefore, it makes user identification very challenging.

- **Local Internet browser cache**: Similar to the caching in the proxy server, here the local user's Internet browser caches requests so future requests for that data can be served faster.

To deal with these problems, some heuristics are used in an attempt to identify unique users:

- **IP address**: A change in IP address can represent different users. However, multiple users might share the same IP address. This might be confused as a single user.

- **User Agent**: A change in the User Agent field in the same IP address can represent a different user. However, one user using two different Internet browsers in the same machine can be mistaken as multiple users.

Therefore, the above heuristics are not capable enough of effectively discovering unique users (R. Cooley et al. 1997).

In order to overcome the above problems, the proposed method uses a different approach to identify unique users. The majority of web applications require registration of a valid username before further interaction with the system. Moreover, users need to login to the system in order to use it. Each request to the web server needs to be authenticated before any processing occurs. It is not convenient for users to be asked for their user account credentials each time they send a request to the web server. A user's credentials can be stored in the user's machine and can be automatically sent to the web server with each request. This piece of information that is stored in the user's computer is known as *HTTP cookie* (or *cookie*). Cookies can be used for a variety of purposes such as authentication and *sessionisation*. It is sent through the HTTP header to the web server.

In the proposed method, the user credentials are extracted from each user's request to the web server and are stored along with other navigational usage data. Therefore, each web request is associated with a username. This information can be used to discover unique users from the web usage data.

To characterise Spam 2.0 behaviour, it is necessary to investigate spam users' behaviour on Web 2.0 platforms. After the user identification step, the usage data for website visitors (i.e. users who do not have user account) is removed. Website visitors cannot contribute to content generation unless they sign up to the web application.

5.5.2.3 Session identification

Session identification or sessionisation is the task of identifying a user's web usage behaviour during each visit (R. Cooley et al. 1997). Each session contains specific information about the usage data for each website visit.

Previous studies use simple heuristics to divide usage data into different sessions. For example, if the time between different requests exceeds a given time-limit, it is considered as a new user session (R. Cooley et al. 1997). Many studies use 30 minutes as a time-out for each session (Robert Cooley et al. 1999). However, such a simple heuristic cannot be strong enough to discover all the user's sessions. For example, it might take more than 30 minutes for a user to reply to a post in the forum; therefore, this might be mistaken for two sessions.

In order to overcome this problem, the proposed method produces a *session cookie* for usage data to differentiate new sessions. A session cookie is a type of cookie that is valid only while users use a website. Once users close their Internet browsers, or if they do not visit the website for a certain amount of time (30 minutes), and logout/login to the website, the session cookie would expire. For each session cookie, the proposed method assigns a unique session number (session ID) to differentiate user sessions.

Figure 5.5 presents a flowchart for the pre-processing part of the proposed method. The process is as follows.

Step1: The pre-processing tasks start by obtaining HTTP requests.

Step2: Unneeded data is eliminated by the data cleansing task.

Step3: The next two steps – user and session identification - are done in parallel. For user identification, the HTTP cookie header is investigated. The username is extracted from the cookie and associated with the HTTP request. Session identification is done through session cookies. A session ID is assigned to each session cookie to track the user during each session.

Step4: All HTTP requests that do not contain any usernames are removed.

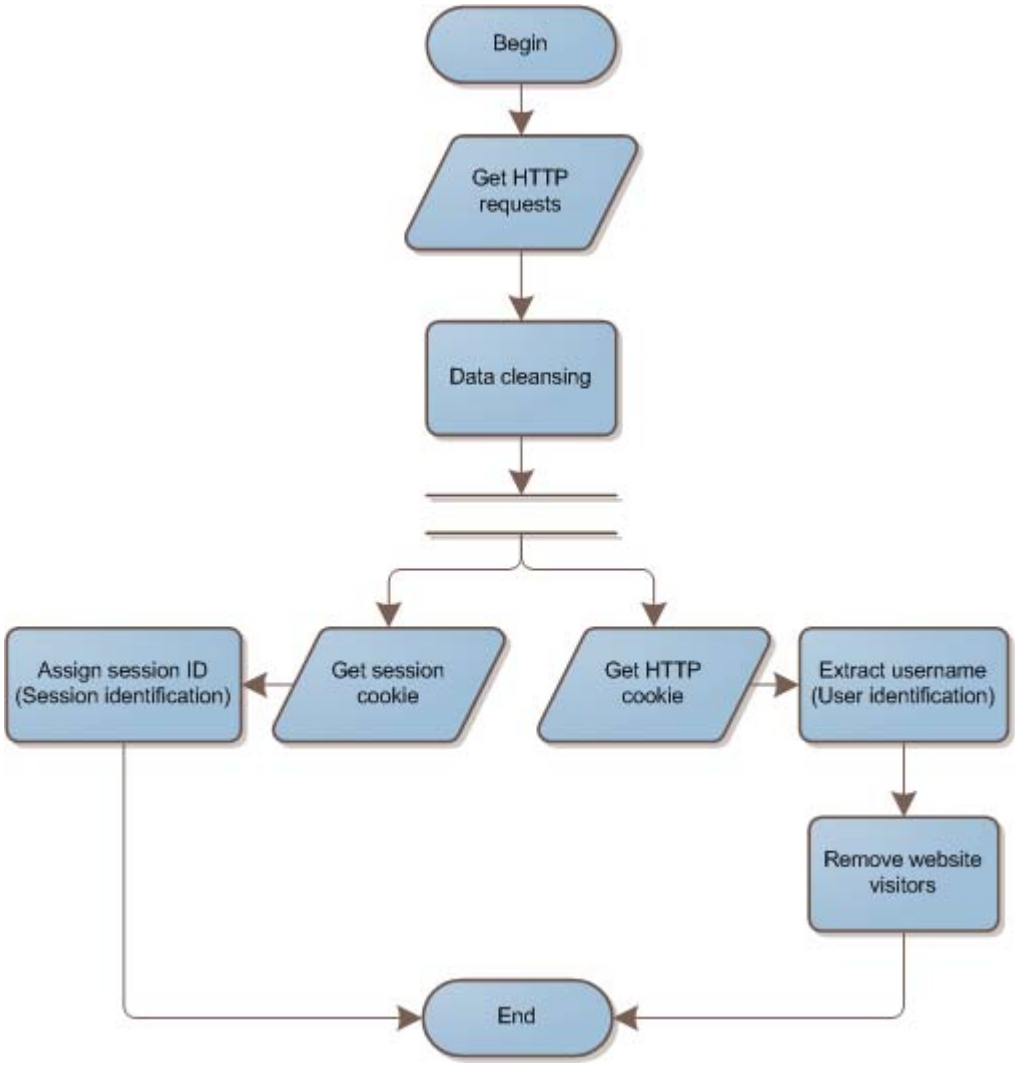


Figure 5.5: Flowchart for pre-processing step.

The above process makes the data ready for the next step – knowledge discovery.

5.5.3 Knowledge discovery

Knowledge discovery is the third and the last part of HoneySpam 2.0.

Input: Session and user identified usage data

Output: Statistical report of spam users' activity (e.g. usage and demographic analysis)

The pre-processed data from the previous part needs to be analysed to reveal characteristic of Spam 2.0.

In this part of the proposed method, general data mining techniques are used to generate usage statistics as follows:

- Web page visit dwell time
- Form usage dwell time
- Field focus dwell time
- Amount of content-submission per hour/day
- Frequency of Internet browsers
- Referrer URLs
- Return rate
- Submitted content topics
- Demographic analysis (e.g. country name)

Web page visit dwell time

The web page visit dwell time is the amount of time a user spends on each web page. It can be calculated by looking at each web page timestamp. Dwell time is defined as:

$$d = t_{i+1} - t_i \quad where\, 1 < i < |S| \qquad 5.1$$

d is dwell time for i^{th} requested web page in the session *S* at time *t*.

It is clear from the dwell definition that it is not possible to calculate dwell time for the last visited page in a session. For example, a user navigates to the last page on the website then closes his/her web browser. To overcome this problem, the average dwell time spent on other web pages in the same session is considered as the dwell time for the last page.

The rationale for investigating this measurement is to study how long spam users spend on a website. It is generally understood that spam users do not spend as much time as genuine users. This claim can be investigated through this measurement.

Form usage dwell time

The form usage dwell time is the amount of time a user spends to fill in all the fields in the form and then submit that form. The form usage dwell time can be calculated through E.q. 5.2.

$$d' = t_\alpha - t_\beta \quad where\, t_\beta < t_\alpha\ \alpha, \beta \epsilon S \tag{5.2}$$

α is a web page which contains the form and β is a web script that process the submitted form data. The rationale for investigating this measurement is to study the spam users' form usage.

Field focus dwell time

The field dwell time can be calculated through each field-focus and un-focus timestamp. The field dwell time definition is as follows.

$$d'' = t_u - t_f \tag{5.3}$$

t_u is field-unfocus event's time while t_f is the field-focus event's time. The rationale for investigating this measurement is to study whether or not spam users are using a mouse or keyboard to interact with the website. It can also show the speed of spam users' form completion.

Amount of content-submission per hour/day

The amount of content submission per hour/day represents how frequent spam users distribute spam content on Web 2.0 applications. It can also provide insight into the speed of Spam 2.0 distribution in a sample web application. This amount would be measured in terms of number of posts, replies etc.

Frequency of Internet browsers

The frequency of Internet browsing can provide a figure on the data mimicked by spam users in the User Agent field of HTTP header. Spam users use their own tools and not necessarily common Internet browsers. This can be investigated through this measurement.

Referrer URLs

Referrer URLs can indicate the origin of user navigation - whether it was through search engines, other website, or direct visits. Hence, this measurement can provide insights into how spam users find their targets.

Return rate

The return rate can show whether or not spam users are willing to distribute Spam 2.0 to targets that have already received their spam. It can also show whether or not they use the same user account to distribute spam content.

Submitted content topic

The submitted content topic provides insight into the nature of the Spam 2.0 content. It shows the frequency of top content categories and Spam 2.0 topics.

Demographic analysis

The demographic analysis provides insight into the origin of each spam user. By examining the IP address and hostname, it is possible to discover the request's country of origin.

Figure 5.6 illustrates the flowchart for the knowledge discovery part of HoneySpam 2.0. The process consists of the following steps.

Step1: Content, web and form usage data is gathered from the previous part of HoneySpam 2.0.

Step2: Form usage data including form and field dwell time are measured based on form usage data

General usage reports including content submission rate, frequency of internet browsers, frequency of referrer URLs, return rate and demographic are generated based on web usage data

The content theme is generated based on the submitted content.

Step3: The above reports are used to generate statistical reports.

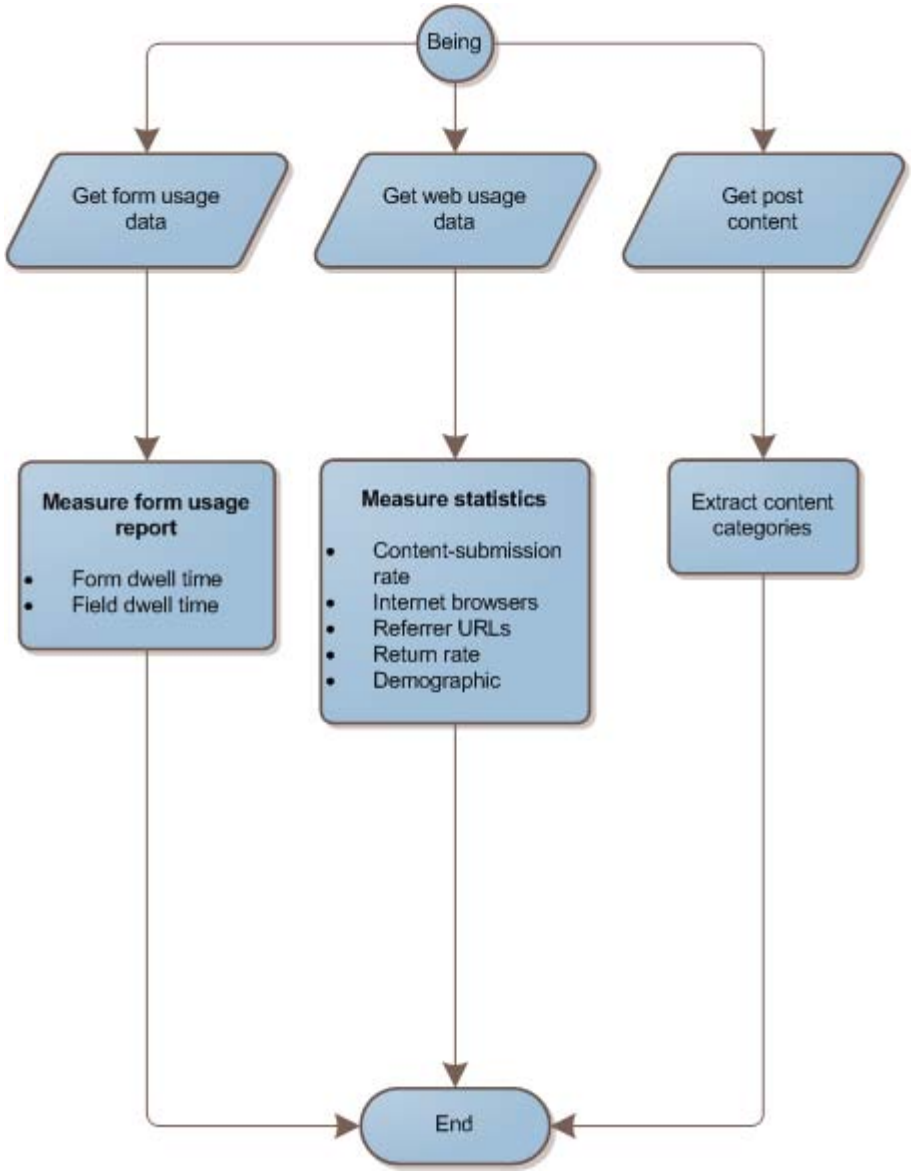


Figure 5.6: Flowchart showing knowledge discovery process of HoneySpam 2.0.

5.6 Prototype Implementation

The latest, commonly-used technologies are used to implement each part of HoneySpam 2.0.

Tracking part: To implicitly capture real examples of Spam 2.0, web development techniques best-practices are used to implement the proposed method (Cosentino 2002). This allows HoneySpam 2.0 to be operational in real-world scenarios. Additionally, it is necessary for the test environment to provide real-world inputs for HoneySpam 2.0.

To implement and test the proposed method, server-side programming and scripting languages are used. HoneySpam 2.0 server side code is implemented by using *PHP*. The database for collecting the

captured data is implemented using *MySQL* engine. MySQL is a type of relational database management system (RDBMS) and the data is stored in the form of tables (e.g. columns and rows) (DuBois 2006). Structured Query Language (SQL) is used to access and manage the MySQL database including insert, update, delete, and query the data and the schema (Molinaro 2005). The HoneySpam 2.0 client side code (FTC) is implemented using AJAX.

PHP is selected due to its popularity as a web programming/scripting language. Additionally, the majority of the popular Web 2.0 platforms are programmed in PHP, JavaScript and the AJAX technique is selected because it is well accepted for programming live and interactive web applications (Holdener III 2008). Additionally, most of the Internet browsers currently support JavaScript hence HoneySpam 2.0 client side code can be executed in the majority of the Internet browsers.

This makes HoneySpam 2.0 adaptable with current web applications and therefore, HoneySpam 2.0 can easily integrated in live web applications to capture and characterise real-world examples of Spam 2.0.

Pre-Processing part: The pre-processing part is implemented using PHP language. The PHP codes fetch the data from MySQL database tables and perform the pre-processing tasks. The output is saved in a special file format or directly in the database for further processing.

Knowledge Discovery part: The last part of HoneySpam 2.0 is implemented using PHP and MATLAB. MATLAB is a well-known tool for data processing. By having a significant amount of features, MATLAB can facilitate data analysis. In the proposed method, MATLAB is used for generating general statistical reports. The PHP codes are used to prepare the input data for MATLAB.

Two common test environments – online discussion board and social bookmarking system – are selected to test HoneySpam 2.0. An online discussion board is implemented using the common *Open Source* forum, *Simple Machine Forum* (SMF) (SMF Team 2011). Social bookmarking system is implemented using *Pligg*, a common social bookmarking Web 2.0 platform (Heikkinen 2011).

5.6.1 Tracking part

The tracking part of HoneySpam 2.0 is composed of FTC and NTS. It is modelled using 3 PHP and 2 JavaScript files as follows:

- sb_unb.php
- sb_mysql.php
- sb_ft.php
- sb_ft.js
- sb_form_assign_[FORM NAME].js

The PHP files are executed on the virtual web server operating with *Linux* (Suse Server Enterprise 11), *Apache* server and PHP compiler version 5.

As mentioned earlier, adaptability is one of the main design factors in HoneySpam 2.0 since it is required to be integrated with multiple Web 2.0 platforms and capture the data. HoneySpam 2.0 *hooks* its custom commands and specialised features to the web application in order to process and capture the usage data. The following sections provide a detailed description of the tracking part of HoneySpam 2.0. Figure 5.7 presents the class diagram for the NTC.

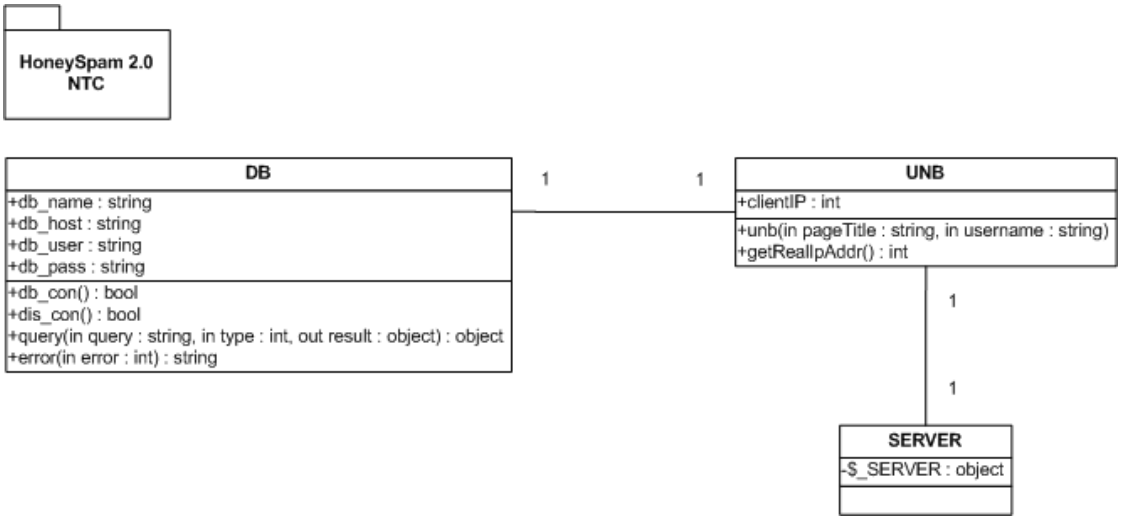


Figure 5.7: Class diagram for the NTC.

A class diagram in UML is a static diagram to model objects, attributes, operations, and interactions among object in the software (Miles & Hamilton 2006)**. The upper part of the class diagram shows class name, the middle part holds class attributes, and the lower part contains the methods (operations) of the class.**

5.6.1.1 Navigation Tracking Component (NTC)

Navigation Usage Tracking (NTC) is coded in the *sb_unb.php* and *sb_mysql.php* files. *sb_unb.php* contains an algorithm to capture HTTP header data, extract user account information from HTTP cookies, and handle session ID in a session cookie. *sb_mysql.php* contains the *DB* class to interact with the MySQL database, by inserting, fetching, updating, and removing the captured data.

sb_mysql.php

sb_mysql.php defines the *DB* class to handle the connection to MySQL database. Figure 5.8 illustrates the class diagram for *sb_mysql.php* file. This class is used for all MySQL connections.

As shown in Figure 5.8, three variables contain MySQL database credential information and four methods are used to handle MySQL interactions.

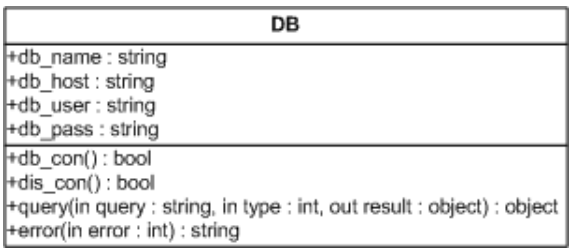


Figure 5.8: Class diagram of DB class

sb_unb.php

sb_unb.php files contain the main logic for capturing users navigational behaviour (UNB) data. It extracts following fields from the HTTP header:

- requested URL,
- type of request method (i.e. GET and POST method),
- IP address,
- User Agent,
- and referrer URL

sb_unb.php assigns a session ID and a timestamp for each HTTP request and stores them along with the above extracted field into the database. The session ID is handled through PHP's *session_start()* local function. This function creates or resumes a session based on an identifier in the cookie (Cosentino 2002).

Two parameters – *web page title* and *username* are passed to this class and are stored in the database. The former is the title of the web page and the latter is the username of the user visiting the website. Different web applications store usernames in different ways (e.g. HTTP cookie); hence, this information needs to be previously extracted and then passed to *sb_unb.php*. Figure 5.9 shows the class diagram for *sb_unb.php*.

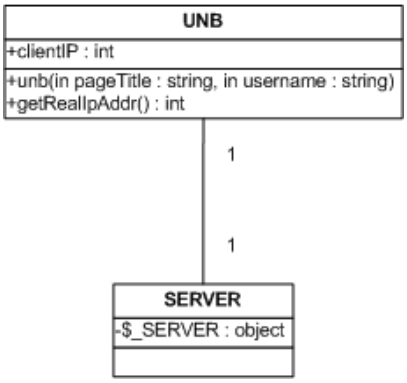


Figure 5.9: Class diagram for the NTC

As shown in the figure, the *unb* function obtains the web page title (*pageTitle*) and username as parameters which are then passed to *unb* each time a new request is sent to it. *unb* obtains HTTP header information from PHP's *Server and execution environment array* (*$_SERVER*). This array is created by the web server once a new connection is made (Cosentino 2002).

Depending on the web server configuration, the client IP address is stored in different fields in the HTTP header. The *getRealIpAddr()* verifies each HTTP field to extract a client's valid IP address. The IP address is later passed to the *unb* for storage.

The algorithm for the NTC is provided in Code 5.1.

Code 5.1: Pseudo-code for NTC

```
Algorithm navigation_usage_tracking
    input: webpage title
         user account name
         HTTP header array $_SERVER
    output: web navigation usage along with username, session ID
and webpage title
1.  get client IP address by examine HTTP_CLIENT_IP,
HTTP_X_FORWARDED_FOR, REMOTE_ADDR field in HTTP header.
2.  construct request URL from SERVER_NAME, SERVER_PORT,
REQUEST_URI
3.  call session_start();
4.  make new instance of DB class.
5.  insert session ID, timestamp, HTTP_REFERER, HTTP_USER_AGENT,
webpage title, request URL, client IP address, username into the
database.
```

5.6.1.2 Form Tracking Component (FTC)

FTC is implemented in *sb_ft.php*, *sb_ft.js*, and *sb_form_assign_[FORM NAME].js*. Figure 5.10 shows the class diagram for the FTC.

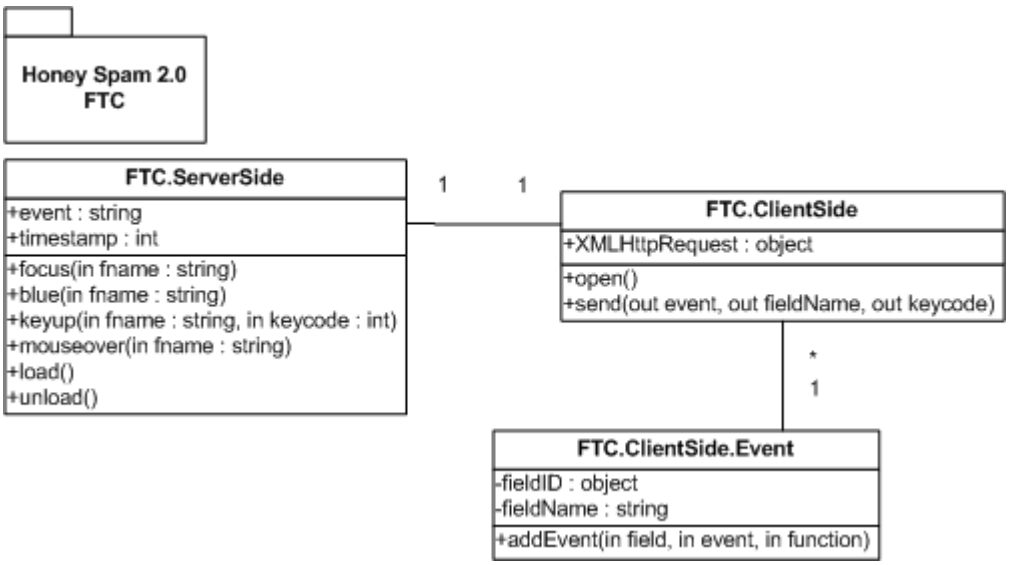


Figure 5.10: Class diagram for the FTC.

sb_ft.php

sb_ft.php is the server-side code for the form tracking component. It captures form event actions that are sent through the client-side code and stores them in the database using *sb_mysql.php*. The following form events' timestamps are captured by *sb_ft.php*:

- *Field focus*, when a field receives user focus. It can be caused by a user clicking on the field in the form.
- *Field blur*, when a user leaves field. It can be caused by a user selecting other fields in the form.
- *Field key up*, when a user releases a key on the keyboard. The key up time stamp along with key code is captured.
- *Field mouse movement*, when the mouse passes over a field.
- *Form load*, when a form is fully loaded on the user's Internet browser.
- *Form unload*, when a form is unloaded on the user's Internet browser.

Additionally, *sb_ft.php* stores the HTTP header data for each form event action including IP address, the User Agent, session ID, and username.

Figure 5.11 illustrates the class diagram for the FTC server-side code.

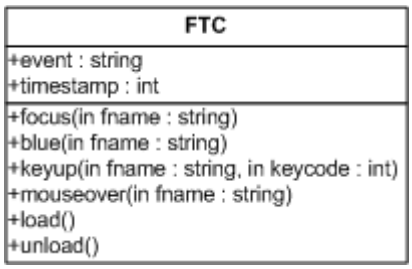


Figure 5.11: Class diagram for the FTC server-side code.

For simplicity, functions for storage of HTTP header data, extracting IP address, session ID and username are not included in the figure. The algorithm for the FTC server-side code is presented in Code 5.2.

Code 5.2: Pseudo-code for the FTC server-side

```
Algorithm form_tracking_server_side
    input: field name
           event name
           key code
           user account name
           HTTP header array $_SERVER
    output: form usage data along with HTTP header information.
1.  get field name, event name
2.  get HTTP header information from $_SERVER
3.  if (event  = keyup) { get key code name }
4.  switch (event)
      a.  case focus call focus()
      b.  case blur call blur()
      c.  case keyup call keyup()
      d.  case mouseover call mouseover()
      e.  case load call load()
      f.  case unload call unload()
5.  insert session ID, field timestamp, event name,HTTP_REFERER,
HTTP_USER_AGENT, webpage title, request URL, client IP address,
username into the database.
6.  if (event  = keyup) { insert key code into the database }
```

sb_ft.js

sb_ft.js is the client-side code for the FTC. This Javascript code handles the connection to the FTC server-side code i.e. *sb_ft.php* by using AJAX web development technique. In traditional web development, web pages need to be reloaded to process each HTTP request/response to/from the server. However, with the emergence of the AJAX web development technique, web pages can be updated in the background (Holdener III 2008).

sb_ft.js uses *XMLHttpRequest* (*XHR*) object to transmit data between the client and the server (Flanagan 2006). XHR is used to transmit form event actions to the FTC server-side code without reloading the web page. XHR is an API that is available in Javascript language.

Figure 5.12 shows the class diagram for the *sb_ft.js*. Two functions – *open()* and *send()* are required for AJAX connections.

Figure 5.12: Class diagram for the FTC client-code.

sb_form_assign_[FORM NAME].js

sb_form_assign_[FORM NAME].js is FTC client-code designed to assign an event handler to each form field and capture form events. Depending upon the number of forms and forms' fields, this file is created and customised. As shown in Table 5.2, different event handlers are assigned to different form fields.

Table 5.2: Event handlers for different form fields.

Form field	Event handler
Text field	Focus, blur, key up, mouse over
Checkbox	Focus, blur, mouse over
Button	Focus, blur, mouse over
Select list	Focus, blur, mouse over

sb_form_assign_[FORM NAME].js operates as follows.

1. Searches for given field ID or name
2. Assigns event handler to the form field
3. Captures events
4. Calls AJAX function declared in *sb_ft.js* to transmit the data to the FTC server-side code.

The class diagram for the FTC client-code event handler is shown in Figure 5.13.

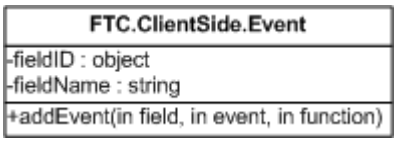


Figure 5.13: Class diagram for the FTC client-code event handler.

5.6.2 Pre-processing part

Since the data is captured in the web server, it is easier to utilise server-side codes to process the data. Hence, the pre-processing part is implemented using PHP. However, the computation is more complex and time consuming. PHP scripts can be executed and run on the server for a limited period of time. Scripts which exceed this limit are terminated by the web server. To overcome this problem, PHP command line scripts are used to process the data (Gutmans et al. 2004). PHP command line script does not have any execution time limit and it can be used to accomplish any time consuming tasks.

The pre-processing part of HoneySpam 2.0 is modelled to

- Cleanse the data
- Identify users and sessions

Pre-processing is performed when data is retrieved from the database or during data capture (i.e. tracking part). Hence, the pre-processing tasks are not coded in separate files. Data cleansing is performed when data is being fetched from the database. User and session identification are done while HoneySpam 2.0 is capturing the data.

5.6.2.1 Data cleansing

Data cleansing tasks is done through the construction of specific SQL database queries. Table 5.3 lists the conditions that need to be considered when data is fetched from the database.

Table 5.3: Data cleansing tasks in the pre-processing part of HoneySpam 2.0.

Condition	Description
Username NOT IN (‘’, ‘administrator’)	Not to include data from visitors and researchers who monitor the system.
UserAgent NOT IN (‘Googlebot’, ‘Slurp’, …)	Not to include data from known web robots (Google, Yahoo! etc.)
URL NOT LIKE (‘%.jpg’, ‘%.js’, ‘%.css’, …)	Not to include data from image, javascript, stylesheet etc. files requests.

5.6.2.2 User Identification

Regardless of the majority of web usage mining algorithms that use specific heuristics to identify users in the logs (R. Cooley et al. 1997), the proposed method uses a *cookie* to identify users.

The majority of Web 2.0 platforms require users to sign-up or register as a *member* in order to contribute to the website content. After user registration, users are required to *login* to the website for authentication. Once users are logged in to the website, their user credential information including username is stored as a cookie.

The cookie information is sent through HTTP headers to the web server. Usernames for each HTTP request to the web server can be extracted from the cookie data.

In the proposed method, cookie information is extracted and stored when the NTC stores usage information to the database. In the PHP language, cookie information can be retrieved from *$_COOKIE* array (Cosentino 2002). The proposed method uses *$_COOKIE* array to extract the username and use it as a parameter in other functions such as *unb*, *focus*, key up etc.

5.6.2.3 Session Identification

Unlike traditional web usage mining algorithms (R. Cooley et al. 1997), the proposed method uses session cookies to handle session identification tasks.

As mentioned earlier, for each HTTP request, the proposed method calls *session_start()* to create or update the current session. The session identifier is passed through the cookie along with the HTTP request.

One of the main challenges in session identification is *session expiry time* which is the number of seconds for which a session is valid for the client, after which time a new session ID would be assigned to the client. The session ID needs to be updated once it has expired. A common heuristic used in web usage mining is to set this time to 30 minutes (Spiliopoulou et al. 2003). In the proposed method, the session expiry time is set to 30 minutes. The following algorithm presents pseudo-code for the session identification task.

Code 5.3: Pseudo-code for session identification.

```
Algorithm session_timeout
    input: session creation timestamp, c
         session expiration time, T
    output: session identification
1.  if (session.c = null)
2.    session = session_start()
3.    session.c = current.time()
4.  if (current.time() - session.c > T)
5.    session = regenerate session id
6.    session.c = current.time()
```

In Code 5.3 function *current.time()* returns the current time stamp. For regeneration of the session id, PHP's *session_regenerate_id()* local function is used.

5.6.3 Knowledge discovery part

The knowledge discovery part is the last part of HoneySpam 2.0. It generates a general usage analysis reports and consists of PHP files for extracting data related to general usage analysis (GUA).

The data for usage analysis is extracted using *sb_gua.php* and *sb_mysql.php*. *sb_gus.php* is used to query database and fetch the data to measure the following:

- Total and average tracking records
- Content categories
- Origin countries
- Internet Browsers

Additionally, *sb_gua.php* extracts and stores data into CSV files format. The CSV files are used in MATLAB in order to draw figures as discussed in the next section. s*b_mysql.php* is used to handle the database connections.

sb_gua.php

This PHP script is used to query MySQL database, fetch the appropriate data to measure features. It is also used to store the data in CSV file format.

Total and average tracking reports

Table 5.4 lists SQL syntax statements to query and fetch that data from the MySQL database to measure the total and average tracking reports.

Table 5.4: SQL syntax statements to query database for general usage analysis.

SQL Syntax	Description
COUNT(*expression*)	To calculate total number and sum of whole records or specific columns in the tables.
SELECT DISTINCT *expression*	To return distinct (not duplicated) values in a table
GROUP BY *expression*	To group and return the data in one or more columns.
AVG(*expression*)	To return the average value of a column in a numeric format.

Content Categories

To extract content categories or themes for the submitted content, word frequency for the submitted contents is calculated. Next, by manual investigation of the top 100 most frequent words, the category for the submitted content is measured. The following assumptions are made when measuring the word frequency:

- Words are separated using one space.
- Words are made up of alphanumeric characters.
- Words might contain a hyphen or apostrophe.

- HTML tags, words starting with apostrophe or hyphen, and words made solely from numbers (e.g. 123ab) are not counted.

- ‘the’, ‘a’, ‘of’, ‘is’, ‘to’, ‘for’, ‘by’, ‘be’, ‘and’, ‘are’ etc. are not counted.

The algorithm to measure word frequencies is presented in Code 5.4.

Code 5.4: Pseudo-code for the word frequency algorithm.

```
Algorithm word_frequency
    input: set of submitted contents by users, C
           set of excluded words, C'
    output: a sorted list of word frequencies, L
1.  for each ci in C
2.    if ci in C'
3.      go to the next ci
4.    end if
5.    if ci in L
6.      ci.count = ci.count + 1
7.      update ci.count in L
8.    else
9.      ci.count = 1
10.     add ci & ci.count to L
11.   end if
12.   sort L descending ci.count
13.  repeat
14. return L
```

Origin Countries

A well-known protocol, *WhoIS*, is used to find out the country of origin of the incoming request. WhoIS is a standard protocol to query information about domains, hosts, networks and IP addresses from a remote WhoIS database known as *Regional Internet Registries* (RIR) (Daigle 2004). RIR can be queried using domain/host name or an IP address to obtain information about network, *nameservers*, registration date, demographic and contact information of the internet resources (Harrenstien 1982).

American Registry for Internet Numbers (ARIN) and *Asia-Pacific Network Information Centre* (APNIC) are used as RIR to lookup the network information (Hunt 2002). IP addresses for each incoming request are queried using *ARIN’s Whois RESTful Web Service* (Whois-RWS) to discover the country of origin. Figure 5.14 presents WhoIS information for the Curtin University network using APNIC. It is clear from the figure that the network belongs to Australia.

```
% [whois.apnic.net node-3]
% Whois data copyright terms
http://www.apnic.net/db/dbcopyright.html
inetnum:        134.7.0.0 - 134.7.255.255
netname:        CUT-NET
descr:          Curtin University of Technology
country:        AU
...
person:         Ian Hill
nic-hdl:        IH156-AP
e-mail:         CITS-Contract-renewals@curtin.edu.au
address:        Curtin University of Technology
address:        Curtin IT Services
address:        Building 309 Kent Street
address:        Bentley 6102
address:        Western Australia
```

Figure 5.14: WhoIS information for the Curtin University network extracted from APNIC.

Internet Browsers

The User Agent header in the HTTP request contains information about the client's Internet browser, its version, and details about the clients system (Berners-Lee 1945). This information is used to identify the client's software while navigating through the website. Figure 5.15 illustrates a sample User Agent header. From the figure it can be inferred that:

- The Internet browser's name is *Internet Explorer* (*Microsoft Internet Explorer* identify itself as Mozila/4.0).
- Compatibility flag indicates that this browser is compatible with a common set of features.
- The Version token defines the browser and its version (*Internet Explorer 7*).
- Platform token identifies the operating system and its version (*Windows Vista*).

Figure 5.15: A sample User Agent header.

Web crawlers also identify themselves using the User agent field.

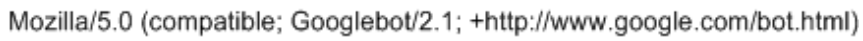


Figure 5.16: Google web crawler User Agent header.

Figure 5.16 shows the user agent header for a Google robot. It claims to be a *Mozilla* based user agent version 5. It has a compatible flag. The name is Googlebot and its version is 2.1. Its URL address for more information is *www.google.com/bot.html*.

The User Agent header discovers how frequently spam users mimic genuine user identities through this header.

CSV file

sb_gua.php also extracts the required data from the MySQL database and stores it in CSV file format. The CSV files are used in MATLAB to generate the following figures and graphs.

- Frequency of posts per day
- Frequency of user registrations per day
- Frequency of online users per day
- Frequency of visits on each web page
- Frequency of users spending time on each session
- Frequency of revisits

MATLAB is used to produce each figure. Additionally, some enhancement is done using *Microsoft Excel 2007* on figures (Jacobs et al. 2008).

This concludes stage four of the conceptual process. The next section focuses on experimental settings where the proposed prototype is tested.

5.7 Experimental Setting

The experimental settings are the *fifth* stage of the conceptual process where the proposed prototype is tested and examined. Deploying HoneySpam 2.0 involves 3 steps including Deployment, Customisation and Promotion.

The deployment step presents a detailed description of the software and hardware infrastructure of the HoneySpam 2.0 server. The customisation step explains different configuration settings for each copy of HoneySpam 2.0. The last step, Promotion, presents the strategy for attracting real-world samples of Spam 2.0.

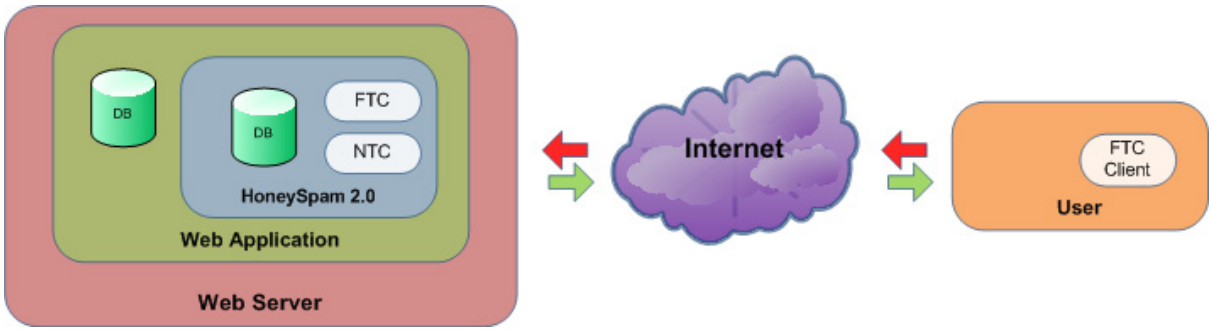


Figure 5.17: Entire implementation of HoneySpam 2.0 for 6 commercial and 5 free hostings.

Figure 5.17 illustrates the entire implementation of HoneySpam 2.0. HoneySpam 2.0 is embedded inside the web application and stores tracking data in its own database. The web application runs inside a web server and has its own database. The web server is accessible publicly through the Internet. Finally, HoneySpam 2.0 client-side code is executed on the user's machine.

5.7.1 Deployment

The HoneySpam 2.0 was deployed in 6 commercial servers named *HSCOM1 – HSCOM6* and 5 free web hosting servers named *HSFREE1 – HSFREE5*. The software and hardware specifications of each web server are presented in Table 5.5.

Table 5.5: Deployment specification of each host.

Name	OS	Location	CPU	Software	Database	Uptime
HSCOM1	Linux (SUSE) 64bit	Australia	Intel® Xeon® 2.66 GHz	Apache 2.2.10 PHP 5.2.6	MySQL 5.0.67	25/5/09 – 9/8/09 * 27/7/10 – 16/11/10
HSCOM2 – HSCOM5	Linux (SUSE) 64bit	Australia	Intel® Xeon® 2.27 GHz	Apache 2.2.10 PHP 5.2.6	MySQL 5.0.67	5/1/10 – 16/2/11
HSCOM6	Linux (SUSE) 64bit	Australia	Intel® Xeon® 2.27 GHz	Apache 2.2.10 PHP 5.2.6	MySQL 5.0.67	21/5/10 – 17/2/11
HSFREE1 – HSFREE5	Linux	USA	AMD Opteron 1.9 GHz	Apache 2.2 PHP 5	MySQL 5	25/5/09 – 9/8/09 *

* Only 60 days of data was collected due to server downtime periods caused by maintenance issues.

All the commercial hosts run under a Virtual host environment to meet the security requirements of HoneySpam 2.0 (ref. Section 5.3). The detailed deployment description of each host is as follows.

5.7.1.1 HSCOM1

HoneySpam 2.0 is integrated with two web applications – an online discussion board and social networking platform. The online discussion board is implemented using common *Open Source* forum, *Simple Machine Forum* (SMF) (SMF Team 2011). The social bookmarking system is implemented using *Pligg*, a common social bookmarking Web 2.0 platform (Heikkinen 2011).

A copy of HoneySpam 2.0 for each web application, two separate directories and MySQL databases, have been created. Additionally, HSCOM1 has an Internet connection provider that is different from those of other commercial hosts.

HSCOM1 operated for two periods (HSCOM period 1 and period 2). The dates for each period are shown in Table 5.6.

5.7.1.2 HSCOM2-HSCOM5

HoneySpam 2.0 is integrated with 5 common web applications including – wiki, blog, social networking, online discussion board, commenting platforms as well as a simple contact form. A popular *MediaWiki* web application is used as wiki (Mediawiki 2011). MediaWiki is a free wiki web

application used in Wikipedia. *Wordpress Multi-User (Wordpress Mu)* a blog and publishing platform to create multiple blogs is used as a blog and commenting platforms (Wordpress 2011). SMF and *phpBB* is used for an online discussion board (phpBB 2011). Finally, Pligg is used for social bookmarking web application.

5.7.1.3 HSCOM6

HoneySpam 2.0 is integrated with phpBB, a popular open source online discussion platform. HSCOM6 is run and configured separately from other commercial hosts (ref. Section 5.7.2.3).

5.7.1.4 HSFREE1-HSFREE5

Pligg and SMF are used as a social bookmarking and online discussion board platform on all free web hostings.

It is worth mentioning that the client-side code is executed on the user's Internet browser to accumulate real-world data for FTC.

The following section discusses each host configuration.

5.7.2 Customisation

The customising step explains different configuration settings for each copy of HoneySpam 2.0. The aim of the different customisation settings is to address the following:

1. **Dedicated vs. Shared IP address**: Dedicated IP address assigns one IP address to one particular website, while shared IP address is used to share one IP address for multiple websites. This choice is made in order to investigate whether or not spammers try to discover their targets by examining IP addresses in the same range. For example, by examining all IP addresses that belong to the same *sub network*, other active hosts are identifiable.

2. **Empty vs. Not-empty website**: Web applications can be launched either empty (with default content in standard installation) or it can be pre-populated with some content. A non-empty website is filled up with relevant content that is accumulated from other similar websites. This choice is to investigate whether or not spammers target websites with the content or empty websites. Empty websites do not have good rankings in the search engines; hence, they might

not be accessible in top search engine results. This choice can also investigate whether or not spammers find targets through search engine results.

3. **Live vs. Not-active website**: Live websites actively become populated by active users. However, non-active websites are not actively being used. This choice can investigate whether or not spammers try to target active or not active websites.

A detailed configuration description of each host setting is presented next.

5.7.2.1 HSCOM1

HSCOM1 is hosted on a shared IP address. This IP address is used by other websites that have hosted on HSCOM1's web server. A well ranked domain name that has been previously used for other purposes is mapped to this web server.

No content has been added to Pligg, and SMF running on this host and web applications default configuration settings are used. The registration spam filtering methods (e.g. CAPTCHA) have been disabled on both web applications.

5.7.2.2 HSCOM2 – HSCOM5

HSCOM2 – HSCOM5 are hosted on dedicated IP addresses but in the same sub-network. IP addresses have not been used previously and are newly registered. Four new domain names are registered and assigned to each IP address.

The choice of domain names is based on popular searched keywords. A list of popular keywords was extracted from common search engines, and then by using of online domain-suggestion tools such as *www.domaintools.com*, domain names are generated. The choice of domain names can make websites more visible in the search engine results (Perkins 2001).

No content has been added to any web applications running on the hosts. The registration spam filtering methods (e.g. CAPTCHA) were disabled on all the web applications.

5.7.2.3 HSCOM6

HSCOM6 uses an IP address shared with HSCOM5.The hosted website was regularly filled with relevant content. Ten users were actively contributing to the content generation of this live website. The registration spam filters were enabled on this host.

5.7.2.4 HSFREE1 – HSFREE5

All the free hostings share their IP addresses. Five sub domain names were assigned to each host. Sub-domains consist of the domain name of the host provider, such as *http://name.FREE-HOSTPROVIDENAME.com*. Five free redirection domains are assigned to each free host. The choice of the redirection domains was based on popular searched keywords.

The following section explains the different strategies that are used to advertise each host.

5.7.3 Promotion

Four different strategies have been used to attract spam users as follows.

5.7.3.1 Online Advertisement

Google AdWords service was used to advertise HSCOM2-HSCOM5. Google AdWords provides keyword-based advertisement services (Geddes 2010). For example, when people search for keywords in the Google search engine, if the keyword is relevant to the chosen keyword for a particular website, the advertisement appears next to the search result.

5.7.3.2 Linking from other websites

Hyperlinks were created from other websites to all hosts. This strategy was used to promote all the hosts including both commercial and free.

5.7.3.3 Listing in the public directory service

The addresses for free hosts were added to the public directory services such as dmoz.org (dmoz 2011). The public directory service provides a list of website links based on their categories. This strategy is used to promote HSFREE1-HSFREE5. Their link was removed after the experiment to ensure the quality of the directory.

5.7.3.4 Traffic redirection

By employing the server-side redirection technique (Smith 2002), the traffic to HSCOM1 was redirected to HSCOM2 for a period of 7 months.

During implementation of the above strategies, one of the important challenges was to ensure that genuine human users did not interact with any of the hosts. To achieve this, the following approaches were adopted.

- Websites were kept empty or were filled up with non-useful content. This was done to ensure that genuine human users did not contribute to the website content. HSCOM1-HSCOM5 as well as HSFREE1-HSFREE5 were purposely kept empty.
- All the hosts were also manually moderated to ensure that no human users participated in content or web usage generation.

HSCOM6 is the only host that contains both human and spam users data. Initially, it was launched only as a forum for human content contribution; however, later spam users also contributed to the website. Human and spam users were manually identified.

This concludes stage five of the conceptual process. The next section focuses on the results and observations based on the proposed prototype experiments.

5.8 Results and Observations

This is the *sixth* stage of the conceptual process. This section discusses the results that were observed after running the proposed prototype i.e. HoneySpam 2.0.

All the hosts were running for specific periods as shown in Table 5.5. The results and observations are for general statistical analysis purposes.

The summary of the data that was tracked by HoneySpam 2.0 on each host is provided in Table 5.6.

Table 5.6: Summary of tracking data for HSCOM1 for two periods.

Summary of data for HSCOM1	HSCOM1 (1)	HSCOM1 (2)	HSCOM6
No. of all records	11,481	2,756,825	322,207
No. of records after data cleansing	6,794	1,867,767	423,36
Total No. of spam users	643	16,930	171
Total No. of unique used IP addresses	314	4,234	191

HoneySpam 2.0 running on HSCOM2-HSCOM5 and HSFREE1-HSFREE6 did not attracted any spam which shows that there were no Spam 2.0 activities on these hosts. The reason behind this is discussed in the next Section 5.9. All the results and observations are based on HSCOM1 period1, HSCOM1 period2, and HSCOM6 that are discussed in this section.

5.8.1 General statistical results

Figure 5.18a shows the frequency of spam users' visits to each web page on HSCOM1. It is clear from the figure that spammers mainly visited one web page and there are no more than 300 visits.

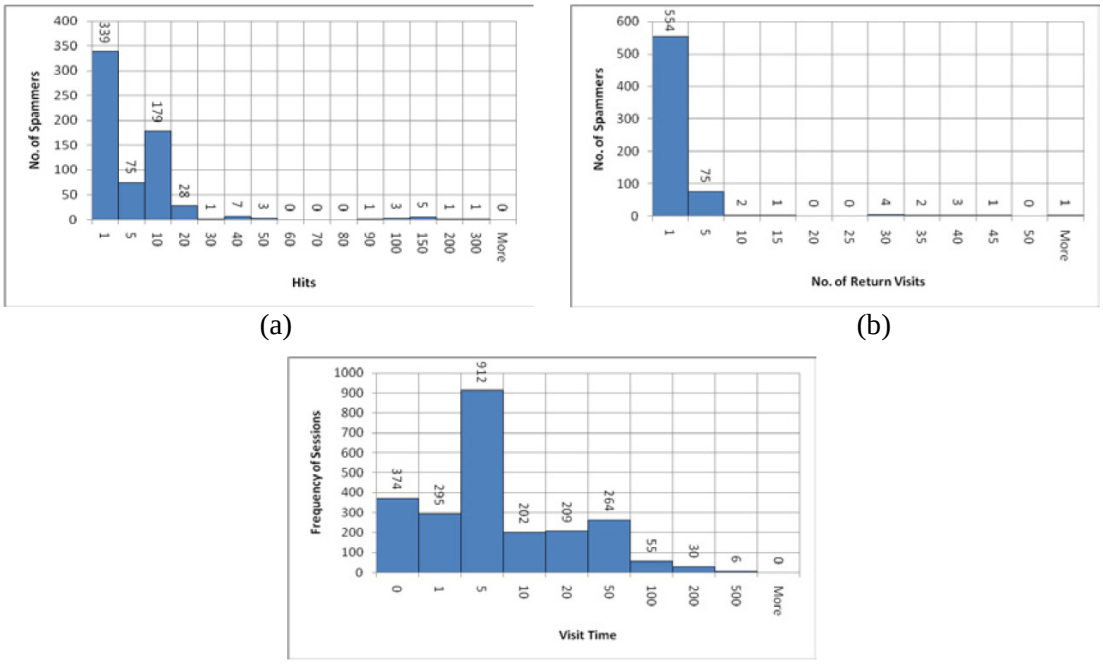


Figure 5.18: (a) Frequency of spammers visit on each web page in HSCOM1. (b) Frequency of spammers' return visits on HSCOM1. (c) Frequency of spammers visits time (sec) per session on HSCOM1.

The frequency of spammers' return visits is presented in Figure 5.18b. The majority of spammers visited HSCOM1 only once; however, there are spam users that have more than 5 return visits.

The amounts of time spammers spent on the HSCOM1 each time they visited the website is illustrated in Figure 5.18c. It is clear from the figure that the majority of the spammers spent between 1 to 5 seconds in each session. However, some did not spend time on the website which means that they sent just one request to the web server (ref. Section 5.9).

Figure 5.19a shows the frequency of spammer visits to each HSCOM1 web page. It is clear from the figure that the majority of spammers visit between 5 and 20 web pages. Some spammers visit only one web page. However, some have more than 1000 web page visits.

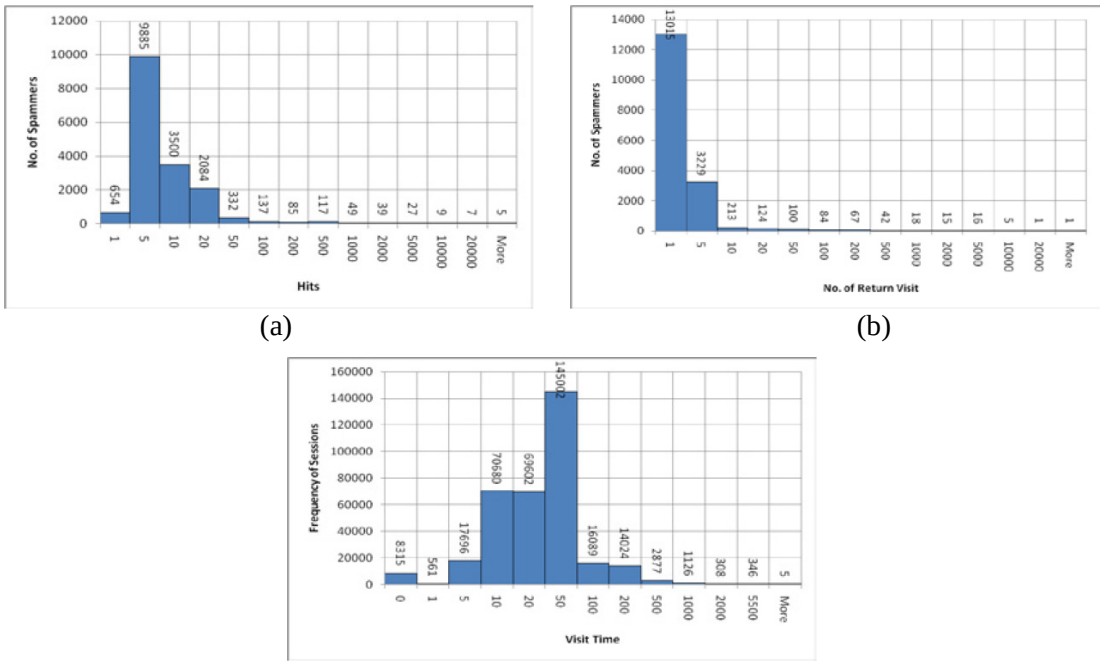


Figure 5.19: (a) Frequency of spammers' visit to each web page during HSCOM1 second period. (b) Frequency of return visits by spammers during HSCOM1 second period. (c) Frequency of spammers' visits time per session during HSCOM1 second period.

Figure 5.19b illustrates the number of spammers' return visits to HSCOM1. It is clear from the figure that most of the spammers re-visited only once followed by fewer than 5 re-visits to the host (1 means only one visit). The frequency of the spammers visit times per session for the HSCOM1 second period is presented in Figure 5.19c. Spammers mainly spend 10 to 50 seconds with the host. The rest spend 100 to 200 seconds and 1 to 5 seconds with the host. Some spammers send only one request to the host; hence, their visit time is zero.

Figure 5.20a presents the frequency of spammers' visits to each web page in HSCOM6. It is clear from the figure that the majority of spammers' visits range from 5 to 50.

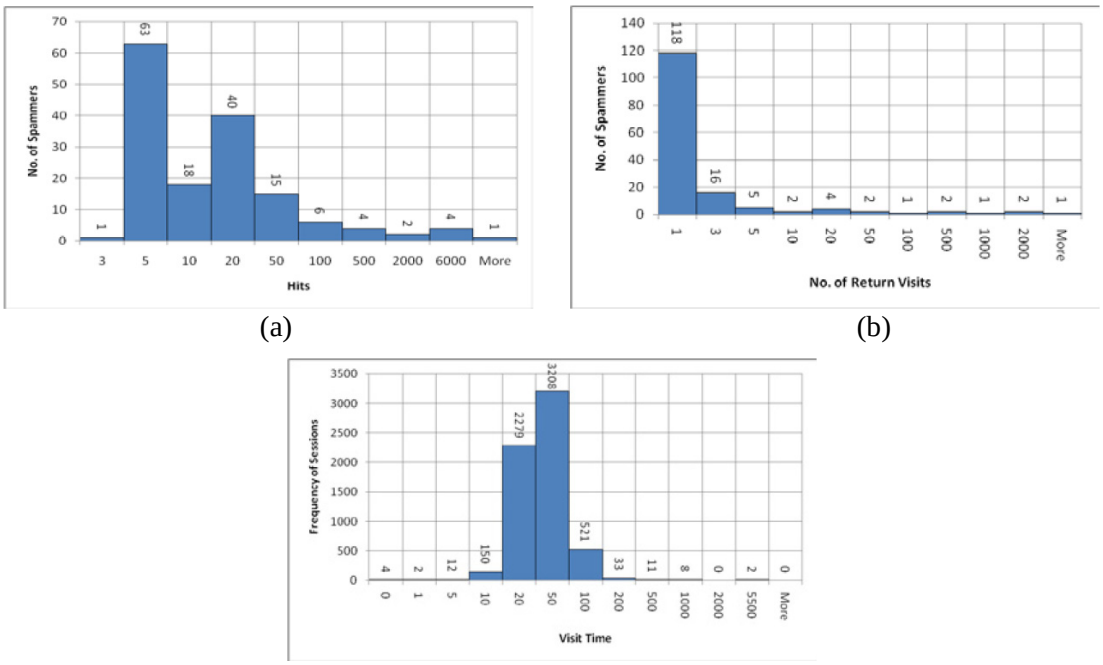


Figure 5.20: (a) Frequency of spammers' visits to each web page for HSCOM6. (b) Frequency of return visits by spammers for HSCOM6. (c) Frequency of spammers' visits time per session for HSCOM6.

Figure 5.20b shows the frequency of spammers' return visits to HSCOM6. As the figure shows, the majority of spammers visit the host only once. Figure 5.20c presents the frequency of spammers' visit times per session in HSCOM6. As is clear from the figure, spammers spend mainly between 20 to 50 seconds with the host. There are very few spammers that spend less than 1 second with the host.

5.8.2 Content categories results

Figure 5.21 illustrates the percentage of the top content categories for the HSCOM1 host. Adult-related keywords are the most dominant content. The drug category promotes regular and adult-related *medicine*. The movie category includes topics about *watching latest movies*, *YouTube*, adult-related cartoons and animations. In the banking category, the main content theme is *low service fee credit cards*, *making money online* etc. The mobile phone category is about ringtones and the latest mobile devices (e.g. *iPhone*). Other categories contain various keywords such as *driving test*, *gym*, *hair loss* etc.

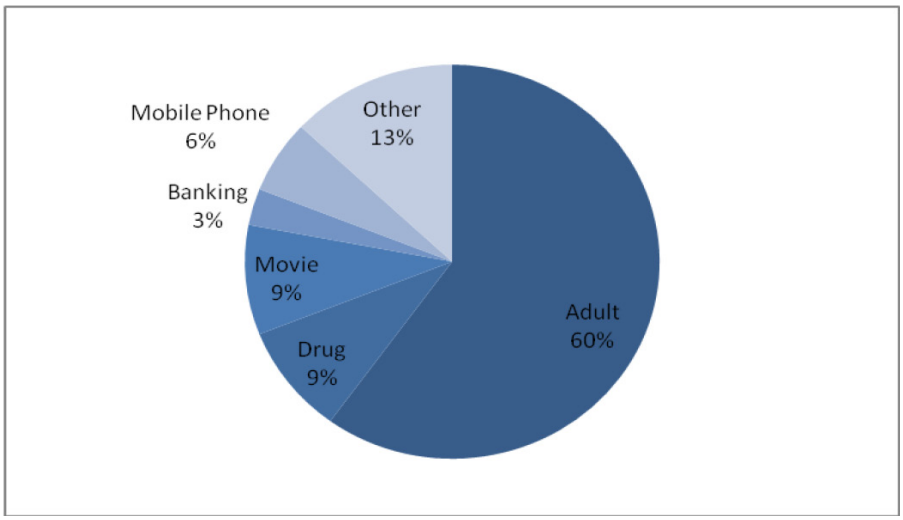


Figure 5.21: Top content categories on HSCOM1

Figure 5.22 presents the percentages of the top content categories for the HSCOM1 host. Adult-related keywords are the most dominant content captured by HSCOM1. The movie category also contains adult-related keywords; therefore, it can be considered as adult content. Free categories include topics about *downloading free software*, *free gifts*, *free postcards* etc. The dating category mainly promotes *online dating websites*. The ringtone category includes topics about the *latest ringtones for mobile phones*, *latest songs as ringtone*. The games and music categories are about *computer games* and the *latest top hit songs* respectively.

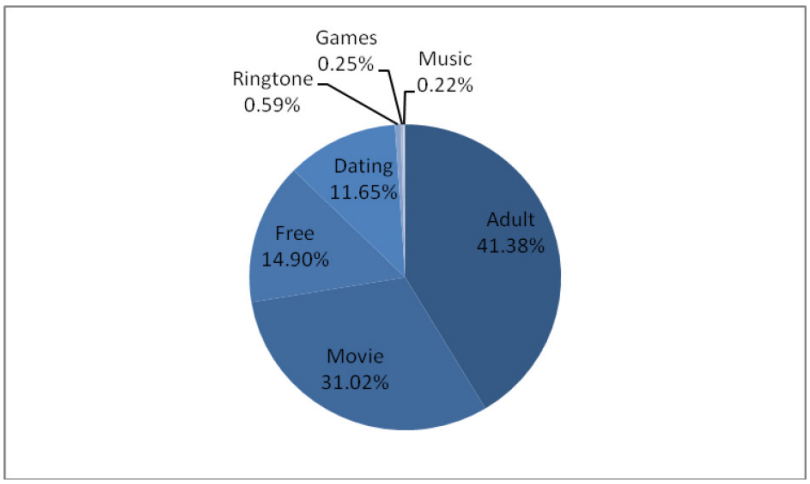


Figure 5.22: Top content categories for HSCOM1 in the second period of running

As illustrated in Figure 5.23, the top content category on HSCOM6 includes only spammers' content contribution. Rather than the adult category, shopping-related topics are common among spam topics. This category contains topics on *buying bags*, *shoes*, *boots* etc. The swear category includes topics

around an offensive word in the sentence. The free category is on *downloading free stuff online.* Finally, the movie category is on *online watching movies* and *online flash movies.*

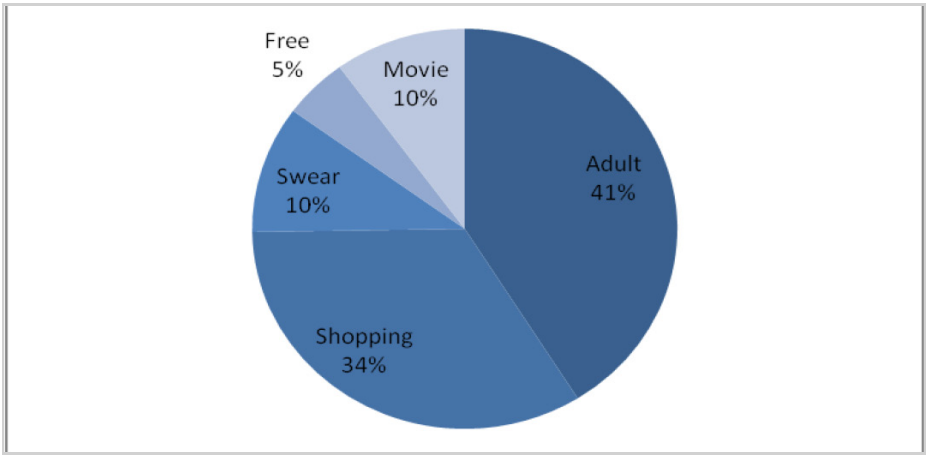


Figure 5.23: Top content categories on HSCOM6 (only spam topics)

5.8.3 Origin countries results

Figure 5.24 presents the top demographic location of spammers' origins on HSCOM1. Demographically, 51% of observed spammers originated from the United Kingdom followed by the United States of America at 22%.

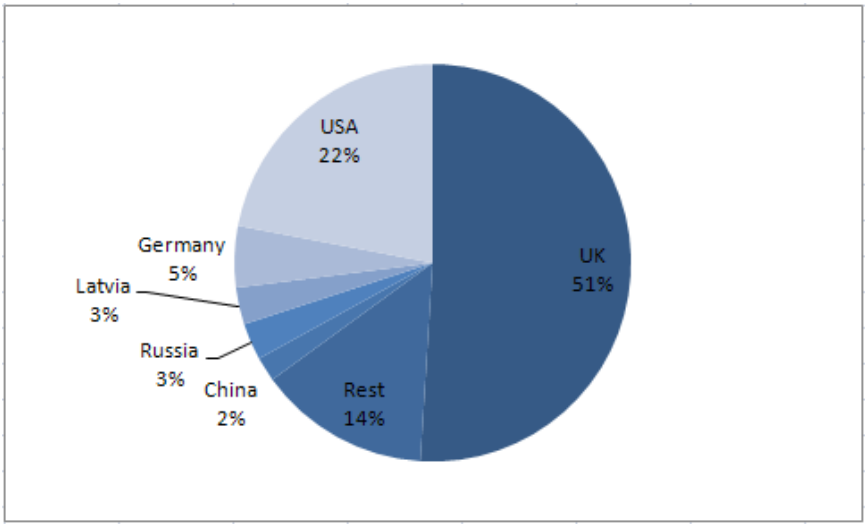


Figure 5.24: Top demographic location of spammers on HSCOM1

Figure 5.25 presents the top demographic location of spammers on HSCOM1 in the second period of running. Spammers originated mainly from Germany with 67.92%. The remaining 32.08% is divided amongst 8 countries including: Luxembourg 8.54%, Netherlands 7.79%, Russia 6.79%, Ukraine

3.23%, USA 3.22%, Latvia 1.85%, UK 0.42% and France 0.22% which is significantly lower than Germany.

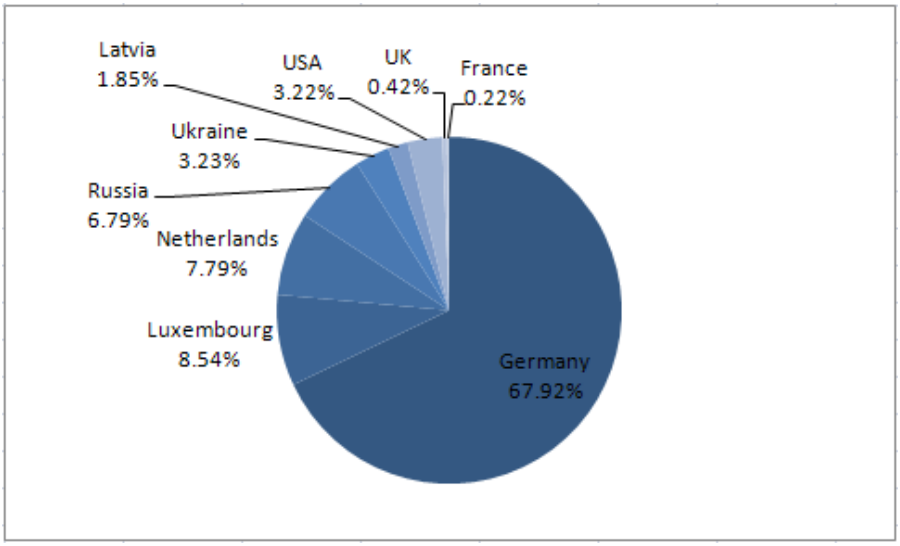


Figure 5.25: Top demographic locations of spammers on HSCOM1, period 2.

Figure 5.26 shows the top demographic location of spammers on HSCOM6. UK has the highest percentage with 56.17% followed by Ukraine 20.50% and Germany 16.27%. The remaining countries are as follows: Netherlands 3.77%, Russia 3.02%, USA 0.15%, Latvia 0.04%, China 0.03%, Poland 0.03%, and Sweden 0.01%.

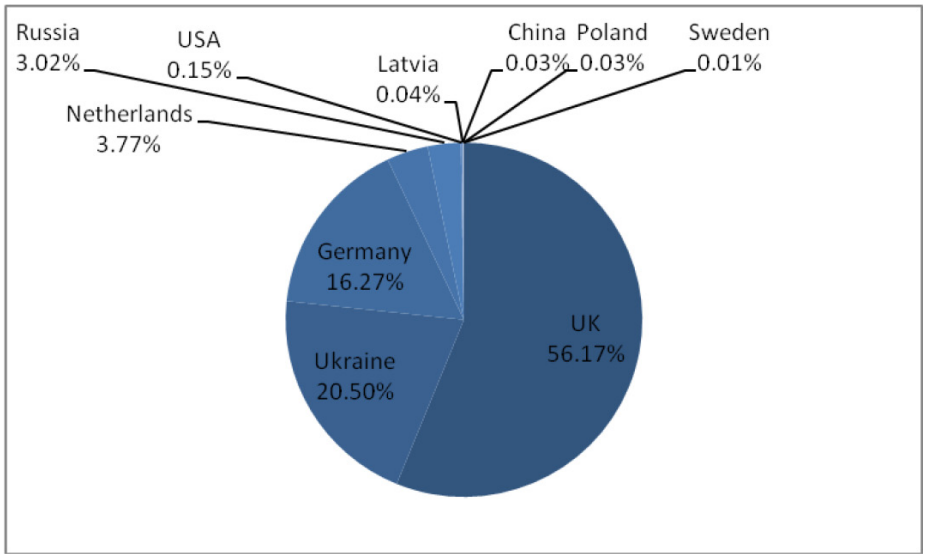


Figure 5.26: Top demographic location of spammers on HSCOM6

5.8.4 Internet browsers results

Figure 5.27 illustrates the top user agent values on HSCOM1. Internet browsers commonly used by spammers are Internet Explorer 71% (including all versions) and Opera 24%. Surprisingly, one relatively unknown browser OffByOne is also used. Additionally, Firefox was used by only 2 web spambots and Safari was not used at all.

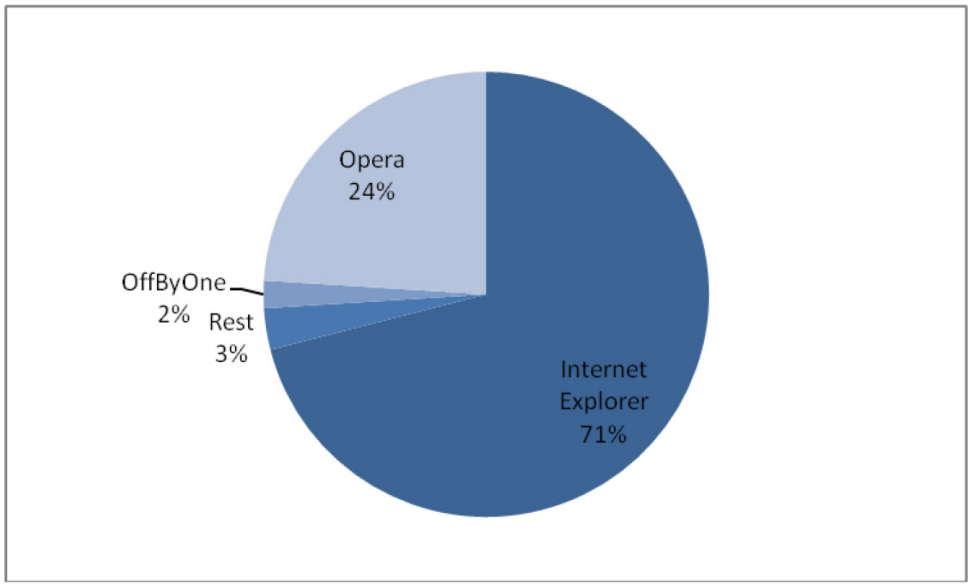


Figure 5.27: Top user agent fields for HSCOM1

Figure 5.28 presents top Internet browsers that were used to send requests on HSCOM1, period 2. Internet Explorer dominates with over 75.56%, Opera at 19.03%, AOL 2.69% and an unknown browser OffByOne 2.72%.

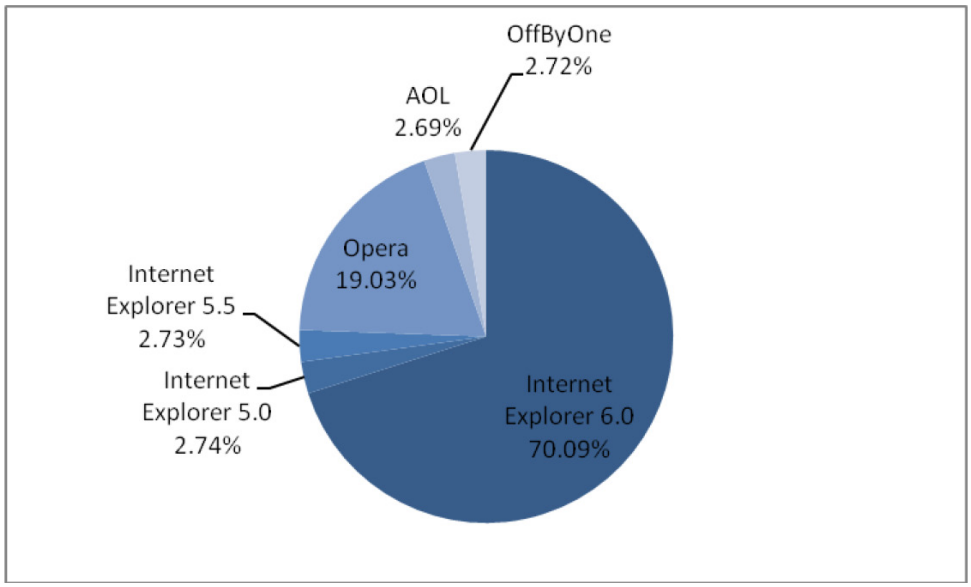


Figure 5.28: Top user agent fields for HSCOM1 second period

Figure 5.29 shows the usage percentage of Internet browsers on HSCOM6. Internet Explorer 6.0 is used in over 70% of requests followed by Internet Explorer 5.5 8%, Internet Explorer 5.0 3%, Opera 14% and AOL 5%.

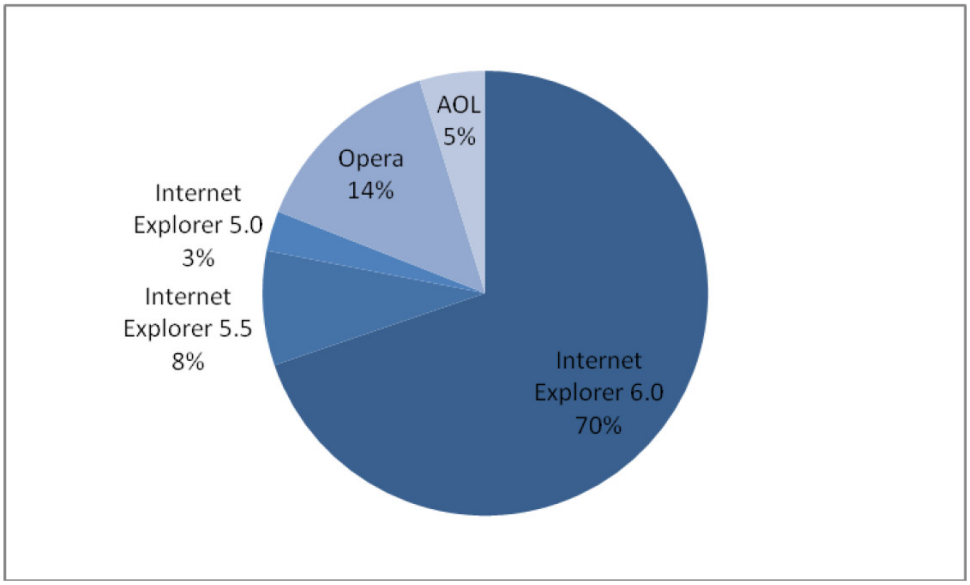


Figure 5.29: Top user agents fields for HSCOM6

This section examined the results of experimentation using the proposed prototype. This concludes stage six of the conceptual process and paves the way for the next stage. A key discussion and observations are presented in the following section.

5.9 Validation and Comparative Analysis

The following observations have been made during general statistical analyses.

Using search engines to find target websites

HSCOM1 (for two periods) and HSCOM6 received the most attention from the spammers. This highlights the possibility that spammers are using search engine intelligence (e.g. Google PageRank scores) to identify and focus on higher ranked websites that potentially yield more visitor traffic. The second period of HSCOM1 proportionally attracted more spammers than did the first period. The reason behind this could be related to search engine indexing, which also suggests that some spammers find their targets through search engine results.

Creating numerous user accounts

It was identified from the experiment that spammers would create a number of new accounts to post their content rather than using their existing accounts. This is shown in Table 5.6, where the total number of user accounts is much higher than the number of unique IP addresses used between these accounts.

It is possible that spammers adopt this behaviour for additional robustness in delaying account banning (e.g. one user account that is associated with numerous spam content items is easier to detect and manage) and/or to handle the event from which their previous accounts have been banned.

Low hits and revisit rates

The observed spammers adopt surfing patterns similar to one another and commonly visit a number of specific web pages. As illustrated in Figure 5.18a-20a, most spammers perform less than 20 page hits in total during their account lifetime.

Figure 5.18b-20b supports the claim that spammers do not revisit web pages many times. It is likely that these spammers revert to creating new user accounts to continue their spamming activities.

Distributing spam content in a short period of time

The amount of time most spammers spend on each visit (session) is less than 5 seconds (Figure 5.18c-20c). This indicates that spammers are distributing spam content within a short period of time, possibly to move on to other websites to distribute more spam content.

However, this behaviour changed during the second operational period HSCOM1 and HSCOM6 where the spammers stayed longer (10 to 50 seconds) on the website. The reason for this is the *Flood Control* protection mechanism in the forum which forced spammers to wait at least 30 seconds before posting new content with the same username.

No web form interaction

While not shown in the figures, the form-tracking component of HoneySpam 2.0 was unable to track form usage behaviour of spammers' tools. These tools adopt a conventional method of directly posting through their own forms / interface to submit content. They have not been designed to be intelligent enough to mimic human behaviour when submitting a web form.

Only two hosts i.e. HSCOM1 and HSCOM6 received Spam 2.0 traffic. The rest of the hosts did not receive any spam traffic. The reasons for this are as follows.

- HSCOM2 – HSCOM5 and HSFREE1 – HSFREE5 did not contain any content. Hence, they were not effectively indexed by the search engine. As the results of other hosts show, spammers use search engines to find and target websites. Since these hosts were not indexed highly by search engines, they did not receive Spam 2.0 traffic.

- HSFREE1 – HSFREE5 were hosted on free hosting providers. Usually, free hosts have lower ranks than commercial hosts in search engine results. Hence, they might not be listed highly in the search engine results.

- HSCOM1 domain name was already used for other purposes. Hence, it has already been indexed by search engines when the domain name was used to track spammers.

- HSCOM6 was an operational and live online discussion forum. Once it was indexed by search engines, HoneySpam 2.0 started receiving Spam 2.0 traffic.

However, more in-depth analysis needs to be undertaken to extract other characteristics of spam users from accumulated data. This is the focus of the next chapter on *Profiling Spam 2.0* behaviour using clustering tools. The next section provides a comparative analysis of HoneySpam 2.0 with existing solutions.

5.9.1 Comparative analysis

In this section, the proposed method is compared with the existing solutions from the literature. As mentioned previously, no studies have been conducted on Spam 2.0 and its characteristic. HoneySpam 2.0 is intended to track and characterise Spam 2.0. However, there are several similar works in the literature which study spam in other domains including email and social networks. These methods will now be compared with the proposed method.

There are three approaches for monitoring spam which include mining web server logs, HoneySpam, and Social Honeypots. A comparison study is depicted in Table 5.7 which shows the proposed HoneySpam 2.0 method compared with existing approaches.

Table 5.7: Comparative study of tracking approaches

	Mining web server logs (Cooley et al. 1999)	HoneySpam (Andreolini et al. 2005)	Social Honeypots (Webb et al. 2008)	Social Honeypots (Lee et al. 2010)	HoneySpam 2.0
Spam 2.0 user identification	No	No	No	No	Yes
Spam 2.0 session identification	No	No	No	No	Yes
Strategy	Mining HTTP access log files	Deployment of fake web server, open relay, open proxy and SMTP server	Creation of profile in social network website by using the automated bots	Creation of profile in social network website by suing the automated bots	tracking navigational and form usage behaviour on web apps
Form usage tracking	No	No	No	No	Yes
Domain	General	Email	Social networking websites	Social networking website	Web 2.0 platforms

5.9.1.1 Mining web server logs

Web server logs are created and updated by the web server whenever a client requests information from a web server by using Hypertext Transfer Protocol (HTTP) (Robert Cooley et al. 1999). These *access* log files contain information about the client IP address, date and time of request, request web page, HTTP code, referrer web page, and the user agent. Figure 5.30 presents an example of 4 HTTP access log entries.

Entire No.	Detail
1	111.111.111.111 - - [15/Aug/2009:07:05:10 +0800] "GET /forum/index.php HTTP/1.0" 200 60 "http://example.com/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
2	111.111.111.111 - - [15/Aug/2009:07:06:12 +0800] "GET /forum/index.php?board=1 HTTP/1.0" 200 6248 "http://example.com/forum/index.php" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
3	111.111.111.111 - - [15/Aug/2009:07:08:32 +0800] "GET /forum/index.php?action=post;board=1 HTTP/1.0" 200 7852 "http://example.com/forum/index.php?board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
4	111.111.111.111 - - [15/Aug/2009:07:10:27 +0800] "GET /forum/index.php?topic=1397 HTTP/1.0" 200 524 "http://example.com/forum/index.php? action=post;board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

Figure 5.30: An example of 4 HTTP access log entries

Such web server logs can be analysed through the *Web Usage Mining* technique to produce inputs for restructuring website designs (Robert Cooley et al. 1999). Web usage mining can be used to aid website navigation, provide similar web page recommendation, and place web pages for easier access. For example, web usage mining can be utilised to extract the following associate rules from a sample pharmaceutical website as presented in Table 5.8.

Table 5.8: An example of usage of web usage mining technique

Rule No.	Detail
1	47% of the visitors who access a web page about Hair Loss also accessed a web page on Acne.
2	52.1% of the visitors who accessed web pages about Chest Pain and Blood Pressure also accessed a web page on Breast Cancer.

However, web usage mining techniques cannot provide effective insights into Spam 2.0 content for the reasons given below.

1. Identifying spam usernames and tracking their navigation usage is critical in characterising Spam 2.0 behaviour. As is clear from Figure 5.30, usernames are not identifiable from raw web server logs. Additionally, user identification heuristics in web usage mining that are

mentioned in Section 5.5.2.2 are not effective enough to differentiate users' exact behaviour on the Web 2.0 applications.

2. Session identification heuristics (e.g. 30-minute timeout) in web usage mining as mentioned in Section 5.5.2.3 cannot effectively identify spam users' sessions. Figure 5.18a-20a shows that the majority of spam user sessions last for less than 50 seconds. Additionally, the spam users User Agent and IP address did not change over one session.

3. Web server logs do not provide information about form usage data. This usage data currently can be tracked through other techniques such as client-side scripting languages.

The above problems have been addressed by HoneySpam 2.0 which can effectively identify spam usernames and session. AJAX technology used inside HoneySpam 2.0 can track form usage data.

5.9.1.2 Social honeypots

Regarding Spam 2.0 detection, (Webb et al. 2008) proposed a social honeypot for detection of spammers in social networking websites. They hosted 51 social honeypots in a social networking website. The result of their work shows that temporal and geographic patterns of social spam have unique characteristics. The extension of this work has been proposed in (Lee et al. 2010) .

Social honeypots are capable of monitoring spammers' behaviour and log their data such as

- Profile information such as about me, age, relationship status, no. of friends etc,
- Background image which links to another website,
- Unsolicited friend requests in the social networks,
- Personal messages and comments in the social networks,
- Other textual information inside profile.

The overall framework for a social honeypot-based approach is depicted in Figure 5.31. Each social honeypot is associated with a legitimate profile inside a social network website to collect social spam behaviour.

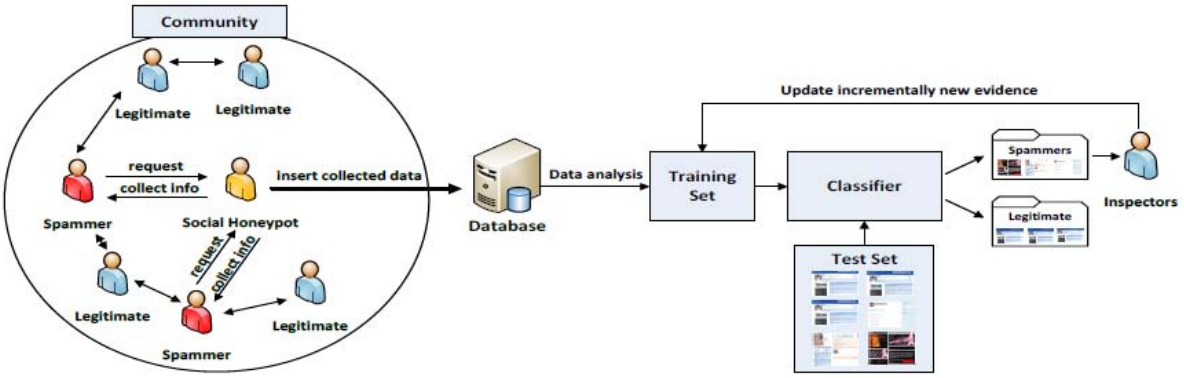


Figure 5.31: Framework for social honeypot-based approach (Lee et al. 2010).

However, the application of a social honeypot is not feasible for monitoring Spam 2.0 behaviour in general due to a number of practical considerations:

- Social honeypots collect mainly contextual information from spam profiles. Hence, they cannot be applied to collect spam users' navigational data.
- Social honeypots cannot track and monitor form usage data.
- Social honeypots are limited to only one spam domain i.e. social networking website. They cannot be extended to other platforms due to the nature of the tracking strategy.

HoneySpam 2.0 has a broader scope than social honeypots which is limited only to the social spam domain and social networking websites. The information collected by HoneySpam 2.0 is not traceable by social honeypots due to the latter's design constraints.

5.9.1.3 HoneySpam

(Andreolini et al. 2005) proposed a framework to address *email* spamming actions in one organisation. The authors proposed a method called *HoneySpam* to combat email spam activities such as *email harvesting* and *anonymous operations*. Email harvesting is an operation performed by spammers to collect and extract email addresses from web pages by browsing websites. Anonymous operations are activities that are performed by spammers to hide their traces. This can be done through use of the *open relay* (SMTP server that does not need authentication to forward emails) and *open proxies* (a network node to forward traffic between client and the servers without authentication). The use of multiple open relays and open proxy chains can hide a spammer's origin IP address.

HoneySpam includes four components: *fake web server*, *fake open relay*, *fake open proxy*, and *fake destination SMTP server* (Figure 5.32). Fake web server is for dynamic generation of web pages that are linked together and contain a significant *special* email address. Fake web servers can not only slow down email harvesting tasks, but also can pollute spammers' email database. Special email addresses, if used by a spammer, help to track the spammer's activity. Fake relay and open proxy if used by a spammer, reveal their origin information such as the IP address. All the communication to and from open relay and open proxy is logged. A fake destination SMTP server is used to redirect all the traffic to a special email address that is generated by a fake web server to a single mailbox for later analysis.

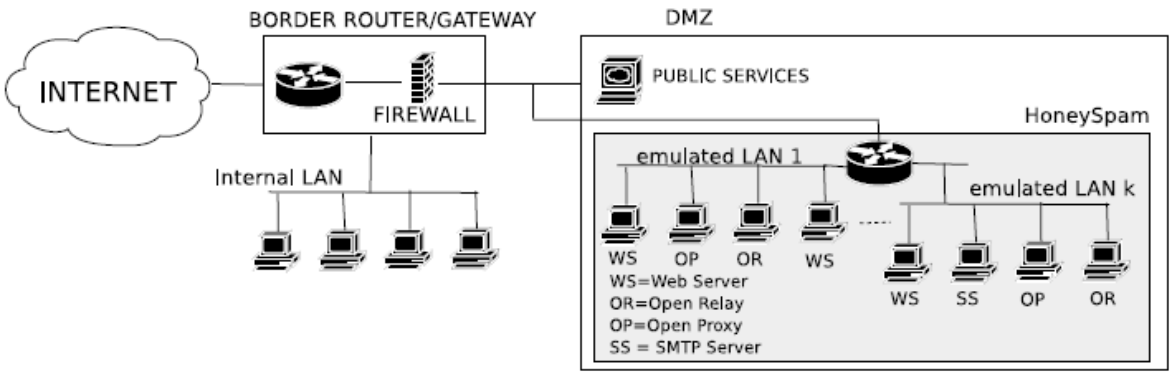


Figure 5.32: Architecture of HoneySpam to fight email spam (Andreolini et al. 2005)

However, it is clear that HoneySpam is unable to address Spam 2.0 problems since it is a framework to counteract email spam and not Spam 2.0. HoneySpam can track and compromise only email spammers, and therefore, it cannot be applied to Spam 2.0.

In this section, the validation and comparative study of the proposed method, HoneySpam 2.0, is discussed. It shows that HoneySpam 2.0 is superior to other existing solutions in addressing Spam 2.0 problems.

5.10 Conclusion

In this chapter, HoneySpam 2.0 was proposed to track, monitor and analyse Spam 2.0 behaviour. HoneySpam 2.0 is used to monitor Spam 2.0 distribution, track spammer interaction with web applications and store Spam 2.0 content. This information is later analysed and investigated to discover the characteristics of Spam 2.0.

HoneySpam 2.0 can be integrated with any web applications. It contains three parts: Tracking, Pre-Processing, and Knowledge discovery. The tracking part captures incoming traffic to the web application including web navigational and form usage data. The pre-processing part is for data cleansing, user and session identification of incoming requests. Finally, the knowledge discovery part identifies and characterises Spam 2.0 behaviour.

HoneySpam 2.0 has been executed on multiple web hosts including 6 commercial hosts and 5 free web hosts.

The following chapter provides an in-depth analysis of Spam 2.0 behaviour by using clustering tools.

Chapter 6

Profiling Spam 2.0

This chapter covers:

- Definition of Actions as a feature set to characterise Spam 2.0
- Profiling spam behaviour using Self-Organising Map (SOM)
- Evaluation and validation of SOM study on real-world data.

6.1 Introduction

Several characteristics of Spam 2.0 were discussed in Chapter 5 when introducing HoneySpam 2.0. HoneySpam 2.0 is used to monitor, track and store spam user activity on Web 2.0 platforms. General Spam 2.0 analysis such as web and form usage activities, demographic analysis, frequency of contents submission for Spam 2.0 were discussed in Chapter 5, which provided an initial understanding of the current trend of Spam 2.0.

However, more in-depth analysis of Spam 2.0 is required in order to:

- Investigate hidden spam user behaviour
- Understand the intentions of spam users
- Formulate spam users' behaviour in order to develop discriminative feature sets
- Cluster spam users' activities into groups based on feature sets
- Characterise spam users' behaviour in each cluster
- Develop a strategy to utilise feature sets for Spam 2.0 filtering

To achieve the abovementioned goals, this chapter presents a framework to model, formulate, and cluster spam users' behaviour. Based on the initial results of HoneySpam 2.0 (Section 5.8), three feature sets – Action Set, Action Time, and Action Frequency - are presented here to formulate spam users' behaviour on Web 2.0 platforms as well as a neural network clustering mechanism is used to cluster spam users' behaviour based on the three feature sets. The results of this chapter pave the way for the development of a Spam 2.0 filtering solution.

This chapter is organised as follows: Section 6.2 provides an introduction to feature sets to model spam users' behaviour. Section 6.3 introduces the Kohonen's Self-Organising Map (SOM), a neural network clustering mechanism for clustering spam users' behaviour. Section 6.4 provides the framework and algorithm for profiling Spam 2.0. Section 6.5 presents the implementation of the algorithm. The experimental settings for the experiment are presented in Section 6.6. Finally, the results and observations from the experiment are discussed in Section 6.7 .

6.2 Model Spam User Behaviour

Chapter 5 presented HoneySpam 2.0 as a framework to track spammers on Web 2.0 platforms. The tracked data contain information about spam users' web/form usage and submission. However, such raw data cannot reveal many covert spam user behaviours. The data need to be formulated and formatted in order to provide more insight into spam users' activities and model their behaviour.

As mentioned in Section 5.9, there were no interactions with forms. Hence, form usage data is excluded throughout this study.

The focus of this research is on behavioural analysis of Spam 2.0 and researchers have already presented detailed studies in the literature on spam in general (Hayati and Potdar 2008)(Hayati and Potdar 2009a) (Le, Jingbo, and Tianshun 2004). This section presents a framework for modelling and formulating usage behaviour of spam users on Web 2.0 platforms.

In the literature, there is no mechanism provided which models spam users' usage behaviour, especially on Web 2.0 platforms (Hayati, Potdar et al. 2010) (Hayati, Chai et al. 2010). Moreover, raw web usage data by itself is not descriptive and discriminative enough to allow a study of the behavioural aspects of spam users' activities on Web 2.0 platforms. Therefore, in order to formulate web usage data as a more descriptive model, the proposed framework introduces three key concepts:

Transaction identification, Action identification, and Behaviour vector construction which will be discussed in the following section.

6.2.1 Transaction

Transaction refers to the meaningful clustering of usage data (Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava 1999). Transactions are defined so that they are appropriate and meaningful for the given analysis. *Transaction identification* involves activities that are needed to construct meaningful clusters of usage data. In the proposed framework, three transaction identification approaches – Session level, User level and IP level - are used to make meaningful transactions.

6.2.1.1 Session level transactions

The session level transaction identification approach studies usage behaviour in each website visit. This approach reveals whether or not user behaviour changes for different website visits. For example, in an online discussion board, during one session the user might just read the other users' comments. While in another session, the user might reply to other users' posts. The usage behaviour for these two sample transactions is different. The session level transaction identification approach groups the set of user web page requests based on the same session ID.

The session level transactions can provide detailed information about spam user behaviour each time they visit the website. It investigates whether users change their behaviour when they have different intentions, or whether they follow similar behaviour most of the time.

6.2.1.2 User level transactions

The User level transaction identification approach investigates users' behaviour for the total time a user is active on a particular website. User level transaction contains entire sessions that belong to a user. It groups users' behaviour for all of their website visits. The User level transaction identification approach groups the set of user web page requests based on the same username.

The User level transaction studies behaviour of spam users for their whole life-time in the website. It investigates the cluster of different spam for the whole time they have been active in the website.

6.2.1.3 IP level transactions

The IP level transaction identification approach studies the behaviour of a specific host during the total amount of time that a host is connected to the website. The IP level might have more than one

user. The IP level transaction identification approach groups the set of user web page requests based on the same IP address.

The IP level transaction studies behaviour of spam hosts by investigating the behaviour of all the spam users that have the same IP address and clusters different behaviours within that host set.

Figure 6.1: Relationship of the three proposed approaches to transaction identification

Figure 6.1 illustrates the relationship among three the transactions identification approaches. It is clear from the figure that IP level transactions have many-to-many relationships with the User level, while Users level transactions contain multiple sessions (one-to-many relationship).

The above three transaction identification approaches are used to define transactions in the proposed framework. In the next step, web usage data is organised into meaningful groups based on each transaction.

6.2.2 Action

'Action' refers to a user set of requested web objects that are required to perform a certain task or achieve a particular purpose.

For instance, (1) a user can navigate to the registration page in an online discussion board, (2) fill in the required fields and (3) press the submit button in order to register a new user account. This three-step procedure can be modelled as a "Registering a user account" Action.

An Action integrates users' activity on web platforms into meaningful groups. Each group contains meaningful information about users' tasks. The importance of Action is as follows.

- It acts as a suitable discriminative criterion to model user behaviour on Web 2.0 platforms. Actions highlight the intentions of users who perform tasks on Web 2.0 platforms.
- It is extendable to many other Web 2.0 platforms. For instance, the "Registering a user account" Action is performed on numerous Web 2.0 platforms (forums, social networks etc.), as users often need to create an account in order to contribute content.

- It is language-independent. Actions are content-independent and model web usage behaviour, which is then language-independent.
- It can be implicitly formulated. Actions can be implicitly gathered and formulated while users navigate though websites. The collection and monitoring of Actions happen in the background and do not interrupt users while they are using the system.

Actions can be generated manually. They can be specific for one Web 2.0 platform or general across multiple platforms. Table 6.1 presents examples of Actions.

Table 6.1: Examples of general (G) and specific (S) Actions

Action	Web 2.0 platform	Type
Registration of user account	Forums, Wikis, Social networks, Comment systems etc.	G
Login & Logout	Forums, Wikis, Social networks, Comment systems etc.	G
Search	Forums, Wikis, Social networks, etc.	G
Read content	Forums, Wikis, Social networks, Comment systems etc.	G
Edit / Update content	Forums, Wikis, Social networks, Comment systems etc.	G
Add content	Forums, Wikis, Social networks, Comment systems etc.	G
Update user's profile	Forums, Wikis, Social networks, Comment systems etc.	G
Compose private message	Forums, Social networks	S
Send friendship request	Social networks	S
Create new page	Wikis	S
Send event invitation	Social networks	S

The main reason for studying Actions is that spam users' intentions on Web 2.0 platforms are different from those of genuine humans. Spam users comparatively are active in content distribution, whereas genuine humans spend more time on content consumption than distribution (Steven J. J. Tedjamulia 2005). Therefore, factors such as frequency of different Actions, the amount of time a user spends on performing each Action etc. can be intelligently interpreted to differentiate between spam users and genuine ones.

In this research, three models or feature sets – Action Set, Action Time, and Action Frequency - are proposed to formulate web usage data. These feature sets are used to differentiate between spam and genuine behaviour.

6.2.2.1 Action Set

An Action Set is a binary model. It shows whether or not an Action is undertaken by a user. This feature set shows whether or not spam users perform different Actions in different transactions. For example, a spam user might concentrate only on those Actions that relate to submitting spam content.

6.2.2.2 Action Time

Action Time is a feature vector which represents the time taken to perform each Action. This feature set shows how much time spam users would spend on performing each Action in different transactions. For example, the amount of time to write a reply for a spam user might be a lot shorter than for a human user since spam content distribution is typically done automatically.

6.2.2.3 Action Frequency

Action Frequency is a feature vector to represent the frequency of each Action. This feature set shows how frequently spammers perform each Action in a different transaction and which Actions are quite common and which are not. For example, Action Frequency for content submission is more common with spam users than are other Actions.

Actions are formulated in a vector format known as a Behaviour vector to model a user's web usage behaviour. Behaviour vector construction is discussed in the next section.

6.2.3 Behaviour vector

The behaviour vector is a representation or model of a user's web usage behaviour in the form of a multi-dimensional vector. It is propagated with a user's Actions at different transaction levels. Three Behaviour vectors are presented for Action Set, Action Time and Action Frequency.

6.2.3.1 Action Set vector

Given a set of web pages $W = \{w_1, w_2, \ldots, w_{|W|}\}$, A is defined as a set of Actions, such that

$$A = \{a_i | a_i \subset W\} = \{\{w_l, \ldots w_k\}\} \quad 1 \leq l, k \leq |W| \qquad 6.1$$

Respectively s_i is defined as

$$s_i = \{a_j\} \quad 1 \leq i \leq |T|;\ 1 \leq j \leq |A| \tag{6.2}$$

s_i refers to a set of actions performed in transaction i and T is total number of transactions.

In order to build the input vector, each action (a_i) is assigned as a feature item. Hence, an Action Set is represented as a bit vector

$$\overrightarrow{aS} = (v_1^i, ..., v_{|A|}^i) \tag{6.3}$$

where

$$v_j^i = \begin{cases} 1 & a_j \epsilon s_i \\ 0 & otherwise \end{cases} \tag{6.4}$$

6.2.3.2 Action Time vector

The amount of time a user spends on a specific web page in his/her navigation sequence is considered as 'dwell time'. As defined in E.q. 6.6, Dwell time is used to construct an Action Time vector.

$$d = t_{i+1} - t_i \quad where\ 1 < i < |S| \tag{6.5}$$

d is a dwell time for i^{th} requested web page in transaction S at time t;

In E.q. 6.6, it is not possible to calculate dwell time for the last visited web page in a session. For example, a user navigates to the last web page on the website then closes his/her web browser. In the proposed method, the average dwell time spent on other web pages in the same session is considered as the dwell time for the last web page.

Action Time is defined as a vector where

$$d_j^i = \begin{cases} \frac{\sum_{k \epsilon a_j} d'_k}{h_j^i} & a_j \epsilon s_i \\ 0 & otherwise \end{cases} \tag{6.6}$$

d^i_j is a dwell time for Action a_i in s_i which is equal to total amount of time spent on each web page in a_i . In cases where a_i occurs more that once, the average dwell time would be calculated. Action Time vector is:

$$\overrightarrow{aT} = (d^i_1, ..., d^i_{|A|}) \quad 6.7$$

6.2.3.3 Action Frequency vector

Action frequency is a vector where h^i_j is the frequency of j^{th} Action in s_i , otherwise it is zero. The Action Frequency vector is:

$$\overrightarrow{aF} = (h^i_1, ..., h^i_{|A|}) \quad 6.8$$

Each of above Behaviour vectors is multi-dimensional. In order to understand the behaviour vector as well as profile characteristics of each behavioural vector, a tool called a self-organising map (SOM) is used. SOM is utilised to visualise as well as cluster data. The next section provides a discussion on clustering spam user behaviour using a SOM.

6.3 Cluster Spam User Behaviour

Behavioural vectors are multi-dimensional and are therefore easy to visualise. In order to understand this data, a *Self-Organising Map* (SOM) is used to reduce the dimension of data as well as cluster data into similar groups. Each cluster provides an insight into the nature of Spam 2.0 behaviours and their characteristics.

6.3.1 Self-Organising Map (SOM)

SOM – a neural network clustering algorithm – can be employed for clustering and visualisation of high dimensional data (Kohonen 1990). The output of the SOM is a map which is a visual representation of an input vector. Maps can be generated in two or three dimensions but two-dimensional maps are more common (Kohonen 1990).

SOM has two phases: training and mapping. In the training phase, an input vector in the form of a one-dimensional array is presented to the map. Next, the weights of the components on the map (called nodes) which neighbour the input vector gain more strength. The node which is closest to the input vector is called Best Matching Unit (BMU). Therefore, in the mapping phase, similar input vectors are in the same region and can be grouped together as a cluster.

6.3.1.1 SOM definition

SOM starts by initialising each node's weight, w_i , vector treated as a random variable that is to be assigned a small value (w_i *presented here is representation of SOM weight vector not set of webpages as discussed in Section 2.2.1*). Next, an input vector x is presented to the map and the distance between all nodes on the map and x is calculated as E.q. 6.9.

$$d_j = \|x - w_j\| = \sqrt{\sum_{k=0}^{|w|} (x_{ik} - w_{kj})^2} \quad 6.9$$

The minimum d_j is selected as BMU. After finding BMU, SOM updates w_i for BMU and its neighbours nodes according to E.q. 6.10 so they get closer to the input vector.

$$w_i(t+1) = w_i(t) + \alpha(t) h_{ci}(t)[x - w_i(t)] \quad 6.10$$

where t is the time-step, $h_{ci}(t)$ is a Gaussian neighbourhood kernel and decreases with time

$$\alpha(t) = \alpha(0)\frac{(1-t)}{T} \quad 6.11$$

$\alpha(t)$ is linear learning rate, T is the training length (neighbourhood size less than number of nodes in one dimension of map).

6.3.1.2 Unified distance matrix

Unified distance matrix (U-Matrix) visualises the distance between adjacent nodes in SOM. It can be employed to determine the number of clusters in SOM as well as their closeness. In this study, SOM and the U-matrix are used as visualisation tools to cluster different Spam 2.0 behaviours.

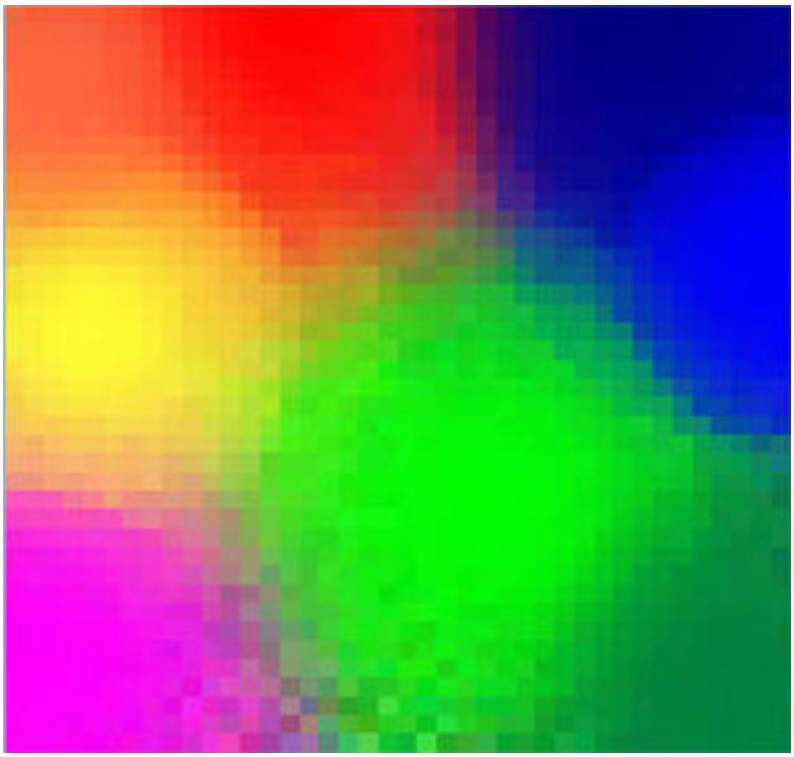


Figure 6.2: U-Matrix for mapping of 8 RGB colour vector on two dimensions

For example, colours in Red Green Blue (RGB) model the three primary colours that generate other colours (Desai 2008). The RGB colour model is presented as a triple (red, green, blue) that contains three components corresponding to red, green and blue. The value of each component varies between zero to a defined maximum value. Figure 6.2 illustrates the U-Matrix for 8 RGB triple vectors on two dimensions.

SOM and U-Matrix are used to cluster and project spam users' data. The following section provides discussion on input vectors that are used with SOM.

6.4 Algorithm

Three Behavioural vectors – Action Set, Action Time and Action Frequency are used as inputs to SOM. The output of this process is a 2D U-Matrix map for each Behavioural vector. The projection of input data from a high dimension to a 2D map allows understanding of each feature set, grouping similar input vectors, and studying each cluster of Spam 2.0 behaviour.

The three transaction levels and Behavioural vectors that were discussed earlier in this chapter are combined to construct 9 input vectors. Table 6.2 presents 9 feature vectors that are used in SOM.

Table 6.2: Nine feature vectors used in SOM.

	Action Set (*aS*)	**Action Time (*aT*)**	**Action Frequency (*aF*)**
Session (*S*)*	(*S*, *aS*)	(*S*, *aT*)	(*S*, *aF*)
User (*U*)	(*U*, *aS*)	(*U*, *aT*)	(*U*, *aF*)
IP (*I*)	(*I*, *aS*)	(*I*, *aT*)	(*I*, *aF*)

** Where S, U and I are set of session, users and IP addresses.*

Each input vector is used as input for SOM and is projected using U-Matrix. To study the cluster, the centre node of each cluster is chosen to describe the characteristic of that particular cluster.

6.4.1 Flowchart

Figure 6.3 illustrates the flowchart diagram for the clustering algorithm. This process consists of the following steps:

Step 1: Get Behavioural vectors for each feature set – Action Set, Action Time and Action Frequency.

Step 2: Generate 9 input vectors based on each feature set and three transaction levels.

Step 3: Use input vectors to train SOM.

Step 4: Generate U-Matrix corresponding to each input vector.

Step 5: Investigate the centre node of each cluster in U-Matrix to profile Spam 2.0 behaviour.

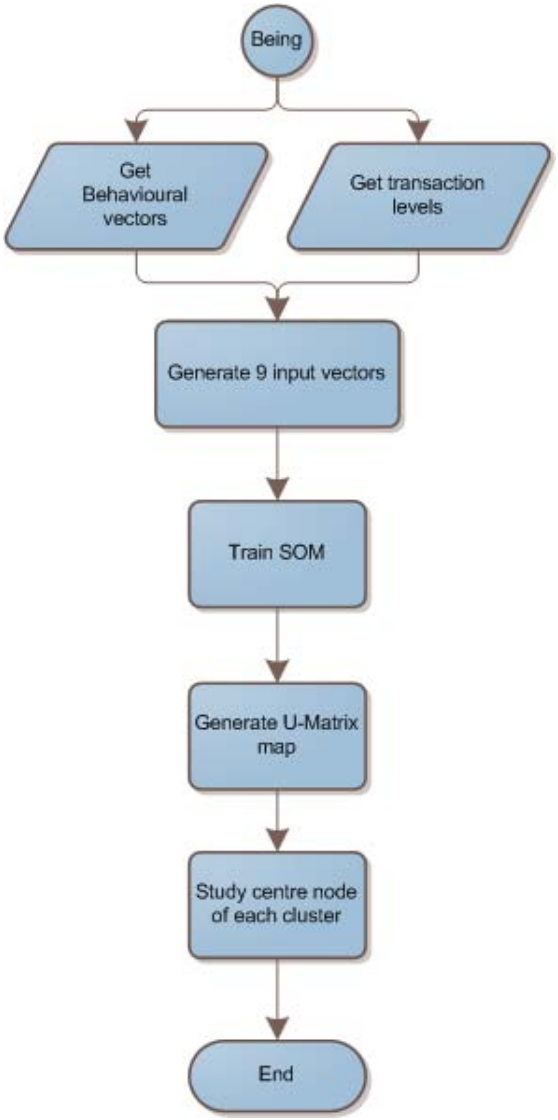


Figure 6.3: Flowchart for the proposed algorithm

6.5 Implementation

PHP and MATLAB are used to implement the proposed framework. MATLAB is well-known tool for data processing and data analysis. In the proposed method, MATLAB is used to generate general statistical reports. Additionally, the *MATLAB Neural Network Tool Box* (*SOM Tool Box)* is utilised for SOM and U-Martix generation. The PHP codes are used to perform transaction identification, generating Actions and Behavioural input vectors.

6.5.1 Transaction identification

As mentioned earlier, the transaction identification task considers three levels: session, user and IP. In order to group transactions at each level, the following SQL statement is employed.

***GROUP BY** (session, user, IP column name)*

For example, in order to group data in session, user, and IP transaction levels, the following SQL statements are created.

- ***GROUP BY** (session_id)*
- ***GROUP BY** (username)*
- ***GROUP BY** (clientIP)*

These SQL statements are queried in conjunction with the other SQL queries to retrieve data for each transaction level.

6.5.2 Action Set vector

Action Set is constructed by following algorithm Code 6.1.

Code 6.1: Pseudo-code for the Action Set

```
Algorithm action_set(T,A)
    input: set of transaction by the given user, T
             set of Actions, A
    output: set of Action Set for the given user, s
1.   s = null
2.   for each t_i in T
3.     create new action a
4.     for each w_j in t_i
5.       add w_j to a
6.       if a in A
7.         if a in s
8.           discard a
9.           else
10.            s = s + a
11.          end if
12.        t_i = t_i - a
13.        if length(t_i) > 0 then
14.          create new action a
15.        end if
16.      else
17.        if j = length(t_i) then
18.          discard a
19.        end if
20.      end if
21.    repeat
22.  repeat
23. return s
```

For example, suppose

$W = \{w_1, w_2, w_3, w_4, w_5\}$

$T = \{t_1, t_2\}$ (*set of transactions for the given user*)

where

$t_1 = \{w_1, w_2, w_1, w_5\}$ (in this transaction the user navigates from w_1 to w_2 and then again to w_1 and w_5)

$t_2 = \{w_1, w_3\}$ (the user navigates to w_1 and then to w_3)

Set of defined Actions is defined as $A = \{a, b, c\}$

where

$a = \{w_1, w_2\}$ (*register to the website*)

$b = \{w_3, w_4\}$ (*post new topic*)

$c = \{w_1, w_5\}$ (*login to the website*)

the set of performed Actions by the given user is

$s = \{a, c\}$ (*user register and then login to the website*)

On line 7, the algorithm checks whether the Action has already been added to the set, *s*; if so, *a* would not be discarded. Otherwise, *a* would be added to the output set.

6.5.3 Action Time vector

Action Time algorithm – the amount time spent on performing each action – is presented in Code 6.2.

Code 6.2: Pseudo-code for Action Time algorithm.

```
Algorithm action_time(T,A)
    input: set of transaction by the given user, T
             set of Actions, A
    output: set of Action Time for the given user, s
1.  s = null
2.  for each t_i in T
3.    create new action a
4.    for each w_j in t_i
5.      add w_j to a
6.      if a in A
7.        if a in s
8.          s_a.time = (s_a.time + a.time) / 2
9.        else
10.         add a, a.time to s
11.       end if
12.       t_i = t_i - a
13.       if length(t_i) > 0 then
14.         create new action a
15.       end if
16.     else
17.       if j = length(t_i) then
18.         discard a
19.       end if
20.     end if
21.   repeat
22.  repeat
23. return s
```

The main difference between Action Set and Action Time algorithms is on line 8 and 10. On line 8, the average Action time is added to the set and on line 10, Action name and Action time are added to the set.

6.5.4 Action Frequency vector

Similarly to the Action Set algorithm, the Action Frequency algorithm is defined in Code 6.3.

Code 6.3: Pseudo-code for Action Frequency algorithm.

```
Algorithm action_frequency(T,A)
    input: set of transaction by the given user, T
             set of Actions, A
    output: set of Action Time for the given user, s
1.  s = null
2.  for each t_i in T
3.    create new action a
4.    for each w_j in t_i
5.      add w_j to a
6.      if a in A
7.        s = s + a
8.        t_i = t_i - a
9.        if length(t_i) > 0 then
10.         create new action a
11.       end if
12.     else
13.       if j = length(t_i) then
14.         discard a
15.       end if
16.     end if
17.   repeat
18.  repeat
19. return s
```

In the Action Frequency algorithm, the count for each Action *a* is added to the output set.

6.5.5 Input vector

The combination of three feature vectors – Action Set, Action Time, and Action Frequency and three transaction levels – Session, User, and IP - results in 9 input vectors that can be used by the SOM and U-Matrix generation. The following three algorithms are used to construct – Action Set, Action Time, and Action Frequency input vectors for each transaction level Code 6.4.

Code 6.4: Pseudo-codes for the construction of feature input vectors.

```
Algorithm action_set_input_vector(k,sk,A)
    input: given user, k
             set of user Action Set, sk
             set of Actions, A
    output: Action Set input vector, aS
1.  for each ai in A
2.    if ai in sk
3.      aSi = 1
4.    else
5.      aSi = 0
6.  return aS

Algorithm action_time_input_vector(k,sk,A)
    input: given user, k
             set of user Action Time, sk
             set of Actions, A
    output: Action Time input vector for the given user, aT
1.  for each ai in A
2.    if ai in sk
3.      aTi = ai.time
4.    else
5.      aTi = 0
6.  return aT

Algorithm action_frequency_input_vector(k,sk,A)
    input: given user, k
             set of user Action Frequency, sk
             set of Actions, A
    output: Action Frequency input vector for the given user, aF
1.  for each ai in A
2.    if ai in sk
3.      aFi = ai
4.    else
5.      aFi = 0
6.  return aF
```

Algorithm `action_set_input_vector(`k, s_k`,A)`obtains Action Sets for a user. It iterates through *A*, and if Action a_i is performed by the user, the value for the aS_i would be 1 otherwise 0. The algorithm returns Action Set input vector, aS_i, on line 6.

The algorithm `action_time_input_vector(`k, s_k`,A)` iterates through Actions, if the Action exists in *s*, on line 4, algorithm assigns dwell time value for aT_i otherwise the algorithm assigns a zero value. On Line 6, the algorithm returns Action Time input vector, *aT*.

Similarly to the Action Set and Action Time input vector algorithm, Action Frequency input vector algorithm `action_frequency_input_vector(`k, s_k`,A)`iterates through Actions, if Action exists

in user session, *s*, line 3, the frequency of the action is added to input vector aF_i otherwise the zero value would be assigned. The algorithm returns the Action Frequency input vector, *aF*, on line 6.

6.6 Experimental Setting

This section discusses SOM and U-Matrix experiments settings. The dataset is collected through HSCOM1 forum data (Section 5.7.1.1) over a period of one month. Table 6.3 presents the parameter specifications of the experiment.

Table 6.3: Parameter specifications for the experiment

Parameter	Value	Description
No. of session transactions	3726	Total number of transactions as session level
No. of user transactions	1067	Total number of transactions as user level
No. of IP transaction	871	Total number of transactions as IP level
No. of Actions , |A|	11	Manually defined Action list.

For transaction identification, HSCOM1 forum data is grouped into three transaction levels: session, user and IP. 11 Actions are manually extracted from the dataset (*A*). The details of each action are presented in Table 6.4. Feature sets – Action Set, Action time, and Action frequency - are measures based on this pre-defined Action list.

Table 6.4: Summary of Actions performed by spammers on HSCOM1 forum data

Action Key	Action Index	Description
1	A	View root page
2	B	View topics
3	C	Start new topic
4	D	View topic (view their own created topic)
5	E	View user profile information
6	F	Edit user information
7	G	Start new poll
8	H	User authentication (Login)
9	I	Reply to topic

Action Key	Action Index	Description
10	J	Recently submitted topics
11	K	View overall forum statistic

In order to provide a clear insight into each Action, Figure 6.4 represents the frequency of each Action at the three transaction levels in the dataset.

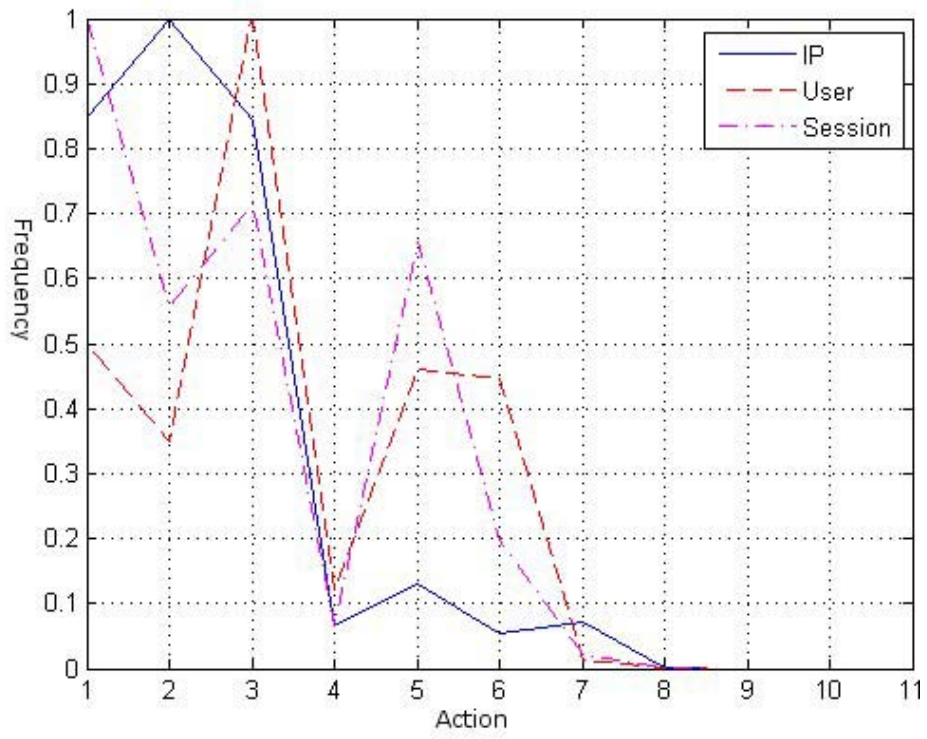


Figure 6.4: Frequency of Actions on HSCOM1 forum data

As is clear from the figure, “Starting new topic” is the most common Action among spam users in all the transactions levels. The Action “View root page” and “View user profile information” are common in session level transactions. The Action “View topic” is also common in the IP level transaction.

Nine input vectors are constructed. Table 6.5 presents a sample input vector for Action Time that is used in SOM.

Table 6.5: Sample Action Time input vector used in SOM experiment

	a_1	a_2	…	$a_{\|A\|}$
t_1	5	4	…	0
t_2	0	4	…	0
…	…	…	…	…
$t_{\|T\|}$	0	2	…	1

6.7 Results

In the following section, the results from each experiment and the characteristics of the *centre* node inside each cluster are discussed.

6.7.1 Experiment 1: Session level result

Based on three experiments at session level, Figure 6.5 illustrates the U-Matrix for (*S*, *aF*). The dark line represents large distances between nodes. The light node areas represent nodes that are close to each other. The dark lines separate the map into the major clusters. The details of each cluster are shown in Table 6.6.

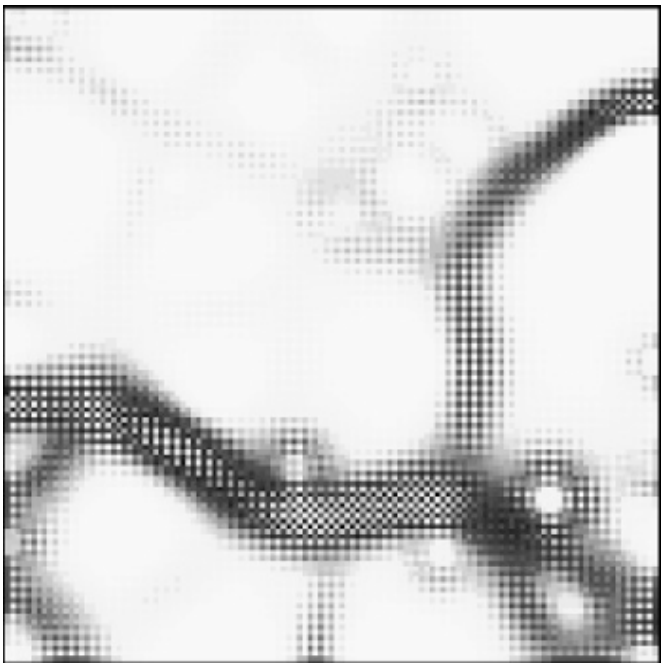


Figure 6.5: Result of U-matrix for session and frequency of actions.

Table 6.6: Major characteristics of spam users at the session level

	Cluster 1	Cluster 2	Cluster 3
Dwell time	N/A	2,3	4,6,7
Frequency	1	2,3	4,9
Actions	C, A	AE, BB, BCB	AEFF, AEFFABCBD

Three major clusters are extracted from the above experiment and are labelled: *Content Submitters*, *Profile Editors* and *Mixture*.

Cluster 1 (Content Submitter): Spam users in this cluster perform the *Starting New Topic* action. They did not perform any other action at this level. Hence, the dwell time for their Action cannot be calculated. This behaviour reveals that spammers do not navigate through websites. They directly submit the content on the forum. Additionally, in their Action Set there is no request for Action C (start new topic), thereby indicating that they do not interact with the server form in order to submit their content.

Cluster 2 (Profile Editor): The goal of spam users in this cluster is to edit their own profile page. A profile page consists of various types of information such as name, email address, homepage URL,

signature, etc. Spam users navigate from Action A (view root page) to Action F (edit user information) with a frequency of one for each Action. The dwell time for their action is between 2-3 seconds. This strengthens the claim that they do not use forms to submit their profile details, since their dwell time is very low (2-3 seconds).

Spam users, by modifying the profile information, are able to promote either their spam URLs or content.

Cluster 3 (Mixture): Spammers in this cluster performed both of the abovementioned sets of Actions. They navigate through the following action sequence (A,E,F,F,A,B,C,B,D). The average frequency for each Action is 2, which means they perform the above sequence twice in each session. This behaviour again shows that spammers did not use forms to submit content or modify their profile, since this sequence cannot feasibly be performed in 4 – 7 seconds. Additionally, they navigate to view their submitted content in order to increase the view count of their topic, as well as to possibly ensure that their submitted content is published.

6.7.2 Experiment 2: User level results

User level tracking provides information about the characteristics of each spammer for their total active time in the forum. It can show whether their behaviour has changed over time. The results of three different experiments at the user level are:

Cluster 1 (Content Submitter): Spam users use their user account only to submit spam content directly to the website. However, they use their username once since only one session belongs to each user in this cluster. The dwell time is zero (or unknown as only 1 web page request), the frequency of the Action is either one or two and they perform only Action C (start new topic).

Cluster 2 (Content Submitter and Profile Editor): Spam users start their Action navigation by visiting the root page, modifying their profile page and then submitting spam content. Their average dwell time is 2.5 seconds for each Action with a total frequency of 8 actions per user level. Their navigation sequence is (A,E,F,F,A,B,C,B). Once the Actions were performed, the spammers did not use the user account again.

Cluster 3 (Mixture): In this cluster, the average dwell time of spam users is 7 seconds for navigating navigation through (A,A,E,F,F,A,B,C,B,B,D) with a frequency of 11. They combine both profile

editing and content submission behaviours. They view their own created topic at the end of their navigation set. Similar to the previous clusters, spam users have just one session for each user account in this cluster.

An investigation of spam users at the user level reveals that once they have created a user name, they used it once only and they do not change their behaviour during their lifetime. Table 6.7 presents details of each cluster. Figure 6.6 illustrates U-Matrix map for (*U, aF*). It shows that the majority of spam users at the user level have similar characteristics as the map dominated by light areas.

Figure 6.6: U-Matrix representation of user level and action frequency.

Table 6.7: Major characteristics of spam users in the user level

	Cluster 1	Cluster 2	Cluster 3
Dwell time	N/A	2.3	2.5
Frequency	1	8	11
Actions	C, CC	AEFFABCB	AAEFABCBBD

6.7.3 Experiment 3: IP level results

The IP level presents a high level view of spam users' behaviour. The experiment resulted in the discovery of three major clusters which include:

Cluster 1 (Mix): This cluster consists of 4 – 5 frequencies for each Action Set. Spam users in this cluster navigate through the root web page, modify their profile and submit spam content. Finally, they end their navigation by viewing their own created content. The frequency shows that they created multiple accounts under one IP address name to perform their actions.

Cluster 2 (Content Viewer): Spam users in this cluster only navigate to Section D (viewing their own topic). This behaviour was not seen in the previous levels. Further investigation revealed that spam users change their IP address once they submit the spam content. Therefore, some use multiple IP addresses for one user (many-to-many relationship between IP and user transaction level).

Cluster 3 (Big Hit Mix): Action Frequency in this cluster is more than 100 for submitting spam content or to modify profiles. This cluster has similar characteristics to those of cluster 1. However, the higher Action Frequencies show that spammers use multiple usernames to perform the same Actions.

The experiment involving an IP address dwell time map does not result in clear clusters. This may be due to a change of IP address by spammers at the end of navigation which results in unknown dwell time as well as multiple usernames per IP address that produce various dwell times. Table 6.8 presents details of each experiment at IP level. Figure 6.7 represents the U-Matrix map of IP level and Action Frequency.

Figure 6.7: U-Matrix for IP level and Action Frequency

Table 6.8: Major characteristics of spam users in the IP level

	Cluster 1	Cluster 2	Cluster 3
Frequency	8	11	100+
Actions	AEFFABCB	D…	AEFABC…

6.7.4 Discussion

The U-matrix visualisation in some of the experiments produced a number of minor clusters. While the investigation was carried out on these minor clusters, they did not result in useful distinctive behaviour for Spam 2.0 characterisation. For example, a cluster was identified for spam users who perform Action A (visiting the root page) multiple times. This Action does not reveal any interesting characteristics since it can be seen among genuine users as well. Hence, such reports for minor clusters were not included in the results.

A self-organising map and neural networks in general do not allow much user interpretability in understanding why nodes belong to each cluster. It can be viewed as a black box algorithm which is fed some inputs and generates the output.

6.8 Conclusion

This chapter presents a framework to investigate, formulate, and cluster Spam 2.0 behaviour. Based on initial results of HoneySpam 2.0 (Section 5.8), three feature sets – Action Set, Action Time, and Action Frequency - are presented here to formulate Spam users' behaviour on Web 2.0 platforms. A neural network clustering mechanism is used to cluster spammer behaviour based on the three feature sets. The following observations have been made during the unsupervised classification experiment.

- Spam users focus on specific and limited Actions in the web applications.
- Spam users use multiple user accounts to spread spam content, hide their identity and bypass restrictions.
- Spam users do not fill in submission forms and directly submit the content to the Web server in order to efficiently spread spam.
- Spam users can be grouped into four different categories based on their Actions: content submitters, profile editors, content viewers and mixed behaviour.
- Spam users change their IP address based on different Actions to hide their tracks

The following chapter provides the first early-detection based Spam 2.0 filtering method based on behaviour analysis.

Chapter 7

Early-Detection based Spam 2.0 Filtering

This chapter covers:

- Introduction to Early-detection-based Spam 2.0 filtering solution.
- Presentation of a framework to detect Spam 2.0 by analysing web usage data and evaluate its feasibility in combating the distribution of Spam 2.0.
- Utilising Action Set, Action Time and Action Frequency for Spam 2.0 identification.
- Evaluation, validation and a comparative study of the performance of the proposed solution with real-world data and state of the art approaches.

7.1 Introduction

This chapter presents a method for **Early-Detection based Spam 2.0 Filtering** called **EDSF**. This Spam 2.0 filtering method is used to identify spam behaviour prior to or after spam content submission. EDSF investigates the behaviour of the spam users e.g. automated tools, spambots etc.

The EDSF, the proposed method presented in this dissertation, automatically distinguishes Spam 2.0 submission from genuine human content contribution on Web 2.0 platforms. EDSF makes use of web usage navigation behaviour to build up a discriminative feature set. Such a feature set is later on passed to a machine learning classifier known as a Support Vector Machine (SVM) for classification.

The theoretical foundation for EDSF, along with its algorithm design to address all the requirements laid out in Chapter 4, is discussed here. Testing, experimentation and observation of the results along with their validation are carried out at the end of this chapter.

7.2 General Overview of EDSF

The main assumption of EDSF is that human web usage behaviour is intrinsically different from Spam 2.0 behaviour. The reason for this, as outlined in Section 6.7, is due to spam users having different intentions. Spam users visit the websites mainly to spread spam content rather than contribute to the content. Hence, by investigating and mining web usage data, it is possible to distinguish them from genuine human users.

Web usage data can be implicitly gathered while users surf though websites. This raw data needs to be converted into a different format to highlight behavioural differences between spam and genuine users. Therefore, three new feature sets namely, Action Set, Action Time, and Action Frequency, are used in EDSF to study spammers' behaviour.

As mentioned earlier, an Action can be defined as a user set of requested web objects in order to perform a certain task or purpose (ref. Section 6.2.2). Actions can be a suitable discriminative feature to model user behaviour within forums but can also be extendable to many other Web 2.0 platforms.

EDSF utilises Action Set, Action Time, and Action Frequency to formulate web usage behaviour. As discussed in Section 6.2.2, Action Set is a set of Actions performed by a user. Action time is the amount of time spent on doing a particular Action. Action Frequency¸ is the frequency with which one certain Action is performed.

EDSF use these three features in a machine learning classifier (SVM) to differentiate spam from genuine users.

This chapter presents:

1. An introduction to Early-Detection based Spam 2.0 Filtering (EDSF) approach.
2. A framework to detect Spam 2.0 by analysing web usage data and evaluate its feasibility in combating the distribution of Spam 2.0.
3. The implementation of Action Set, Action Time and Action Frequency for Spam 2.0 identification.
4. An evaluation and validation EDSF with real world data.

5. A comparative study of EDSF with state-of-the-art Spam 2.0 filtering approaches.

7.3 Requirements

The following requirements are laid down for the proposed solution. This represents the *first* stage of the conceptual process where requirements are elicited and prioritised. The requirements of the solution are:

1. **Extendibility**: the method should be extendable to all Web 2.0 platforms.
2. **Adaptability**: the method should be adaptable to new Spam 2.0 behaviours.
3. **Automaticity**: the method should automatically distinguish Spam 2.0 from genuine human behaviour.
4. **Independency**: the method should be independent from its platform. For example, it should be language- and content-independent.
5. **Implicitness**: the method should implicitly accumulate behavioural data.
6. **Discriminative features**: the features should be discriminative enough to differentiate genuine human user behaviour from spammers' behaviour.

The design rationale is based on the abovementioned requirements. A design decision for each requirement is made in the design rationale along with a discussion of how the requirement is met.

7.4 Design Rationale

This section provides a design rationale to meet all the requirements listed above. This is the *second* stage of the conceptual process. The following design decisions are proposed in order to address each requirement.

1. Develop an early-detection based Spam 2.0 filtering method that is language- and platform-independent (Req. 1, 4).
2. Capture server side web usage data while the user interacts with the website (Req. 1, 5).

3. Develop an automated supervised machine learning classifier than can be trained with new instances of Spam 2.0 (Req. 2, 3).

4. Formulate web usage data into the Action format i.e. Action Set, Action Time, and Action Frequency (Req. 6).

In the development of any solution, the above design decisions should be considered. All these design decisions are included in the proposed method to detect Spam 2.0 behaviour on web applications. This concludes stage two of the conceptual process. The next stage is the theoretical foundation for the proposed solution.

7.5 Theoretical Foundation

The theoretical foundation for the EDSF is discussed in this section which represents the *third* stage of the conceptual process. Entire design decisions are analysed and an algorithm is proposed. The theoretical foundation as illustrated in Figure 7.1 has four main parts: tracking, pre-processing, feature measurement, and classification.

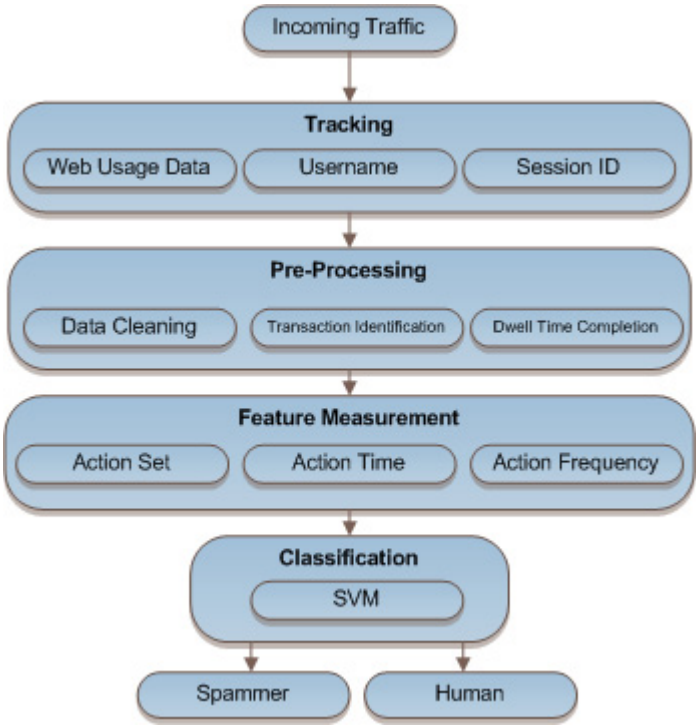


Figure 7.1: EDSF framework consisting of four parts

Incoming Traffic

Incoming traffic shows a user entering the website through a web interface such as the homepage of a forum.

Tracking

Tracking records web usage data including the user's IP address, username, requested webpage URL, session identity, and timestamp.

Pre-processing

Pre-processing includes three components: data cleansing, transaction identification and dwell time completion.

Feature Measurement

Feature Measurement extracts and measures three feature sets: Action Set, Action Time, and Action Frequency from web usage data.

Classification

A machine learning classifier known as Support Vector Machine (SVM) is used to distinguish spam users from genuine human behaviour.

The proposed solution explained above addresses all the requirements described in Section 7.3 . A detailed description of each part of EDSF is given below.

7.5.1 Tracking

Input: Incoming traffic to the Web 2.0 platform

Output: Web usage data

Tracking is the entry for all incoming traffic. It tracks the requested URL, type of request method (i.e. GET and POST method), the time of request, the IP address, the User Agent, and the referrer URL.

The tracking part of EDSF is similar to HoneySpam 2.0 tracking (Section 5.5.1). However, since current trends of Spam 2.0 do not interact with the form (Section 5.9), no form usage data is collected during the tracking period. Therefore, the existence of any form usage data can be an indicator of genuine behaviour on a web application. In order to disregard such simple features for classification and further investigate navigational behaviour, form usage was not included in the EDSF experiment.

Similar to the *Session identification* task in HoneySpam 2.0 (Section 5.5.2.3), a unique session identifier is assigned for individual browsing sessions. This makes it possible to track navigations on each occasion that the user visits the website. The tracker stores the gathered data along with the corresponding username for each record. The users' credentials information is extracted from each HTTP cookie and is stored along with other navigational usage data similar to the *User identification* task in HoneySpam 2.0 (Section 5.5.2.2).

Figure 7.2 presents a sequence diagram of the tracking process.

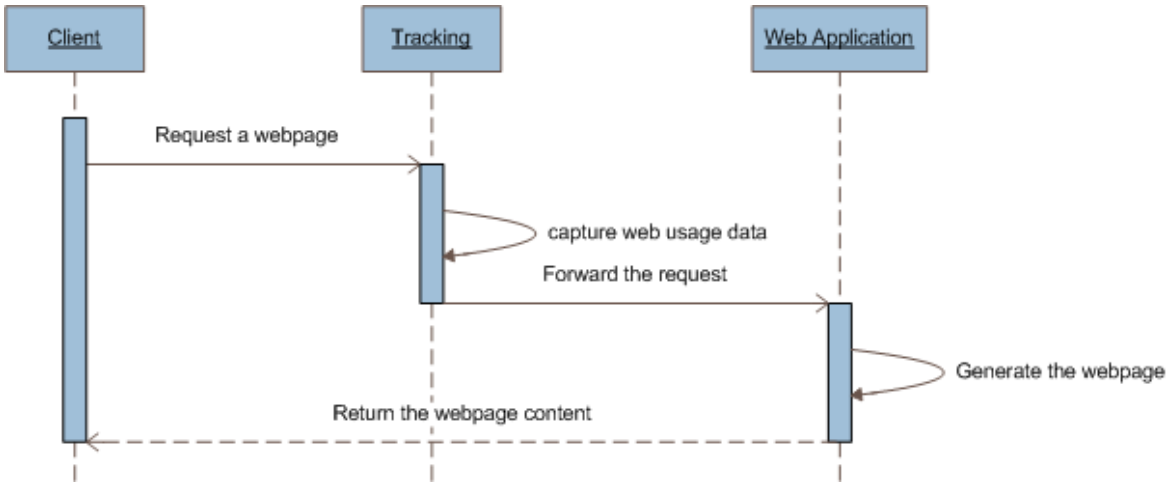


Figure 7.2: The sequence diagram of the EDSF tracking process

Step 1: The user requests a web page.

Step 2: The tracking component first captures the web usage data associated with the initial request

Step 3: The tracking component forwards the request to the web application to be processed.

Step 4: The web application returns the produced web page.

Step 5: The web application returns the web page to the client.

It is clear from the above steps that there is no form tracking module involved in the tracking process of EDSF. The only data that is accumulated is the client's navigational data.

7.5.2 Pre-processing

Input: Raw web navigation usage data

Output: Clean and transaction-identified usage data

Pre-processing includes three tasks: data cleansing, transaction identification and dwell time completion.

7.5.2.1 Data cleaning

Data cleansing removes irrelevant web usage data including that of:

- researchers who monitor the forum,
- visitors who did not create a user account, and
- crawlers and other Web robots that are not spambots.

7.5.2.2 Dwell time completion

Dwell time is defined as the amount of time a user spends on a specific web page in his/her navigation sequence. It can be calculated by looking at each record timestamp. Dwell time is defined as:

$$d = t_{i+1} - t_i \quad where\ 1 < i < |S| \tag{7.1}$$

d is a dwell time for i^{th} requested web page in session *S* at time *t*;

In E.q. 7.1, it is not possible to calculate the dwell time for the last visited web page in a session. For example, a user navigates to the last web page on the website then closes his/her web browser. In such cases, the average dwell time spent on other web pages during the same session is considered.

7.5.2.3 Transaction identification

In transaction identification, a meaningful cluster of user navigation data in extracted. Web usage data is grouped into three levels of abstraction: IP, User and Session. The highest level of abstraction is IP and each IP address in web usage data may consist of multiple users. The middle level is the user level where each user can have multiple browsing sessions. Finally, the lowest level is the session level which contains detailed information of how the user behaved for each website visited.

In the EDSF, transaction identification is done at the session level for the following reasons:

- The session level can be built and analysed quickly while other levels need more tracking time to obtain a complete view of the user's behaviour.
- The session level provides more in-depth information about user behaviour when compared with the other levels of abstraction.

A transaction is defined as a set of web pages that a user requests during each browsing session.

Figure 7.3 presents a flowchart for the pre-processing of the EDSF. The procedure of the pre-processing part is as follows.

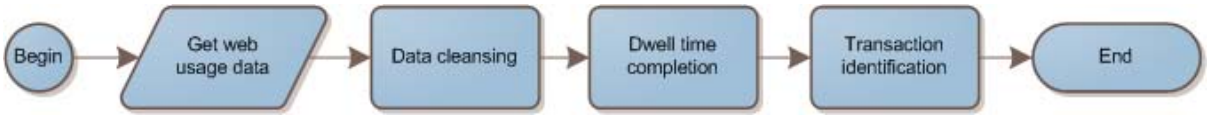


Figure 7.3: The flow chart diagram for the pre-processing part of EDSF

Step 1: The pre-processing tasks begin with raw web navigational data.

Step 2: Unwanted data is wiped out during data cleansing.

Step 3: The dwell time for each web page is calculated.

Step 4: The data is grouped based on session transaction levels and is passed to the next stage.

7.5.3 Feature Measurement

Feature measurement extracts and measures three feature sets: Action Set, Action Time, and Action Frequency from web usage data to build input vectors for classification. These features are measures

as described in Section 6.2.3. However, the features are used inside the classifier for detection of Spam 2.0 behaviour.

Figure 7.4 presents the flowchart for the feature measurement of EDSF. The process consists of the following steps.

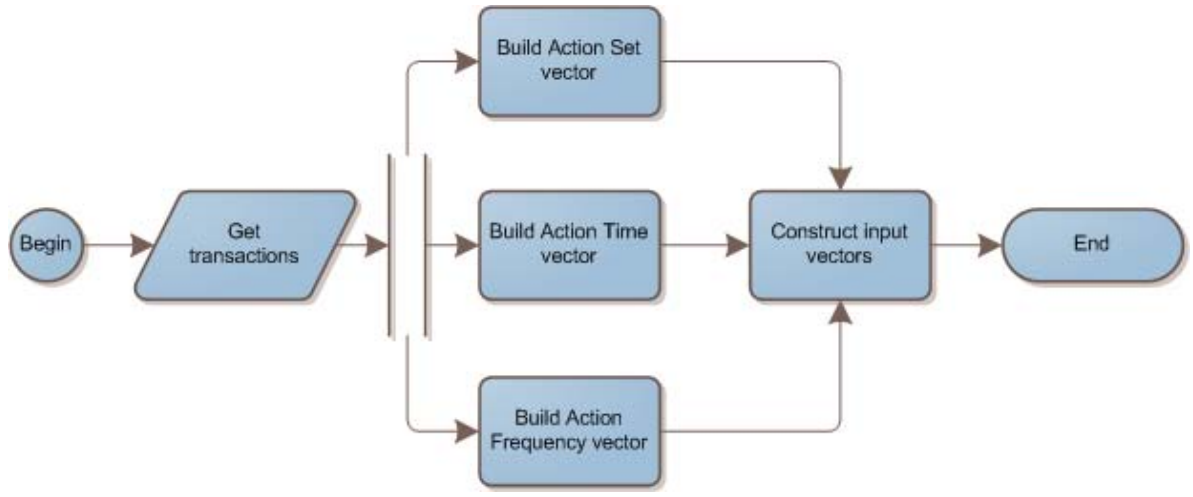


Figure 7.4: Flowchart for feature measurement part of EDSF

Step1: Obtain the transaction set from the previous part of the EDSF.

Step2: Calculate Action Set, Action Time, and Action Frequency for each transaction.

Step3: Construct input vectors based on the features. The input vectors are used in the next part of EDSF.

7.5.4 Classification

A Support Vector Machine (SVM) is used as the machine learning classifier in EDSF. SVM is a machine learning algorithm designed to be robust for classification, especially binary classification (C. Chang and C. Lin 2001).

Given a set of input data, SVM predicts the class/category to which the input belongs. Similar to other supervised machine learning classifiers, SVM requires a set of training examples. SVM builds up a model based on the training example. The model is used to predict a new example category. The SVM model generates a *hyperplane* in finite/infinite dimensional space that is a margin between different categories. This hyperplane is used to separate classes.

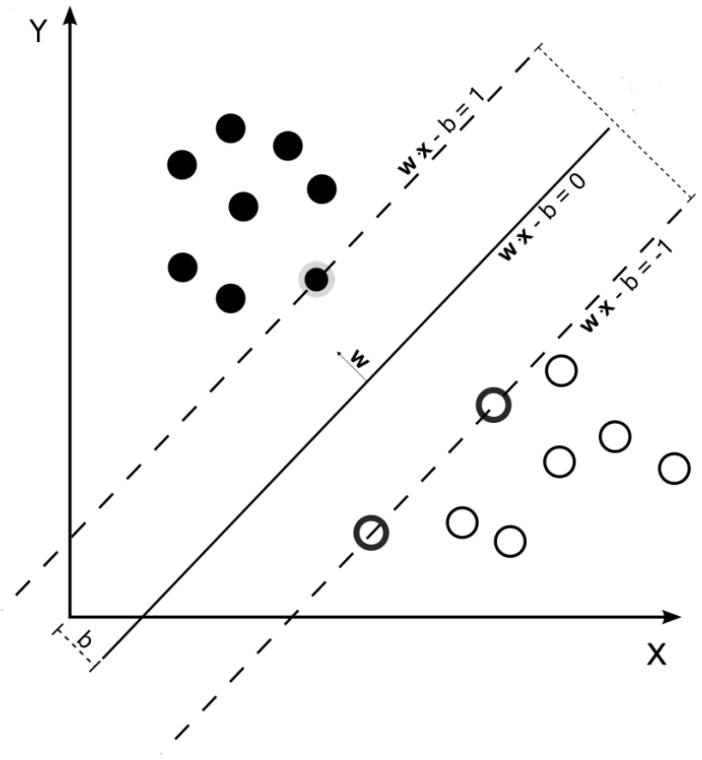


Figure 7.5: SVM decision function

SVM trains by *n* data points or features $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and each feature comes with class label y_i. In the spam filtering tasks there are two classes {*human*, *spammer*}, that can be assigned numerical values of -1 and +1 to each class accordingly. SVM then tries to find an optimum hyperplane to separate the two classes and maximise the margin between each class. A decision function on new data point *x* is defined in E.q. 7.2 where *w* is the weight vector and *b* is the bias term.

$$\phi(x) = sgn(w, x + b) \qquad 7.2$$

As illustrated in Figure 7.5, there are two classes of data (black dots and white dots) that have been separated with a line (hyperplace in high dimensional spaces). For each new example $\phi(x)$ assign a class.

Figure 7.6 presents the flowchart for the classification of EDSF. The process consists of the following steps.

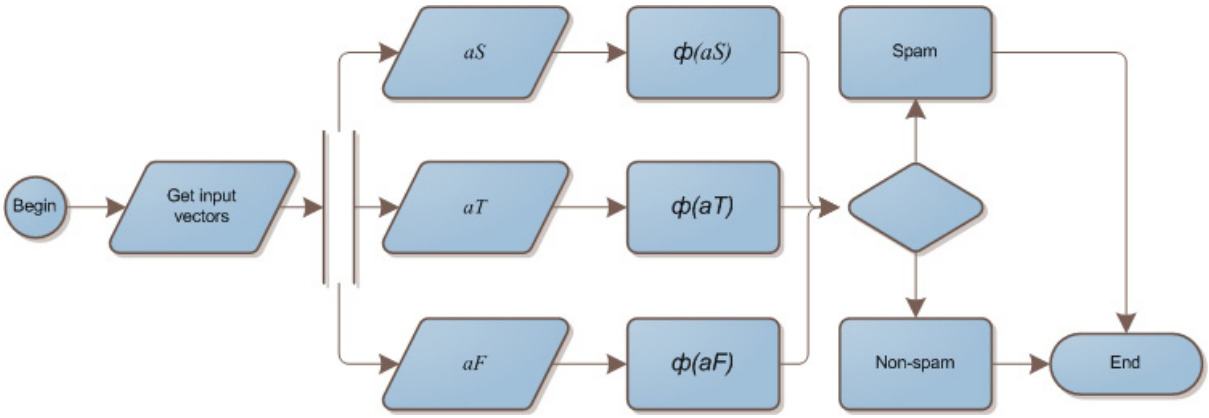


Figure 7.6: Classification part of EDSF

Step 1: Obtain the input vectors from the pre-processing part.

Step 2: Select a set of input vectors as a training set for the classifier, and use the remainder as a validation set.

Step 3: Train the classifier based on each training set.

Step 3: Run the classifier based on each input vector – Action Set (*aS*), Action Time (*aF*), Action Frequency (*aF*).

Step 4: Generate results based on the outcome of the classifier.

This concludes the *third* stage of conceptual process where the algorithm for EDSF has been proposed. The following section describes the prototype implementation of EDSF.

7.6 Prototype Implementation

The *fourth* stage of the conceptual process is where the theoretical foundation is implemented as a prototype. In order to fulfil the requirements (Section 7.3) the following technologies are used to implement the prototype.

Tracking

Similar to the tracking part of HoneySpam 2.0 described in Section 5.6.1, PHP programming language and a MySQL database are used to implement the tracking part of EDSF. However, as

described previously (Section 6.5.1), form usage data is not monitored by EDSF; hence, there is no prototype implementation for the form tracking component.

The tracking part of EDSF can implicitly gather web usage data and is extendable to all web applications.

Pre-processing

Pre-processing part is implemented using PHP language. The outcome of this is stored in the MySQL database which can be used later for feature measurement.

Feature Measurement

The feature measurement of the EDSF is implemented using PHP language. Once the input vectors are constructed, the outcome is stored in a specific file format to be used inside the classifier.

Classification

For the classification, *Waikato Environment for Knowledge Analysis (WEKA)* is used. Weka is a collection of machine learning algorithms including SVM (Hall et al. 2009). It is developed in Java programming language.

7.6.1 Tracking part

Similar to the tracking part of HoneySpam 2.0 (Section 5.6.1), the EDSF tracking part is modelled using 2 PHP file. The form tracking PHP files are not included here.

- sb_unb.php
- sb_mysql.php

The PHP files are executed on the virtual web server operating with Linux (Suse Server Enterprise 11), Apache server and PHP compiler version 5.

The description of *sb_unb.php* and *sb_mysql.php* are same as described in Section 5.6.1.1

7.6.2 Pre-processing part

Server-side programming is utilised to process the data. All the data is accumulated on the server (MySQL database). Due to these time-consuming tasks, PHP command line script language is used to pre-process the data.

Pre-processing includes the following tasks:

- Cleansing the data
- Calculation of dwell time
- Identification of transactions

The sections below detail the above tasks.

7.6.2.1 Data cleansing

SQL queries similar to Section 5.6.2.1 have been constructed for this task. The SQL queries are used to:

- remove redundant data,
- remove visitors' data,
- remove the data of researchers who monitor the system,
- remove standard and known robot data,
- remove data for javascripts, image, stylesheet etc., file requests.

7.6.2.2 Dwell time completion

The following algorithm (Code 7.1) is used to calculate the dwell time for each user. The average dwell time is used for the last visited web page in a session.

Code 7.1: Dwell time algorithm

```
Algorithm dwell_time_completion(S)
    input: set of webpages in a session, S
    output: set of dwell_time for a session, S
1.   for each w in S
2.    if i = |S|
3.        dwell_time = (total session time) / |S|
4.    else
5.      dwell_time = w_{i+1}.time - w_i.time;
6.    endif
7.   repeat
8.  return dwell_time
```

The algorithm starts by iterating through all the web pages in the session *S*. The calculation of dwell time is on line 5, where the time visit (timestamp) for the web page is deducted from the time visit for the next visited web page in the session.

Line 2 checks whether the web page is the last web page in the session. If so, on line 3 the dwell time for the last seen web page is calculated by measuring the average dwell time for other web pages in the session.

The algorithm returns the set *dwell_time* on line 8.

7.6.2.3 Transaction identification

For the transaction identification task, an SQL statement similar to that in Section 6.5.1 is constructed to identify session level transactions. The following SQL *where clause* statement is used to group transactions at the session level.

***GROUP BY** (session)*

Since all records have session ids, the above SQL statement can identify all the records that belong to a specific session. Transaction identification in EDSF generates a list of available session ids which can be used later to extract information from each session.

7.6.3 Feature measurement

Three feature sets – Action Set, Action Time and Action Frequency – are measured in this part. These features are used later to construct input vectors to be used inside the classifier. PHP command line scripts are utilised to measure each feature set.

The description of each input vector is discussed in Section 6.5.5.

7.6.4 Classification

To distinguish Spam 2.0 behaviour from genuine human behaviour, a machine learning classifier known as SVM is employed. A well-known classifier machine learning classifier WEKA (Hall et al. 2009) is used for classification. WEKA is implemented in the Java programming language. WEKA can be used either through its Graphical User Interface (GUI) or through API from other Java codes.

The *LibSVM* classifier function in WEKA is used to implement SVM. The LibSVM function is implementation of Chih-Chung Chang and Chih-Jen Lin class library for SVM (Chih-Chung Chang and Chih-Jen Lin 2001). LibSVM includes the following features:

- Multi-classifier
- Different version of SVM formulation
- Cross validation
- Probability estimation
- Implemented in various programming languages, etc.

The input vector is formatted into ARFF format, which is a standard format for WEKA. The ARFF file contains two parts: header and data. The header begins with @RELATION keyword and contains information including:

- name of the relation,
- a list of the attributes (columns), and
- each attribute type.

The data part starts with @DATA. Each record is on one line and each attribute inside each record is separated by a comma. Figure 7.7 presents an example of ARFF file.

```
@RELATION EDSF
   @ATTRIBUTE ActionA  NUMERIC
   @ATTRIBUTE ActionB  NUMERIC
   @ATTRIBUTE ActionC  NUMERIC
@DATA
   1,0,1
   0,0,1
   0,1,0
```

Figure 7.7: Sample ARFF file

The algorithm for the classifier is presented in Code 7.2. If the result of the classification for a given user *i* is 1, the user would be marked as spam and all the user's content contribution would be classified as spam content. Otherwise, the user is a genuine human user.

Code 7.2: Algorithm for the classifier

```
Algorithm classification(Vi)
    input: input vector for a the given user, Vi
    output: the boolean classification result (1 = spam user), Ri
1.  Ri = Φ(Vi)
2.  if Ri = 1
3.    Mark i as spam user
4.    Mark all i's posts as spam
5.  else
6.    Mark i as the genuine human user
7.  return Ri
```

This concludes stage *four* of the conceptual process. The next section focuses on experimental settings where the proposed prototype is tested.

7.7 Experimental Settings

The experimental settings comprise the *fifth* stage of the conceptual process where the proposed prototype is tested and examined. The EDSF experiment involves three processes: Data Collection, Parameter Specification, and Performance Measurement.

Data collection explains the dataset that is used to conduct the experiment. It includes both Spam 2.0 and genuine human data.

Parameter Specification is a discussion about all the parameters and their associated values used for gathering the results. The parameters that are used in the machine learning classifier are also discussed here.

Performance Measurement discusses the method which is used for measuring the performance of EDSF under different parameter settings and datasets.

7.7.1 Data collection

As mentioned previously, there are no publicly available Spam 2.0 data collections which contain both human and spam web usage data for Web 2.0 platforms. Hence, the dataset for this experiment is collected using the proposed HoneySpam 2.0 tool. The dataset contains both Spam 2.0 and genuine human behaviour. The Spam 2.0 dataset is collected through HSCOM1 (Section 5.7.1.1) over a period of one month. This data is collected from the online discussion board (i.e. SMF) running on HSCOM1. The genuine users' data is collected from a live forum with the same configuration as HSCOM1. However, the following concerns are taken into the account when using the genuine users' and Spam 2.0 data for the EDSF experiment.

- *Same web application*: The same version of SMF web application is run on the genuine users' forum.
- *Same site structure*: Both genuine users' and spam forums have the same site structure (e.g. sitemap).
- *Same tracking tool*: The same version of HoneySpam 2.0 is run on the genuine users' forum.
- *Same type of data*: The web usage navigational data is collected over a period of one month for the genuine users' forum which is the same as Spam 2.0 data.
- *Neutralised data*: Both genuine users' and spam data are eliminated from any possible biased information. For example, domain specific information such as the URL name is removed from the genuine users' data.

The genuine users' data is frequently moderated by manual investigators; hence, there is no tracking of spam activity in this data.

In order to further validate the performance of the EDSF, another experiment is conducted on a different dataset. The dataset is accumulated through a live forum that is used by both genuine and spam users. The result of this experiment is discussed in Section 7.9.1.

Table 7.1 presents a summary of the collected data. The dataset contains 16,594 entries consisting of 11,039 spam records and 5,555 genuine users' records.

Table 7.1: Summary of the dataset for the EDSF experiment

Type of data	Frequency
No. of genuine users' records	5,555
No. of spam records	11,039

7.7.2 Parameter Specification

All the parameters and their associated values used are listed in Table 7.2. It also includes the parameter settings for the SVM classifier.

Table 7.2: Parameters and their associated values used in the EDSF

Parameter	Value	Description
No. of transaction , |T|	4,227	Result of transaction identification at session level during pre-processing part.
No. of spam transactions	3,726	Manually classified spam transactions
No. of non-spam transactions	501	Manually classified non-spam transactions
No. of Actions , |A|	34	Result of feature measurement part for different type of Actions.
SVM type	C-SVC	C-SVC is a standard SVM algorithm.
Kernel type	Radial basis function	$exp(-\gamma\|\|x - c_i\|\|^2);\ \gamma = \frac{1}{2\sigma^2} > 0$
Degree	3	Degree of the closeness in the kernel function
Gamma	0.029	1 / number of features
Cost	1	To create soft margin between classes and allow training error. It might result in misclassification.
Epsilon	0.001	Tolerance of termination factor.
Cross-validation	10	To observe the efficiency of the classifier on different training – test (unseen) data.

Table 7.3 lists 34 Actions that are extracted and selected from the dataset. The details of each Action are explained in the table. These Actions can be extendable to other web applications, specifically online discussion boards.

Table 7.3: The list of all the extracted Actions

Action Key	Description
1	View homepage
2	View all the discussion boards
3	View topics in one discussion board
4	View topic (view user own created topic)
5	View overall forum statistic
6	View recently submitted topics
7	View search result
8	View user profile information
9	View other users profile
10	View personal message (Inbox)
11	View members lists
12	View latest reply to a specific topic
13	View help
14	View all the posts by specific user
15	Modify user own profile (user account details)
16	Modify user own profile (personal information)
17	Modify user own profile (change website look-and-feel)
18	Modify user own profile (change notification settings)
19	Modify user own profile (change personal messaging system settings)
20	Use search facility
21	Preview the topic before posting
22	Start new poll
23	Start new topic
24	Reply to topic
25	User authentication (Login)
26	Send personal message to other users
27	Modify personal message
28	Quote other users posts
29	Spell check
30	Find other users in system
31	Use *password forgot* facility to remember the password
32	Edit user own topic/reply
33	Mark all the topics in the system as read
34	Update user own topic/reply

The dataset is split into two sections: training and test. Two-thirds of the data (2,818 records) is used for the training set and the remaining 1,409 records are used as a test set. Each input vector has 34 features and input vectors separately are experimented with using the SVM classifier. Based on the above parameter specifications, three experiments are conducted.

7.7.3 Performance Measurement

7.7.3.1 F1-measure

A common performance measurement technique known as Precision, Recall and F1-measure is used to calculate the performance of the EDSF through each of the three experiments.

As mentioned in Section 2.2.2, precision (E.q. 7.3) refers to the exactness, while recall (E.q. 7.4) shows completeness of the result.

$$P = \frac{tp}{tp+fp} \quad 7.3$$

$$R = \frac{tp}{tp+fn} \quad 7.4$$

It is common to combine precision and recall in order to generate one value (known as *F-Measure*) to compare or evaluate different classification results. F-measure or *F1-measure* is defined as below.

$$F_1 = 2\frac{P \times R}{P+R} \quad 7.5$$

Where P and R are precision and recall values respectively.

If the outcome of these three measures is close to 1, this indicates better system performance. If the resulting value is closer to 0.5, then the output of the system is similar to random prediction. Finally, if the resulting value is closer to 0, this shows the strong inverse ability of the classifier.

7.7.3.2 Confusion Matrix

Confusion matrix is a method used to visually represent actual and predicted classification results (Kohavi and Provost 1998). By evaluating this matrix, the performance of the classifier is investigated. Figure 7.8 presents a sample confusion matrix.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	a	b
	Positive	c	d

Figure 7.8: Sample confusion matrix

In the figure, *a* is the number of correctly classified negative instances, *b* is the number of incorrectly classified negative instances, *c* is the number of incorrectly classified positive instances and finally, *d* is the number of correctly classified positive instances.

This concludes stage *four* of the conceptual process. The following section focuses on the results and observations of the experiments for the proposed prototype.

7.8 Results and Observations

This section discusses the results that were observed after running the proposed prototype i.e. EDSF. This is the *sixth* stage of the conceptual process.

7.8.1 Experiment 1 (Action Set)

The summary of the experiment on Action Set input vector is presented in Table 7.4. This experiment achieved the true-positive rate (99%). The false-positive rate for genuine users is less than 0.002%. However, the average F1 value for the whole experiment is 95%.

Table 7.4: Summary of experimental results for Action Time, *aS*, feature set

	TP	FP	P	R	F
Spam	0.999	0.323	0.958	0.999	0.979
Non-Spam	0.677	0.00002	0.997	0.677	0.806
Average	0. 961	0.285	0.963	0.961	0.958

The confusion matrix corresponding to the first experiment is presented in Table 7.5.

Table 7.5: Confusion matrix for the experiment on Action Set

	Spam	Non-Spam
Spam	3,725	1
Non-Spam	162	339

It is clear from the table that only one spam user is classified as a non-spam user. However, 162 non-spam users are classified as spam users.

7.8.2 Experiment 2 (Action Time)

The result of the second experiment is presented in Table 7.6. The false-positive value for the wrongly classified genuine users is about 2%. The average F1 value for the whole experiment is 92%.

Table 7.6: Summary of experimental results for Action Time, *aT*, feature set

	TP	FP	P	R	F
Spam	0.976	0.399	0.948	0.976	0.962
Non-Spam	0.601	0.024	0.774	0.601	0.676
Average	0.932	0.355	0.927	0.932	0.928

The confusion matrix corresponding to the experiment on Action Time is presented in Table 7.7.

Table 7.7: Confusion matrix for experiment on Action Time

	Spam	Non-Spam
Spam	3,638	88
Non-Spam	200	301

The number of wrongly classified spam users is 88 and the number of genuine users wrongly classified as spam is 200.

7.8.3 Experiment 3 (Action Frequency)

Table 7.8 shows the results of the experiment on Action Frequency. The best average F1, 0.96 for the whole experiment is achieved in this experiment. However, the percentage of wrongly classified genuine users is 0.2%.

Table 7.8: Summary of experimental results for Action Frequency, *aF*, feature set

	TP	FP	P	R	F
Spam	0.998	0.299	0.961	0.998	0.979
Non-Spam	0.701	0.002	0.975	0.701	0.815
Average	0.962	0.264	0.963	0.962	0.960

The confusion matrix for the third experiment is presented in Table 7.9.

Table 7.9: Confusion matrix for the experiment on Action Frequency

	Spam	Non-Spam
Spam	3,717	9
Non-Spam	150	351

The number of wrongly classified spam users is 9. The number of wrongly classified genuine human users is 150.

This section examines the results after experimentation with the proposed prototype. This concludes stage *six* of the conceptual process and paves the way for the next stage.

7.9 Validation and Comparative Analysis

EDSF achieved an average accuracy of 94.70%, which ranges from 93.18% for Action Time to 96.23% for Action Frequency. For the first experiment on Action Set, the number of correctly classified instances is 4,064 and the number of incorrectly classified instances is 163. The number of correctly classified instances for the second and third experiments is 3,939 and 4,068 respectively. The number of incorrectly classified instance for the last two experiments is 288 and 159 respectively.

Although the first and third experiments have the same number of correctly classified instances, the number of incorrectly classified instances for the third experiment is lower than for the first experiment. Hence, it results in better performance of the classifier. The better performance seen in the Action Frequency experiment proves that spammers tend to repeat certain tasks more often when

compared with the genuine humans who perform a larger variety of tasks. Given this result, it is not surprising that spammers utilise automated tools to distribute spam content. These tools are machine coded and designed to perform similar Actions over a period of time.

EDSF performance draws heavily on Spam 2.0 distribution behaviour. Although spam users' routines that are coded inside their automated tools can be changed over time, it occurs at the cost of slower spam distribution.

One of the drawbacks of the EDSF is the significant number of false-positives (the number of incorrect classified genuine users). Wrongly classified spam users are less annoying than incorrectly classified genuine human users. The reasons for these false-positives are as follows.

- The nature of EDSF does not make a distinction between fair spam Actions; hence, EDSF marks genuine users with quite an amount of fair spam Actions as the spam user.
- EDSF investigates only the behaviour of the users at the session level. Although session level transactions give a detailed profile of the users' visits, they do not provide an overall view of the users' behaviour.
- Although Actions are good discriminative features for classification, some Actions might be similar for both genuine and spam users. This increases misclassifications in the results. For example, a genuine user which directly inputs the "creating new thread" URL could be wrongly classified as a spam user since spam users mainly aim for content contribution web pages rather than navigating through other web pages.
- The feature sets – Action Set, Action Time, and Action Frequency – that are used inside EDSF are not capable of modelling the sequence and order of the user navigation through the web platform. This order can provide more information about the characteristics and nature of web usage within Web 2.0 platforms which may result in better classification results. The focus of the next chapter is on incorporating the sequence of requested webpages into Spam 2.0 filtering.

Given that there is no work in the literature on the classification of Spam 2.0 behaviour, the results of EDSF are the initial steps toward identifying Spam 2.0 by utilising web usage behaviour.

7.9.1 Comparative Analysis

In this section, the proposed method is compared with existing solutions from the literature. As mentioned earlier, there are no studies on Spam 2.0 identification using web usage. However, there are some similar works in the literature to detect Spam 2.0 using mainly content-based features**.** The comparison of these methods with the proposed method is discussed here.

There are three well-known approaches for detecting Spam 2.0 including *Akismet*, *Defensio*, and *TypePad*. Additionally, CAPTCHA as a prevention-based Spam 2.0 filtering method is investigated.

All approaches have been implemented and examined against HSCOM6 (Sections 5.7.1.3). An online discussion board is run on HSCOM6. The data accumulated from HSCOM6 includes both spam and genuine users' forum posts and web usage data. Therefore, HSCOM6 data is quite suitable for comparative analysis of different spam filtering methods.

The summary of HSCOM6 data is depicted in Table 7.10.

Table 7.10: Summary of the collected data on HSCOM6

Item	Value
Running time	21/5/10 – 17/2/11
No. of spam posts	3,698
No. of non-spam posts	274
No. of spam users' navigation	42,336
No. of non-spam users' navigation	138,394

The following is a detailed description of each Spam 2.0 filtering approach.

7.9.1.1 Akismet

Akismet is a detection/prevention-based Spam 2.0 filtering service that was originally built for blog comment spam filtering (Akismet 2011). Akismet provides a public API Spam 2.0 filtering service. Developers can integrate the Akismet spam filtering service in their application by using this API. To date, Akismet has been implemented for most of the Web 2.0 platforms including blog, comment systems, forums, wikis, content management systems (CMS).

Akismet mainly identifies a spam post as distinct from a genuine one by examining the content features of the post. The following information must be sent to the Akismet spam filtering service for classification:

- Author: the author of the submitted content such as the username of a user who posts a reply in a forum.
- Author's email: the email address of the author of the content.
- User Agent: the user agent field of the Internet browser submitting the content.
- IP address: the IP address of the author of the content.
- Body: the content of the post whether it be a comment, forum reply, wiki page etc.

Other information such as Referrer URL, type of content, name of author are optional. Akismet also blacklists known spam traffic including spam users' IP addresses for further usage.

For comparative analysis, Akismet API is implemented in PHP language. The entire API's required information such as author, author email address, User Agent, IP address, and body of content are provided to Akismet.

The initial test on Akismet shows comparatively good classification results. However, the main reason for this result was due to the time difference between the data accumulation and the experiment and the fact that Akismet had already blacklisted the spammers IP addresses. Therefore, another experiment is conducted with modified IP addresses. The IP addresses of all the requests to the Akismet service have been set to a single IP address which has not been blacklisted. In this experiment setting, it is possible to evaluate the performance of the Akismet service by simulating real-world scenarios where spammers' IP addresses are not blacklisted.

The confusion matrix for the experiment on Akismet Spam 2.0 filtering service is presented in Table 7.11.

Table 7.11: Confusion matrix correspond to Akismet experiment

	Spam	Non-Spam
Spam	3,526	172
Non-Spam	4	270

The overall accuracy that is achieved is 0.95 with 0.98 precision, 0.61 recall values. The F1-measure value corresponding to the Akismet experiment is 0.75.

7.9.1.2 Typepad AntiSpam

Typepad AntiSpam is another detection/prevention-based Spam 2.0 filtering method (TypePad AntiSpam 2011). It was originally designed to block spam comments on blogs. Typepad AntiSpam provides publicly available API for Spam 2.0 filtering. Its API is fully compatible with Akismet API and it can also be expanded to different Web 2.0 platforms.

The experiment setting for Typepad AntiSpam was kept the same as the Akismet experiment because of their compatibility. Table 7.12 presents a confusion matrix for the result of the experiment with an Typepad AntiSpam service.

Table 7.12: Confusion matrix for Typepad AntiSpam experiment

	Spam	Non-Spam
Spam	3651	47
Non-Spam	14	260

The following results were achieved during the Typepad AntiSpam experiment: accuracy 0.98, precision 0.94, recall 0.84 and F1-measure 0.89.

7.9.1.3 Defensio

Defensio is another detection/prevention based Spam 2.0 filtering service (Defensio 2011). It provides a public API Spam 2.0 filtering service to be integrated on Web 2.0 platforms.

To conduct the experiment, the following information is required to be passed to the Defensio Spam 2.0 filtering servers:

- Platform: Type of Web 2.0 platform that is analysed such as *phpbb*, *wordpress* (blog system), *drupal* (CMS) etc.
- Type of content: Type of content that is sent to be analysed which may be comment, wiki, forum etc.
- Content: the main body of the content that needs to be analysed.
- Client: the name, version and author name and author email address of the program that is used for querying Defensio API.

A program is coded in PHP language to implement Defensio API and conduct the experiment. All the above-mentioned information is mandatory and is passed to Defensio API. The confusion matrix corresponding to the Defensio experiment is presented in Table 7.13.

Table 7.13: Confusion matrix for Defensio experiment

	Spam	Non-Spam
Spam	3,454	244
Non-Spam	15	259

The overall accuracy for the Defensio experiment is 0.93, precision 0.94, recall 0.51 and F1-measure 0.66.

7.9.1.4 CAPTCHA

As mentioned in Chapter 2, CAPTCHA is the most well-known prevention-based spam filtering method preventing automated-generated requests on web platforms (von Ahn et al. 2003). It has been widely used for Web 2.0 platforms such as forums, social networks, comment systems etc. Once a user intends to perform specific actions on web platforms (e.g. register a user account, post a comment), CAPTHA can be utilised to ensure that the request has not originated from an automated code. CAPTCHA has three main steps to ascertain the identity of users (i.e. whether it is a real human user or machine code) as following:

- Step 1: A challenge is generated in the server and sent to the user to solve.
- Step 2: User tries to solve the challenge and returns the response.
- Step 3: The server checks and grades the user's response and provides privileges to perform the user requested action.

The main part of CAPTCHA is the design of the challenge. CAPTCHA challenges which are not well designed might allow a machine code to bypass the restriction, while complex CAPTCHA challenges might keep human users out of the system.

In the CAPTCHA experiment, the default CAPTCHA that comes with the latest version of the *phpBB* forum platform is used. In this experiment setting, all requests for creating a new user account are protected with CAPTCHA. Figure 7.9 illustrates an example of CAPTCHA used in the experiment.

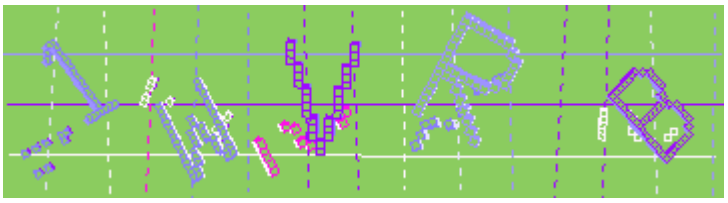


Figure 7.9: Sample CAPTCHA used for the experiment

CAPTCHA was used for the entire running duration of HSCOM6. All the attempts to create new user accounts (i.e. visiting the user registration page of the forum) and overcome CAPTCHA were recorded. Hence, a figure can be provided that measures the performance of CAPTCHA. All the requests from genuine users to solve CAPTCHA on HSCOM6 are removed from the data. Additionally, all the requests coming through known web crawlers are also removed from the data.

For the period of the HSCOM6 experiment, of the 4,159 visits to the user registration page, 171 spam user accounts could successfully bypass CAPTCHA and register a user account in the forum. Hence, about 95.88% of requests are blocked by CAPTCHA. However, this figure might not represent the real performance of CAPTCHA since:

- There may be some attempts to visit the registration page of the forum, but not necessarily to create a user account. Such attempts are included in the above performance measurements. Therefore, the real performance of CAPTCHA might be lower than the abovementioned figure.

- The inconvenience of the CAPTCHA challenges could not be measured. Although complex CAPTCHA challenges might block out most of the spammers, it would also keep real users out of the system, resulting in a higher number of false-positives.

However, the above figure can give an estimated number of the performance value of CAPTCHA compared to other Spam 2.0 filtering methods.

7.9.1.5 EDSF

In order to equally compare the results of the proposed method, with other Spam 2.0 filtering methods, an experiment is conducted on HSCOM6 data to measure the performance of EDSF. Table 7.14 provides the summary of the parameters and their associated value in the EDSF.

Table 7.14: Parameters and their associated values for the EDSF experiment

Parameter	Value	Description
No. of transaction , |T|	6,285	Result of transaction identification at session level during pre-processing part.
No. of spam transactions	6,230	Manually classified spam transactions
No. of non-spam transactions	55	Manually classified non-spam transactions
No. of Actions , |A|	29	Result of feature measurement part for different type of Actions.
SVM type	C-SVC	C-SVC is a standard SVM algorithm.
Kernel type	Linear	$x\ c_i$
Degree	3	Degree of the closeness in the kernel function
Gamma	0.029	1 / number of features
Cost	1	To create soft margin between classes and allows training error. It might result in misclassification.
Epsilon	0.001	Tolerance of termination factor.
Cross-validation	10	To observer the efficiency of the classifier on different training – test (unseen) data.

Table 7.15 presents 29 Actions that are used in the experiment. Each Action is a feature inside the input vector.

Table 7.15: Name and description of each Action.

Action Key	Description
1	Visit homepage
2	Visit user control panel
3	Login
4	Edit profile
5	Edit avatar
6	Manage draft
7	Logout
8	Edit signature
9	Manage subscriptions
10	Manage bookmarks
11	Compose message
12	Edit account settings
13	Edit global settings
14	Manage friends
15	Solve CAPTCHA
16	Delete cookies
17	Register
18	View members
19	View profile
20	Find member
21	Post topic
22	Post reply
23	Quote post
24	Edit post
25	View forum
26	View topic
27	View FAQ
28	Search
29	View feed

Considering that the best performance results were achieved in the Action Set and Action Frequency experiments in Section 7.8, two experiments – Action Set and Action Frequency - are conducted. Action Set and Action Frequency are used as feature sets to build the input vectors for the SVM

classifier. Table 7.16 and Table 7.17 provide the confusion matrices for Action Set and Action Frequency on HSCOM6 data.

Table 7.16: Confusion matrix for Action Set experiment

	Spam	Non-Spam
Spam	6,226	4
Non-Spam	13	42

Table 7.17: Confusion matrix for Action Frequency experiment

	Spam	Non-Spam
Spam	6,228	2
Non-Spam	18	37

The average precision, recall, and F1-measure of both experiments using Action Set are respectively 0.95, 0.88 and 0.91 while for the Action Frequency experiment these values are 0.97, 0.83 and 0.89. A summary of the comparative analysis of the abovementioned methods is presented and discussed in the next section.

7.9.1.6 Summary

The five experiments conducted with Akismet, Typepad AntiSpam, Defensio, Captcha and the proposed EDSF, yielded the following results, as presented in Table 7.18.

Table 7.18: Summary of the EDSF comparison analysis

	Akismet	Typepad AntiSpam	Defensio	CAPTCHA	EDSF (aS, aF)
F1-Measure	0.75	0.89	0.66	-	0.91, 0.89
Accuracy	0.95	0.98	0.93	0.95	0.99, 0.99
False-positive rate	0.04	0.01	0.06	-	0.23, 0.32
False-negative rate	0.01	0.05	0.06	-	0.001, 0.0003
Strategy	Detection/ Prevention-based	Detection/ Prevention-based	Detection/ Prevention-based	Prevention-based	Early-detection based
Language Dependency	Yes	Yes	Yes	No	No

The overall F1-measure value of the EDSF is higher than the state-of-the-art Spam 2.0 filtering methods. It shows that the proposed method performs better in regards to Spam 2.0 filtering. The other advantage of the EDSF is its detection strategy. EDSF is an early-detection based method; hence, it can detect spammers before content distribution. Akismet, Typepad AntiSpam and Defensio are mainly content-based, so such methods are not able to prevent spam content from being submitted.

7.10 Conclusion

The early-detection based Spam 2.0 filtering methods attempt to identify spam behaviour prior to or just after spam content submission. Such methods investigate the behaviour of spam users, e.g. automated tools, spambots etc.

The proposed EDSF method in this dissertation automatically distinguishes Spam 2.0 submission from genuine human content contribution on Web 2.0 platforms. The proposed method studies, investigates and identifies Spam 2.0 on a Web 2.0 platform. EDSF makes use of web usage navigation behaviour to build up a discriminative feature set including Action Set, Action Time, and Action Frequency. Such a feature set is later used in a machine learning classifier known as a Support Vector Machine (SVM) for classification.

The experiments illustrated in this dissertation collected data from the HoneySpam 2.0 framework as well as a live moderated forum. The F1-measure is used to determine the performance of the EDSF.

The EDSF achieved an average accuracy of 94.70%, which ranges from 93.18% for Action Time to 96.23% for Action Frequency. For the first experiment on the Action Set, the number of correct classified instances is 4,064 and the number of incorrect classified instances is 163. The number of correct classified instances for the second and third experiments is 3,939 and 4,068 respectively. The number of incorrect classified instances for the last two experiments is 288 and 159 respectively.

The overall performance of EDSF was compared against four other state-of-the-art Spam 2.0 filtering methods. The overall F1-measure value of EDSF is higher than those of existing Spam 2.0 filtering methods. It shows that the proposed method has better performance in regard to Spam 2.0 filtering. The other advantage of EDSF is its detection strategy. EDSF is an early-detection based method and therefore can detect spam users before content distribution.

Given that there is no work in the literature on Spam 2.0 behaviour classification, the results of EDSF are the initial steps toward identification of Spam 2.0 by utilising web usage behaviour. EDSF outperforms current state-of-the-art Spam 2.0 filtering methods for the detection and prevention of spam users on Web 2.0 platforms.

The following chapter describes a novel method for on-the-fly Spam 2.0 filtering.

Chapter 8

On-the-Fly Spam 2.0 Filtering

This chapter presents:

- An introduction to On-the-Fly Spam 2.0 Filtering.
- A framework for detecting Spam 2.0 by analysing web navigation sequence data
- A proposed Action String, a novel feature set for Spam 2.0 detection.
- The evaluation, validation and comparative study of the proposed framework performance with real-world data.

8.1 Introduction

This chapter presents a method for **On-the-Fly early-detection based Spam 2.0 filtering** called (**OFSF**). Early-detection based Spam 2.0 filtering methods strive to identify spam behaviour prior to or after spam content submission.

OFSF automatically distinguishes Spam 2.0 submissions from genuine human content contributions inside Web 2.0 platforms. Similar to the EDSF that is discussed in Chapter 7, OFSF makes use of web usage navigation behaviour to create discriminative feature sets. However, OFSF considers the sequence and order of user web navigation in order to build up a feature vector for classification. Unlike EDSF, OFSF uses a novel rule-based classifier to store, retrieve, and separate spam users from genuine human ones. OFSF makes it possible to spot spam users on-the-fly. While users are interacting with the Web 2.0 platform, OFSF is able to track and identify possible spam navigation patterns.

To handle, i.e. store, retrieve, update, and look up navigation data, OFSF uses a *Trie* data structure. Trie is in the form of tree structured data where each node stores an identifier of the user's web navigation Action while the tree edges contain the order and sequence of Actions.

The theoretical foundation for OFSF, along with its algorithm design which addresses all the requirements specified in Chapter 4, is discussed here. Finally, testing, experimentation and validation are discussed at the end of this chapter.

8.2 General Overview of OFSF

This research continues from the previous work, i.e. EDSF for combating Spam 2.0. The main idea is to actively investigate spam users' behaviour – as an origin of Spam 2.0 instead of seeking discriminate attributes inside spam content.

The viewpoint of OFSF is identical to EDSF. However, OFSF orders the users' Actions in order to distinguish spam users from genuine human ones. Due to the design constraints, the sequence of users' Actions is not included in EDSF.

Figure 8.1 illustrates a sample series of HTTP requests sent from both genuine human users and spam users to submit new content to an online forum ("Post new topic" Action).

1	123.123.123.123 - - [10/Jul/2009:00:19:25 +0800] "GET **/forum/index.php?action=post;board=1** HTTP/1.0" 200 7852 "http://example.com/forum/index.php?action=post;board=1" "Opera/9.0 (Windows NT 5.1; U; en)"
2	123.123.123.123 - - [10/Jul/2009:00:19:25 +0800] "GET **/forum/index.php?action=post;board=1** HTTP/1.0" 200 7852 "http://example.com/forum/index.php?action=post;board=1" "Opera/9.0 (Windows NT 5.1; U; en)"
3	123.123.123.123 - - [10/Jul/2009:00:19:25 +0800] "GET **/forum/index.php?action=post;board=1** HTTP/1.0" 200 7852 "http://example.com/forum/index.php?action=post;board=1" "Opera/9.0 (Windows NT 5.1; U; en)"
4	123.123.123.123 - - [10/Jul/2009:00:19:30 +0800] "GET **/forum/index.php?board=1** HTTP/1.0" 200 6248 "http://example.com/forum/index.php?board=1" "Opera/9.0 (Windows NT 5.1; U; en)"

(a) Sequence of spammer HTTP header requests

1	111.111.111.111 - - [15/Aug/2009:07:05:10 +0800] "GET **/forum/index.php** HTTP/1.0" 200 60 "http://example.com/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
2	111.111.111.111 - - [15/Aug/2009:07:06:12 +0800] "GET **/forum/index.php?board=1** HTTP/1.0" 200 6248 "http://example.com/forum/index.php" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
3	111.111.111.111 - - [15/Aug/2009:07:08:32 +0800] "GET **/forum/index.php?action=post;board=1** HTTP/1.0" 200 7852 "http://example.com/forum/index.php?board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"
4	111.111.111.111 - - [15/Aug/2009:07:10:27 +0800] "GET **/forum/index.php?topic=1397** HTTP/1.0" 200 524 "http://example.com/forum/index.php? action=post;board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

(b) Sequence of genuine human HTTP header requests

Figure 8.1: Series of spam users' HTTP requests (a) vs. genuine human HTTP requests (b) in an online forum to create a new topic

Each HTTP entry consists of an IP address, timestamp, request method, the requested URL, the response status code, the referrer URL and the User Agent. There are some obvious differences between spam users' and genuine humans' requests including:

- The genuine human's sequence of HTTP header requests is more expected or on average. For example, Figure 8.1a shows the normal way that a website is navigated. It starts with users browsing the main page of the forum (line 1), visiting one of the boards (line 2), clicking on the "Post new topic" button to visit *create new topic* webpage (line 3) and finally visiting their own created topic (line 4).

- The spam user order of the HTTP header request does not show an expected sequence. For example, in Figure 8.1b, spam users' navigation starts by directly requesting to *create new topic* (line 1) and repeating the same request two more times (lines 2 and 3). Finally, it finishes by viewing one of the forum boards (line 4).

This factor shows that spam users follow pre-programmed and optimised procedures. Therefore, such key differences in web usage behaviour can be utilised for identification of spam users.

In order to formulate the sequence of Actions and use it for identification of spam users, one possible solution is to assign an identifier (e.g. a unique character) to each Action and build an input vector by putting together these identifiers in the same sequence as they occur. However, in order to use such input vectors in a machine learning classifier, the following problems would arise.

- The number of features is not limited: For example, user A performs 3 Actions in a given transaction, the input vector length for A is 3; user B performs 5 Actions in the same transaction, the input vector length for B is 5. If the feature set is limited to 3, the last two Actions of B would be omitted; while if the feature set is set to 5, A's input vector would contain missing values (Figure 8.2). This problem becomes more significant when the length of the users' input vector substantially varies.

- The number of input vectors is high: Since the sequence of users' Actions is an important criterion for classification, a slight change in the order of Actions would result in a new input vector.

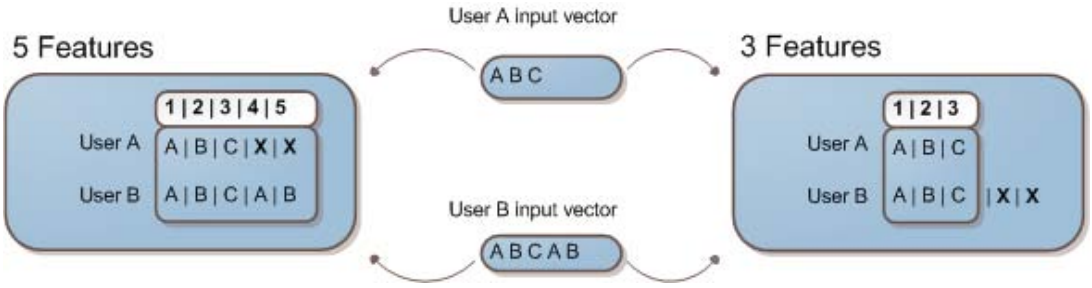


Figure 8.2: The problem of input vector length for machine learning classifiers

To address the problems above, OFSF introduces a new feature set called Action String and a novel rule-based classifier to formulate and distinguish spam users from genuine human users.

The abovementioned problems also highlight a drawback of the EDSF design - its inability to include sequence of Action.

8.2.1 Action String

Action String is a sequence of users' Actions in a given transaction where each Action is identified by an identifier, i.e. alphabet. By placing each Action identifier together in the same order that the user performs the Action, an Action String is generated.

The main advantage of Action String compared to other feature sets – Action Set, Action Time, and Action Frequency – is that it maintains the order of users' Actions. This information can reveal more accurate characteristics of spam or genuine human users' behaviour on the Web 2.0 platform. The following scenario illustrates an advantage of Action Strings.

Suppose user *A* navigates through "View main page", "View board", "Post new topic", and "View main page" Actions and spends 5, 5, 10, and 5 seconds on each Action respectively. Spam user *S* navigates through "Post new topic", "View board", "View main page", and "View main page" Actions in the same span of time. Assume the total number of Actions is 7. To formulate A and S navigational behaviour using Action Set, Action Time, and Action Frequency, the following input vectors can be built up.

- Action Set input vector. A: (1,1,1,0,0,0,0), S: (1,1,1,0,0,0,0),
- Action Time input vector. A:(5,5,10,0,0,0,0), S:(5,5,10,0,0,0,0),
- Action Frequency input vector. A: (2,1,1,0,0,0), S: (2,1,1,0,0,0).

Clearly, there is no difference between the A and S input vectors although the order of performed Actions is different.

Having the same assumption, Action Strings for the A and S are created as follows. Each Action is identified by a unique letter.

- Action String: A: "abca" S: "cbaa"

This example shows that Action String can clearly differentiate between A and S web navigational behaviour.

OFSF makes use of Action String in order to formulate users' web navigational behaviour. Next, OFSF employs a rule-based classifier to differentiate spam users from genuine human users.

8.3 Requirements

The following requirements are specified for the proposed solution. This represents the *first* stage of the conceptual process where requirements are elicited and prioritised. The requirements are as follows.

7. **Extendibility**: the method should be extendable to all Web 2.0 platforms.
8. **Adaptability**: the method should be adaptable to new Spam 2.0 behaviours.
9. **Automaticity**: the method should automatically differentiate Spam 2.0 from genuine human behaviour.
10. **Independency**: the method should be independent of its platform. For example, it should be language- and content-independent.
11. **Implicitness**: the method should implicitly accumulate behavioural.
12. **Discriminative features**: the features should be discriminative enough to differentiate genuine human user behaviour from spam users' behaviour.
13. **Chronological Order**: the order and sequence of users' web navigational behaviour should be clear.
14. **Spontaneity**: the method should differentiate spam users from genuine human users on-the-fly.

The following section defines the design rationale based on the abovementioned requirements. A design decision for each requirement is made in the design rationale along with a discussion of how the requirement is met.

8.4 Design Rational

This section provides a design rationale to meet all the requirements that were listed in the previous section. This is the *second* stage of the conceptual process. The following design decisions are proposed to address each requirement.

5. Develop an early-detection based Spam 2.0 filtering method that is language- and platform-independent (Req. 4).

6. Capture server side web usage data while the user interacts with the website for implicit data accumulation (Req. 5) and extendibility to all Web 2.0 platforms (Req. 1).

7. Develop an automated rule-based classifier that can be trained with new instances of Spam 2.0 (Req. 2) and automatically identify spam users (Req. 3).

8. Formulate web usage data into the Action String to build a feature set that both identifies spam users (Req. 6) and maintains the sequence of performed Actions (Req. 7).

9. Use a tree-based data structure (i.e. Trie) to spontaneously store, retrieve, update, and identify spam navigational patterns (Req. 8).

In the development of any solution, the above design decisions should be considered. All these design decisions are included in the proposed method to detect Spam 2.0 behaviour on web applications. This concludes stage two of the conceptual process. The next stage is the establishment of the theoretical foundation for the proposed solution.

8.5 Theoretical Foundation

In this section, the theoretical foundation for the OFSF is discussed, the design decisions are analysed, and an algorithm is proposed. The theoretical foundation as depicted in Figure 8.3 consists of five main parts: tracking, pre-processing, feature measurement, classifier construction, and classification.

Tracking

Records web usage data including the user's IP address, username, requested webpage URL, session identity, and timestamp.

Pre-processing

Data cleansing, transaction identification and Action identification.

Feature Measurement

Extracts and measures the Action String feature set from the web usage data based on each Action identifier.

Classifier Construction

Populates the rule-based classifier based on Action Strings and prepares the classifier for classification.

Classification

A rule-based classifier based on Trie data structure is used here to distinguish Spam 2.0 from genuine human behaviour.

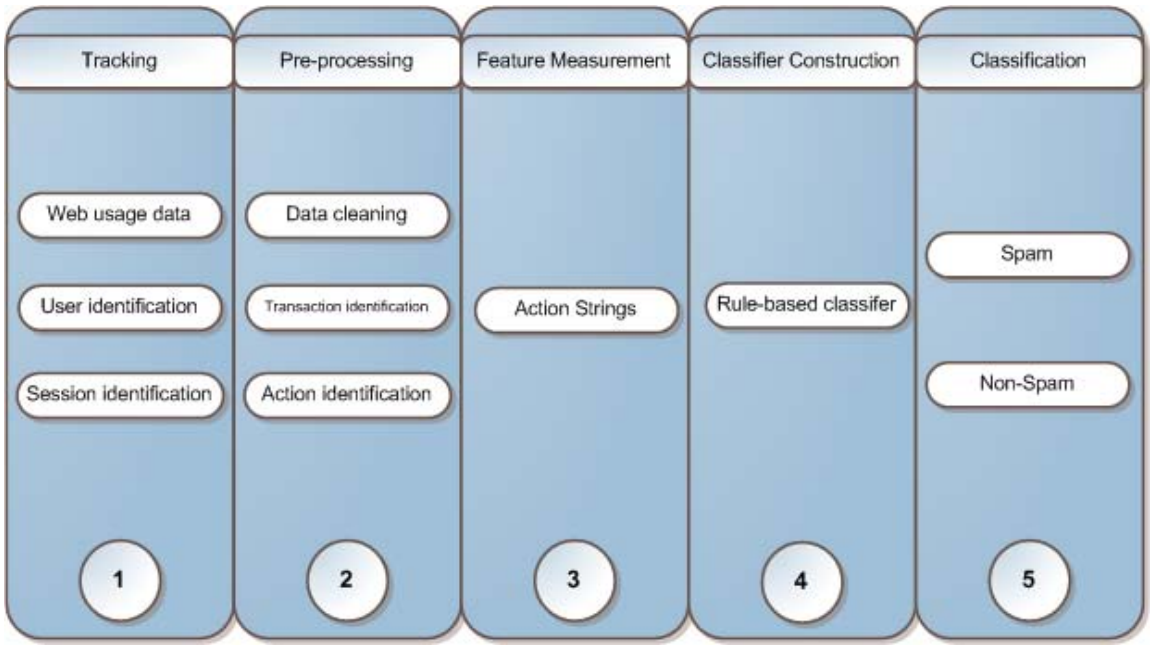


Figure 8.3: The overall framework for OFSF

The proposed solution explained above addresses the requirements described in Section 8.3. Below is a detailed description of each part of OFSF.

8.5.1 Tracking

Input: Incoming traffic

Output: User and session identified web usage data

Tracking monitors web usage data and the navigation stream. For each incoming HTTP request, IP address, visited URL, referrer URL, browser identity, timestamp, session ID and user account name are tracked.

8.5.2 Pre-processing

Input: User and session identified web usage data

Output: cleanse and transaction identified usage data

This part includes three tasks, which are data cleansing, transaction identification and Action identification.

8.5.2.1 Data cleaning

Once data is aggregated through Tracking, it needs to be cleansed from irrelevant information such as visitors who did not create a user account in the system, researchers who monitor the forum, and genuine and known web robots.

8.5.2.2 Transaction identification

Next, data needs to be grouped into meaningful user navigation clusters. This task is known as Transaction Identification. Each tracking record is accompanied by a unique session ID. In OFSF data is grouped based on session IDs. Therefore, requests made by a user during a single visit are clearly identifiable. Session level transactions make it possible to track user requests at each visit. Moreover, defining transactions at the session level can be utilised for fast and on-the-fly classification as well as provide in-depth insights into the users behaviour. Comparatively, in other transaction levels (e.g. IP and User) a longer time span is required to group user behavioural data.

8.5.2.3 Action identification

Based on a predefined set of Action, Action identification tasks extract Actions from users' transaction. The predefined Action set needs to be created for different types of tasks that the user can accomplish on a given Web 2.0 platform. The majority of Actions are general among many Web 2.0 platforms. A summary of such Actions are presented in Table 8.1.

Table 8.1: Summary of common Actions on Web 2.0 platforms

Action Name	Description
Registration	Create a user account
Login/Logout	Login and logout using pre-registered user account
Submitting content	Adding new content as a topic/comment/reply/etc.
Searching	Search the Web 2.0 platform for specific information
Viewing content	Reading articles/posts/comments/etc

For each Action, a unique identifier such as a character is assigned. This identifier is later used to construct the feature set, i.e. Action String.

Figure 8.4 presents a flowchart for the pre-processing part of the OFSF. The process of the pre-processing part is as follows.

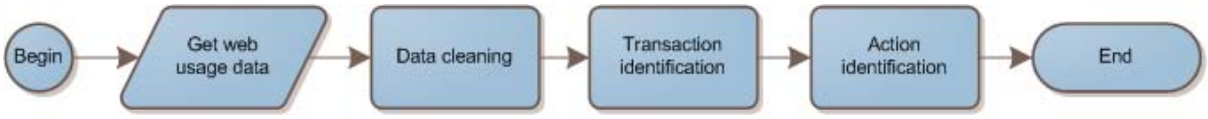


Figure 8.4: Flowchart for Pre-processing part of OFSF

Step 1: The pre-processing tasks start by getting user- and session-identified web usage data.

Step 2: Unneeded data is wiped out during data cleaning task.

Step 3: The data is grouped based on the session transaction level.

Step 4: Actions are extracted from user transactions based on a predefined Action set. Additionally, each Action would be assigned with a unique character.

8.5.3 Feature Measurement

Input: Cleansed and transaction-identified usage data

Output: Action Strings

This part extracts and measures the feature set Action String from users' transactions to build input vectors for classification. Action String is a sequence of users Actions in a given transaction where

each Action is identified by a character, i.e. alphabet. By placing each Action identifier together in the same order that the user performs the Actions, an Action String is generated.

Given a set of web pages $W = \{w_1, w_2, \ldots, w_{|W|}\}$, A is defined as a set of Actions, such that

$$A = \{a_i | a_i \subset W\} = \{\{w_l, \ldots w_k\}\} \quad 1 \leq l, k \leq |w| \tag{8.1}$$

Respectively, aSt_i is defined as

$$aSt_i = (a_j) \quad 1 \leq i \leq |T|;\ 1 \leq j \leq |A| \tag{8.2}$$

aSt_i refers to a *sequence* of Actions (called Action Strings) performed in a given transaction i and T is total number of transactions.

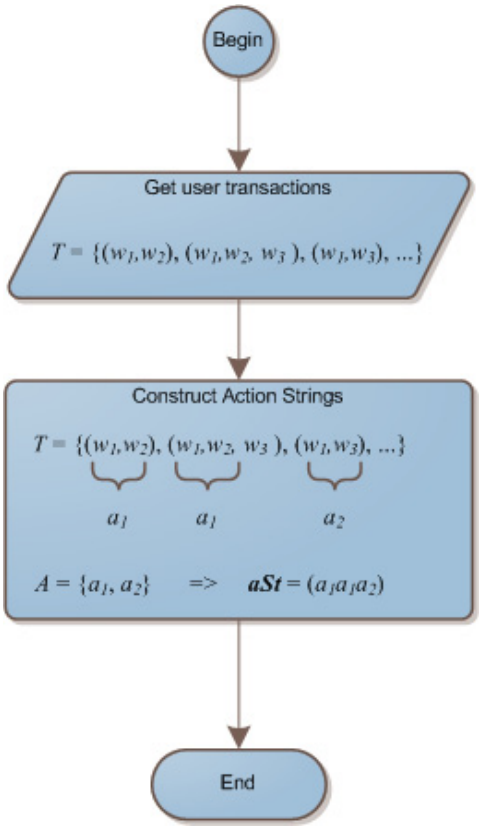


Figure 8.5: The flowchart for feature measurement part of OFSF

Figure 8.5 illustrates the flowchart diagram for feature measurement part of OFSF. The process of this part is consists of the following steps.

Step 1: Get the users' transactions from the pre-processing part.

Step 2: Construct Action Strings (*aSt*) based on user set of Actions (*A*), user transactions (*T*) to be used in the next part of OFSF framework. Here, each action is represented by a_i identifier. *aSt* is constructed by placing a_i together in a sequence.

8.5.4 Construct Classifier

Input: Action Strings

Output: Trie data structure associated with probability of being spam and non-spam for each Action String

Once Action Strings have been created, OFSF constructs a rule-based classifier. The classifier needs to spontaneously identify spam users from non-spam ones. The OFSF classifier is based on Trie structure.

A Trie is a tree-based data structure to store and retrieve information (Edward 1960). Its main advantages are ease of updating, handling, and short access time. A Trie is in the form of a tree structure where each node contains the value of the key with which it is associated. The Trie does not store duplicates, so it is suitable for situations where there is redundancy in data, e.g. web navigational patterns.

OFSF stores the Action Strings in Trie data structure format. Each Trie edge contains an Action identifier (i.e. a character) and each node holds two probabilities – the probability of a specific Action String belonging to spam users and the probability of the Action String belonging to a non-spam users. These probabilities are measured through E.q. 8.3and E.q. 8.4.

$$P_S(s_i) = \frac{f_S(s_i)}{f_H(s_i)+f_S(s_i)} \quad 8.3$$

$$P_H(s_i) = \frac{f_H(s_i)}{f_H(s_i)+f_S(s_i)} \quad 8.4$$

where s_i is an Action String, $f_H(s_i)$ is the frequency of non-spam Action Strings that have prefix s_i. $f_S(s_i)$ is the frequency of spam with prefix s_i , $P_H(s_i)$ is the probability of s_i being non-spam and $P_S(s_i)$ of its being spam, respectively.

Zero probability means that a particular Action String has never occurred before. For example, Figure 8.6 illustrates a sample Trie data structure for a set of *{ABC, ABCD, ABDE, ABC}* Action Strings for a genuine human user and *{ABD, ABD, ABDE}* Action Strings for a spam user.

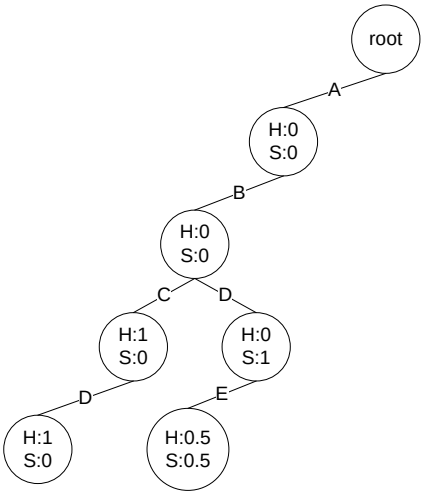


Figure 8.6: Sample Trie data structure for sets of genuine humans and for a spam users. S and H represent the probability of Action String belonging to spam and genuine human users respectively.

The main advantage of OFSF over EDSF is that EDSF requires users to complete their web navigation at a given transaction in order to classify users. However, OFSF can spontaneously classify users as they navigate through a Web 2.0 platform.

The Trie that is used as a data structure in OFSF classifier makes it possible to spontaneously look up for a specific pattern. Therefore, as a user navigates through the Web 2.0 platform, OFSF is capable of classifying the user based on the user's navigational behaviour.

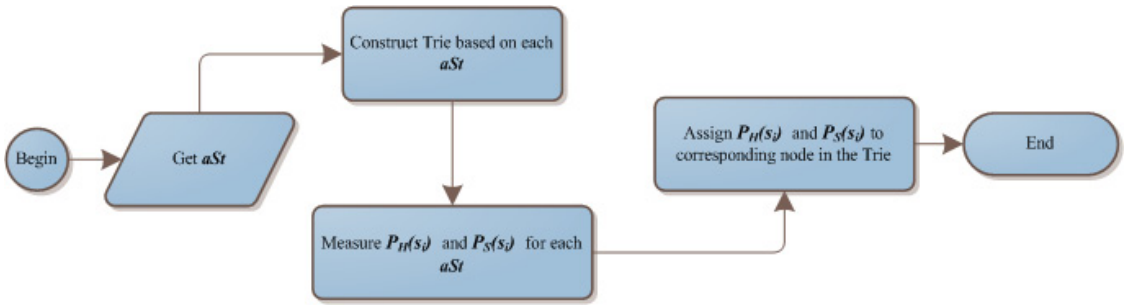


Figure 8.7: The flowchart of OFSF rule-based classifier

Figure 8.7 presents the flowchart of a OFSF rule-based classifier. The process consists of the following steps.

Step 1: Get Action Strings from previous part of OFSF.

Step 2: Build a Trie data structure based on each Action String.

Step 3: Measure $P_H(s_i)$ and $P_S(s_i)$ for each Action String.

Step 4: Assign above-measured probabilities to corresponding node in the Trie.

8.5.5 Classification

Input: Trie data structure associated with probability of being spam and non-spam for each Action String

Output: the classification result whether or not an Action String belongs to a spam user

New incoming Action String s_i is validated through Trie data structure that has been built in during previous part of OFSF (i.e. Construct classifier). Validation s_i can fall into two categories – *Matched* and *Not-Matched*. In the former case, our framework looks at $P_H(s_i)$ and if $1- P_H(s_i) > Threshold$, s_i , this sequence would be classified as a spam Action String. In the latter situation, no decision is made (refer to Section 8.9 for validation and discussion).

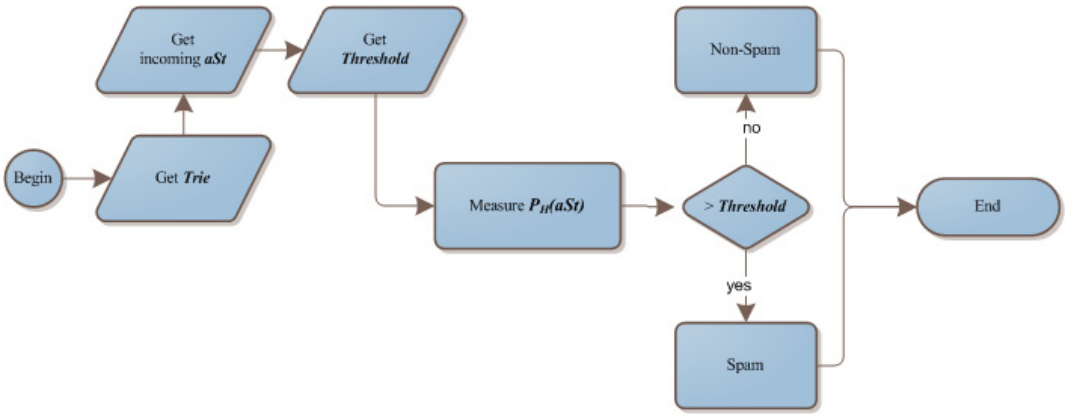


Figure 8.8: The flowchart for OFSF classification part

Figure 8.8 presents a flowchart for OFSF classification. This process consists of the following steps.

Step 1: Get Trie from previous part of OFSF.

Step 2: Get new incoming Action String.

Step 3: Get the *Threshold*.

Step 4: Measure $P_H(s_i)$ for new incoming Action String.

Step 5: If the probability is above that of the given threshold, Action String indicates a spam user.

8.6 Prototype Implementation

The *fourth* stage of the conceptual process is where the theoretical foundation is implemented as a prototype. In order to fulfil the requirements, the following technologies are used to implement the prototype.

Tracking part

Similarly to the tracking part of HoneySpam 2.0 described in Section 5.6.1, the PHP programming language, MySQL database is used to implement the tracking part of OFSF. However, form usage data is not monitored by OFSF. Hence, there is no prototype implementation for the form tracking component.

Pre-processing part

The pre-processing part is implemented using the PHP language. The outcome of this part is stored in a MySQL database which can be used later for feature measurement.

This part also assigns a unique identifier to each Action. These identifiers are used to build Action Strings.

Feature Measurement

The feature measurement part of the OFSF is implemented using PHP language. Once the input vectors are constructed, the outcome is stored in a MySQL table and used inside the classifier.

Construct Classifier

The rule-based classifier is implemented using Java language. This Java code retrieves Action Strings, and calculated the probabilities of spam activity and constructs the Trie data structure.

Classification

This classifier is used to generate the classification results. Additionally, output of the classification is stored in a specific file format to be used within a *Microsoft Excel Spreadsheet* to generate reports and figures.

8.6.1 Tracking part

Similar to the tracking part of HoneySpam 2.0 (Section 5.6.1), the OFSF tracking part is modelled using 2 PHP file. The form tracking PHP files are not included here.

The PHP files are executed on the virtual web server operating with Linux (Suse Server Enterprise 11), Apache server and PHP compiler version 5.

8.6.2 Pre-processing part

The pre-processing part includes the following tasks:

- Cleansing the data
- Identification of transactions
- Identification of Actions

The sections below provide detailed descriptions of the above tasks.

8.6.2.1 Data cleansing

SQL queries similar to those in Section 5.6.2.1 are constructed for this task. The SQL queries are used to:

- remove redundant data,
- remove visitors data,
- remove researchers data who monitor the system,
- remove standard and known robot data, and
- remove data for javascripts, image, stylesheet etc., file requests.

8.6.2.2 Transaction identification

For the transaction identification task, a similar SQL statement to that in Section 6.5.1 is constructed to identify session level transactions. Transactions identification in OFSF generates a list of available session ids which can later be used to extract information from each session.

8.6.2.3 Action identification

Based on a defined set of Actions, this task extracts relevant Actions from users' transactions and assigns a unique identifier (i.e. a character) to each Action. The following algorithm is used for Action identification task.

Code 8.1: Action identification algorithm

```
Algorithm action_identification(T)
    input: set of users' transactions, T
           a set of Actions, A'
    output: set of Actions performed by the user, A
1.  for each t in T and a in A'
2.    if t = a #if t subset of a
3.      assign an identifier to a
4.      add a to A
5.    end if
6.  repeat
7.  return A
```

Code 8.1 begins with obtaining a list of a user's transactions and defined set of Actions. On line 1, it iterates through each user's transaction and a defined set of Actions. Line 2, if a transaction is subset or equal to a defined Action, the Action is assigned an identifier (i.e. a character) and on line 4, a corresponding Action is added to user lists of Actions (*A*).

8.6.3 Feature measurement

The feature measurement part constructs an Action String feature set as a sequence of Actions that are performed by a user in a given transaction. Therefore, Action Strings not only contain the Actions performed by users, but also maintain the order or sequence of these Actions.

The following algorithm is used to construct Action Strings.

Code 8.2: Action String algorithm

```
Algorithm action_string(T,A)
    input: set of transaction by a given user, T
             set of Actions with their identifiers, A
    output: user set of Action Strings, aSt
1.  for each t in T and a in A
2.    if t = a
3.      add a to aSt
4.  return aSt
```

Code 8.2 starts by getting a list of user's transactions and performed Actions. On line 1, it iterates through each transaction and performed Action in their sets to define the order of the performed Action in the user's transaction. Once this match is found (Line 2), the algorithm adds an Action identifier to the sequence of user Action Strings (*aSt*). The algorithm returns the set of user's Action Strings on Line 4.

8.6.4 Construct classifier

A rule-based classifier based on the Trie data structure is constructed in this part of OFSF. This classifier is used in the classification part to identify spam users.

Code 8.3 illustrates the pseudo-code of this part.

Code 8.3: OFSF rule-based classifier construction

```
Algorithm classifier_construction(aSt_i)
    input: set of users Action Strings, aSt_i
    output: a rule-based classifier, Φ
1.  Φ = create an empty Trie
2.  string = null
3.  for each aSt in aSt_i
4.    for each a in aSt
5.      Φ = Trie.add(a,0) #Trie_add(key, values)
6.      string = string + a
7.      if string.length() = aSt.length()
8.        Φ = Trie_add(a, P_H(string), P_S(string))
9.      end if
10.   repeat
11. repeat
12. return Φ
```

The algorithm starts by creating an empty Trie with empty root. For each Action String in the set of all the user's Action Strings, the algorithm iterates through its Actions and adds the Action identifier

(e.g. character) to the Trie data structure. The `Trie.add()` obtains two arguments – key and values – for the key. If the key order (letter alphabetical order) is smaller than for the previous Trie key, the new node is added to the Trie left child; otherwise, the new node would be added to the Trie's right child. The value 0 is assigned for this node. Once the algorithm reaches the last Action in the Action String, it measures $P_H(string)$ and $P_S(string)$ for the Action String. These measures are added to the value of the last node. The algorithm returns the Trie structure with corresponding values on line 12.

8.6.5 Classification

Code 8.4 provides an overview of the OFSF classification part. Each user (u) has multiple Action Strings, and Action Strings contain a series of performed Actions (a_i). For every new incoming u, the algorithm examines the user's Action String against the rule-based classifier (ϕ). If the probability for that specific Action String, $1\text{-}P_S(Action\ String)$ is higher than a given threshold, the classifier marks the incoming u as a spam user.

Code 8.4: OFSF classification algorithm

```
Algorithm classification(u)
    input: incoming user transactions, u
           given threshold value, threshold
    output: whether user is spam or non-spam, result
1.  for each u
2.    aSt_u = action_string(u, action_identification(u))
3.    if 1 - P_s(aSt_u) > threshold
4.     result = 1
5.    else
6.    result = 0
7.  repeat
8.  return result
```

This concludes stage *four* of the conceptual process. The next section, experimental settings, is where the proposed prototype is tested.

8.7 Experimental Settings

The experimental settings constitute the *fifth* stage of the conceptual process where the proposed prototype is tested and examined. The OFSF experiment involves 3 parts as follows:

- Data Collection: explains the dataset that is used to conduct the experiment. It includes both Spam 2.0 and non-spam data.
- Parameter Specification: is a discussion about all parameters and their associated values used for gathering the result.
- Performance Measurement: discusses the method that is used for measuring the performance of OFSF.

8.7.1 Data collection

Since there is no publicly available dataset containing web usage data for spam and non-spam users, the dataset used in this experiment is the same as the dataset used for the EDSF experiment (Section 7.7.1). The summary of the dataset is presented in Table 8.2.

Table 8.2: Summary of the OFSF experiment dataset

Type of data	Frequency
No. of genuine users' records	5,555
No. of spam records	11,039

Additionally, another experiment is conducted for OFSF and is discussed in Section 8.9.1. The dataset for this experiment was collected from a live forum that is used by both genuine human users and spam users. This experiment further validates the performance of OFSF.

8.7.2 Parameter specification

All the parameters used in the OFSF experiment, together with their associated values, are listed in Table 8.3.

Table 8.3: Parameters and their associated values used in OFSF experiment

Parameter	Value	Description
No. of transaction , $\|T\|$	4,227	Result of transaction identification at session level during pre-processing part.
No. of spam transactions	3,726	Manually classified spam transactions
No. of non-spam transactions	501	Manually classified non-spam transactions
No. of Actions , $\|A\|$	34	Result of feature measurement part for different type of Actions.
Window sizes	2 to 10	The window size over test set of Action Strings. The length of the longest Action String in the dataset is 10.
Threshold	-0.05 to 0.05	The threshold value for classification.

Table 7.3 provides a description of 34 Actions that are extracted from the dataset.

To evaluate the effects on OFSF of different training and test sets, the dataset is randomly divided into five datasets (DS1 to DS5). About 65% of the data is used to construct the classifier (training set) and the remaining 35% is used for evaluation purposes (test set).

OFSF has real-time detection or on-the-fly detection capability. OFSF measures Action Strings as they occur. The aim is to identify the spam behaviour pattern in the least number of Actions.

For each Action String in the test set, the following steps are taken:

1. The window size is set to 2.
2. A *window* over test Action Strings is made.
3. Action String's length is limited to the size of the window. For example, if the length of an Action String is 5 and window size is 2, only the first 2 characters of the Action String are considered.
4. The rule-based classifier is run against the window size of the Action String derived from the last step to generate the classification results.
5. The window size is increased by one character.
6. Repeat above steps until maximum window size is reached.

The previous steps can simulate real-world practices where user Action Strings increases over time.

8.7.3 Performance measurement

Common performance measurement techniques known as confusion matrix, Precision, Recall and F1-measure are used to calculate the performance of the OFSF in the experiment. Moreover, a novel performance measurement method known as Matthews Correlation Coefficient (MCC) is used as well (Matthews 1975). MCC is one of the best performance measurement methods of binary classifications, especially when the data between two classes of data is not balanced (Baldi et al. 2000). It considers true and false positives and returns a value between -1 and +1. If the return value is closer to +1, the classification result is better and the decision can be considered to have greater certainty. However, if the result value is close to 0, this indicates that the output of the framework is similar to random prediction. A result value closer to -1 shows a strong inverse ability of the classifier. MCC is defined as follows:

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \qquad 8.5$$

In E.q. 8.5, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives. These values can be extracted from a confusion matrix.

This concludes stage *five* of the conceptual process. In the next section, the results and observations based on the proposed prototype experiments are presented.

8.8 Results and Observations

This section discusses the results that are observed after running OFSF. This is the *sixth* stage of the conceptual process.

Figure 8.9 illustrates the accuracy of OFSF for different window sizes on the five random datasets (DS1 – DS5).

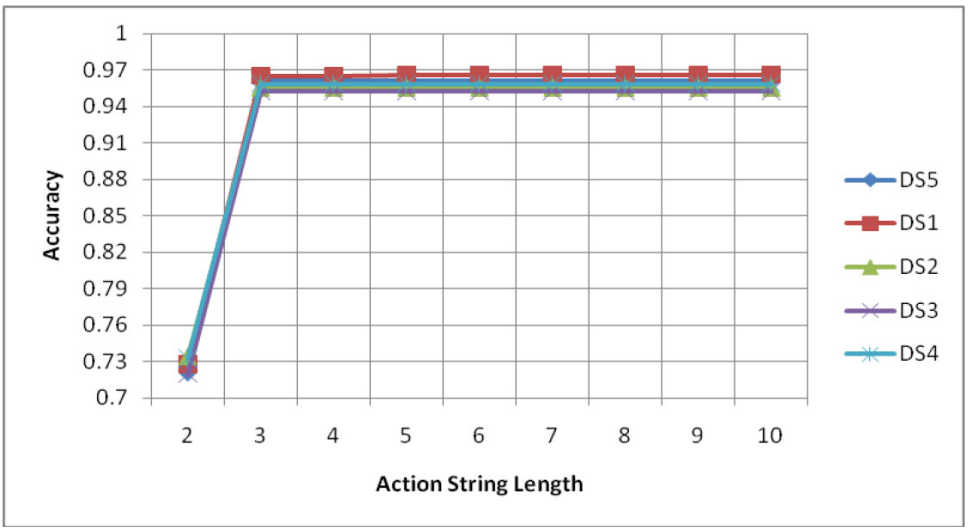


Figure 8.9: Average accuracy per dataset for different window sizes

Table 8.4 presents the average accuracy per dataset and accuracy per datasets for different window sizes.

Table 8.4: Overall average accuracy per dataset and for each different window size

	DS1	DS2	DS3	DS4	DS5
Accuracy	0.939	0.930	0.926	0.933	0.934
Len2	0.727	0.734	0.719	0.732	0.721
Len3	0.965	0.955	0.952	0.958	0.960
Len4	0.965	0.955	0.952	0.958	0.960
Len5+	0.966	0.955	0.952	0.958	0.960

The best accuracy of 96% was achieved on DS1 with *window size* equal to or larger than 5, while overall accuracy is 93% over all datasets. Figure 8.10 presents the MCC value for different window sizes. Overall, the average MCC on all datasets is 0.744. The best MCC value 0.808 was achieved on DS1 for a window size equal to or larger than 5.

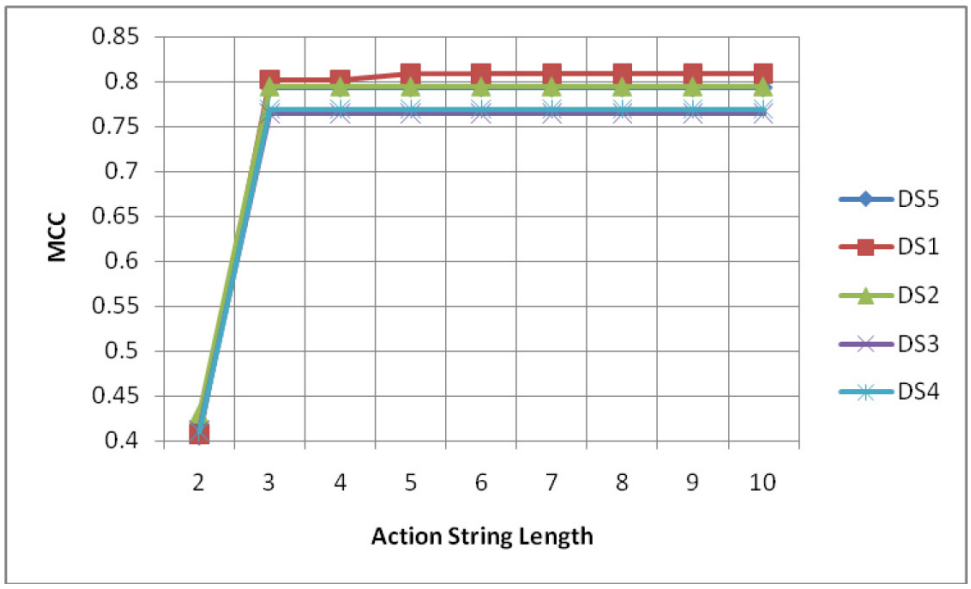


Figure 8.10: MCC value per different datasets on each window sizes

Table 8.5 presents the average MCC per dataset and MCC per dataset for different window sizes.

Table 8.5: The overall average MCC value per dataset and for each window size

	DS1	DS2	DS3	DS4	DS5
MCC	0.762	0.754	0.725	0.729	0.749
Len2	0.406	0.430	0.412	0.410	0.405
Len3	0.802	0.795	0.764	0.769	0.792
Len4	0.802	0.795	0.764	0.769	0.792
Len5+	0.808	0.795	0.764	0.769	0.792

Table 8.6 presents the average F1-measure for each dataset. The best F1 value 0.82 is achieved on DS1 with windows size equal or bigger than 5. The overall average F1 value in all datasets is 0.75.

Table 8.6: The average precision, recall and F1 measure value over all datasets

	DS1	DS2	DS3	DS4	DS5
Precision	0.91	0.96	0.92	0.89	0.92
Recall	0.69	0.64	0.62	0.65	0.66
F1 Measure	0.77	0.76	0.73	0.74	0.76

The threshold value for all the above experiments is set to zero. However, in order to investigate the effect of different thresholds on the OFSF performance, the threshold is changed from -0.05 to 0.05. Figure 8.11 illustrates the performance of OFSF on different thresholds for window sizes varying from 2 to 6 characters.

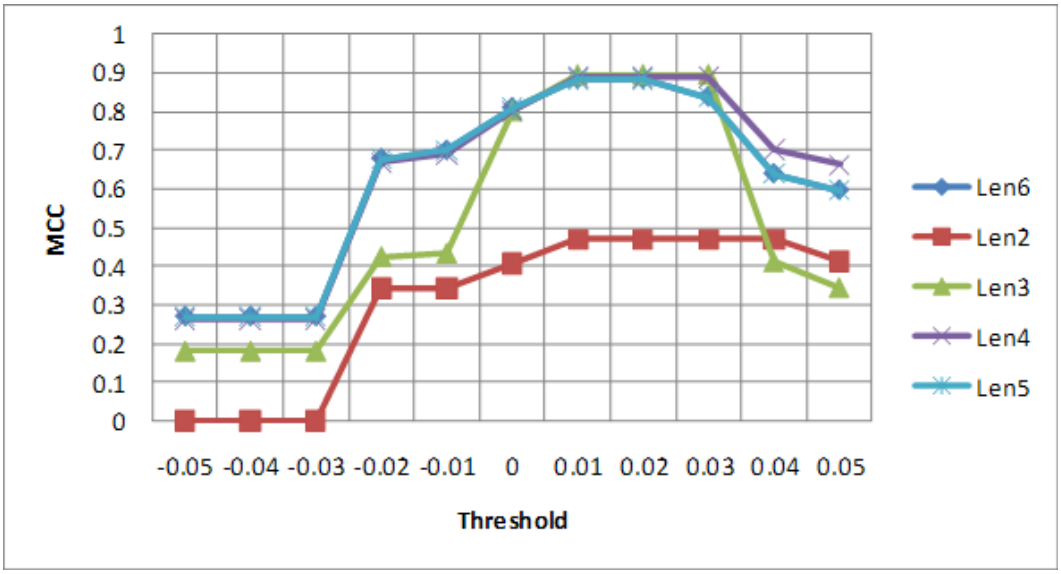


Figure 8.11: Performance of OFSF based on different thresholds for different window sizes (Len2 to Len6)

The best MCC value of 0.88 is achieved for thresholds between 0 and 0.03 for window size equal to or larger than 4 (Len4 to up).

8.9 Validation and Comparative Analysis

The overall average accuracy achieved in the OFSF experiment on 5 datasets is 93% where the best accuracy is 96% on window size equal to or greater than 5 characters. The best F1 value is 0.82 while the average F1 value is 0.75 on all the datasets. The average MCC value is 0.744 where the best MCC value is 0.808 for a window size equal to or greater than 5 characters.

The performance of OFSF becomes better as the window size increases. This shows that OFSF can make better predictions over time as the user uses the system. However, the performance remains the same after certain window sizes. The reason is that the datasets are randomly selected and the training sets do not contain all the Action Strings. Hence, some of the Action Strings are not inside the data structure of the rule-based classifier and OFSF cannot find a match for the incoming Action String. The same trend can be seen in the accuracy of the results, where accuracy remains the same after a

certain window size of 5. Regarding the threshold, it is obvious that moving the threshold toward negative values increases the number of false-negatives, while moving toward positive values increases the number of false positives. However, the best results can be achieved by maintaining the threshold between 0.0 – 0.03.

OFSF can be manually updated by including new Action Strings during Classifier Construction. However, current implementation of OFSF does not automatically update with new Action Strings. For example, if there is no match for the incoming Action String, the classifier does not produce any decision for the Action String. Another drawback of OFSF is that some Action Strings are the same for both genuine human users and spam ones. This situation produces false-positives and false-negatives because the current Action String considers only the web navigation behaviour of the users.

As future work, OFSF can be improved by considering the following:

- An adaptive mechanism to automatically update a classifier based on new unseen Action Strings.
- Include other usage behaviours in Action Strings such as time of each Action, existence of mouse / keyboard activity, etc in order to differentiate genuine human and spam Action Strings apart.

Overall, the average performance of OFSF compared to EDSF is lower. However, EDSF's better performance is achieved at the cost of time and computational resources. Additionally, in terms of on-the-fly identification of spam users, OFSF surpasses EDSF since the EDSF classifier cannot be trained with a partially completed input vector.

8.9.1 Comparative analysis

As mentioned previously, there have been no studies on Spam 2.0 identification by using web usage data. Therefore, to compare the proposed solution with similar solutions and best practices, OFSF is compared with four well-known approaches for detecting Spam 2.0 including Akismet, Defensio, TypePad, and CAPTCHA as well as EDSF.

The same experimental setting that was described in Section 7.9.1 is used here to conduct the comparative analysis. Hence, the HSCOM6 dataset with the same parameters is used.

The results of the experiment on Akismet, Defensio, TypePad, CAPTCHA and EDSF are presented in Section 7.9.1.

In the OFSF experiment, all the parameter settings are the same as those described in Section 8.7.2. However, all non-match Action Strings are considered as false-negative. False-negative is better than false-positive since, for example, it is far more annoying for a user to be blocked from a website than it is for a spam user to be allowed to submit spam content.

Table 8.7 presents the accuracy, F1 and MCC value of the OFSF experiment on HSCOM6 on window size 2 to 10.

Table 8.7: The result of OFSF experiment on HSCOM6

	Accuracy	MCC	F1 Measure
Len2	0.36	0.04	0.01
Len3	0.36	0.04	0.009
Len4	0.98	0.34	0.25
Len5	0.97	0.25	0.15
Len6	0.56	0.05	0.009
Len7	0.97	0.29	0.16
Len8	0.96	0.27	0.14
Len9	0.97	0.29	0.16
Len10	0.98	0.38	0.26
Average	0.79	0.22	0.12

The best accuracy 0.98, MCC 0.38 and F1 0.26 is achieved on window size 10. The average accuracy 0.79, MCC 0.22 and F1 0.12 is achieved for the whole experiment.

The performance of the system increases from window size 2 to 4. However, it drops for window size 5 to 6. The performance increases for window sizes 7 to 10. The behaviour is different from the previous experiment on OFSF, as the performance increases as the window size becomes longer. This

is because, unlike the previous experiment, in this experiment all the non-match Action Strings are considered as false-negatives. This assumption has a significant effect on the performance of the system. Table 8.8 shows the number of non-match Action Strings for each window size.

Table 8.8: The number of Non-Match Action Strings in OFSF experiment

	Len2	Len3	Len4	Len5	Len6	Len7	Len8	Len9	Len10
No Match	2816	3,965	69	133	2,720	168	209	186	108

It is clear from the table that on window size 6, the number of non-matched Actions is very high and this number is at its lowest value at window size 4. However, by deducting the number of non-matches from false-negative rate, the best accuracy 0.99, MCC 0.89 and F1 0.88 can be achieved on windows size equal to or greater than 5. This shows the same behaviour as the previous OFSF experiment described in Section 8.8.

In order to equally compare the result of OFSF with other existing solutions, the OFSF classifier is forced to produce a result even in the case of non-match Action Strings. All non-match Action Strings are considered as false-negatives.

8.9.1.1 Summary

Table 8.9 presents the summary of OFSF comparison analysis against four existing solutions as well as EDSF.

Table 8.9: The summary of OFSF comparison analysis

	Akismet	Typepad AntiSpam	Defensio	CAPTCHA	EDSF (*aS*, *aF*)	OFSF
F1-Measure	0.75	0.89	0.66	-	0.91, 0.89	0.12
Accuracy	0.95	0.98	0.93	0.95	0.99, 0.99	0.79
False-positive rate	0.01	0.05	0.05	-	0.23, 0.32	0.04
False-negative rate	0.04	0.01	0.06	-	0.001, 0.0003	0.20
Strategy	Detection/ Prevention-based	Detection/ Prevention-based	Detection/ Prevention-based	Prevention-based	Early-detection based	On-the-fly detection based
Language Dependency	Yes	Yes	Yes	No	No	No

Although the performance of OFSF is lower than that of other methods, the false-positive rate for OFSF is the second lowest among other methods. This gives OFSF an advantage where the cost of false-positives is high.

8.10 Conclusion

This chapter presents an On-the-Fly early-detection based Spam 2.0 filter called (OFSF). OFSF is capable of identifying spam users spontaneously as they use a Web 2.0 platform. OFSF uses a novel rule-based classifier based on the Trie data structure. This allows the OFSF feature to detect spam users on-the-fly and fast. OFSF exploits a new feature set known as Action String which is the sequence of a user's Actions in a given transaction where each Action is identified by a character, i.e. alphabet.

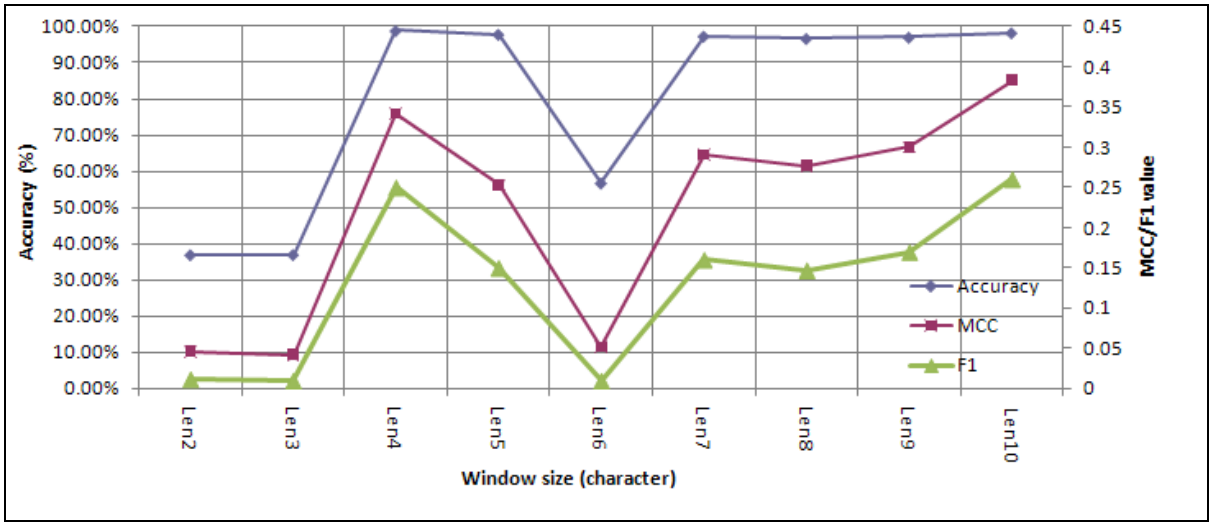


Figure 8.12: The performance of OFSF on HSCOM6

Figure 8.12 illustrates the performance of OFSF on HSCOM6 dataset. The overall performance of OFSF without considering non-match Action Strings is promising. By considering non-match Action Strings as false-negatives and forcing the OFSF rule-based classifier to produce results, the OFSF performance decreases. However, the comparative false-positive ratio is fairly low.

The following chapter concludes this dissertation.

Chapter 9

Conclusion and Future Work

This chapter covers:

- The current issues and problems with spam filtering.
- Solutions proposed by this dissertation to address spam filtering problems.
- Conclusion and future works.

9.1 Introduction

Web 2.0 is commonly associated with web applications that facilitate interactive information sharing, interoperability, user-centred design and collaboration on the World Wide Web. The read/write web 2.0 concepts are the backbone of the vast majority of web services that we use today. In contrast to non-interactive websites, Web 2.0 increasingly emphasises human collaboration through a participation architecture that encourages users to add value to web applications as they use them. Today, Web 2.0 functions are commonly found in web-based communities, applications, social-networking sites, media-sharing sites, wikis, blogs, mashups, and folksonomies. They are widely provided by government, public, private and personal entities.

New media introduce new ways of communicating that are not immune to abuse. A fake eye-catching profile in social networking websites, a promotional review, a response to thread in online forums with unsolicited content and a manipulated wiki page, are examples of new forms of spam in the web referred to as *Web 2.0 Spam* or *Spam 2.0*. Spam 2.0 is simply unsolicited content that is hosted on legitimate Web 2.0 platforms. As mentioned in Chapters 1 and 2, this new form of spam has produced new challenges for the spam filtering community.

Current literature highlights several key problems associated with Spam 2.0 filtering including the parasitic nature of Spam 2.0, user inconvenience associated with current anti-spam methods etc.

The main motivation of this research is to address the Spam 2.0 problem and to develop a framework for early-detection and prevention of Spam 2.0.

This chapter provides a summary of the issues and problems addressed by this study along its main contributions. The next section explains the main problems with which this dissertation was concerned.

9.2 Problems and Issues

In this dissertation, three main problems and challenges associated with Spam 2.0 filtering have been addressed. As mentioned in Chapter 3, these issues are:

- Problems with detection-based Spam 2.0 filtering methods.
- Problems with prevention-based Spam 2.0 filtering methods.
- Problems with early-detection based Spam 2.0 filtering methods.

9.2.1 Problems with detection-based Spam 2.0 filtering methods

As mentioned in Chapter 1, detection-based spam filters deal with the content of spam in order to identify spam patterns. Detection methods look for spam words, templates, attachments etc. inside the content.

However, the state-of-the-art, detection-based spam filtering approaches are very fragile. The main socio-economic and technical problems with existing solutions that were outlined in Chapter 3 include:

1. Solutions are passive
2. Consumption of network and storage resources
3. Inability to automatically adapt to new spam patterns
4. Inability to adapt to different platforms and languages

It is clear that current detection-based Spam 2.0 filtering methods are still susceptible to the aforementioned issues which need to be tackled in order to eliminate Spam 2.0. Therefore, these

issues are the drivers of this research and the subsequent development of a new Spam 2.0 filtering method that can be extended across multiple platforms and languages as well as offer robust resistance against new Spam 2.0 tactics.

9.2.2 Problems with prevention-based spam 2.0 filtering methods

As mentioned in Chapter 1, prevention methods strive to limit automated access to the system. By utilising computational, time, and economical challenges, prevention methods make spam distribution difficult.

However, as mentioned in Chapter 2, the current state-of-the-art, prevention-based spam filtering approaches are very fragile. The main socio-economic and technical problems associated with the existing solutions that were outlined in Chapter 3 are as follows:

1. Weak/complex challenges result in ineffective Spam 2.0 filtering
2. Computer programs are becoming better at bypassing CAPTCHA
3. Computer programs are becoming faster at meeting POW challenges
4. Current prevention-based methods are vulnerable to relay attacks
5. Blacklists are too slow to protect Web 2.0 platforms
6. Blacklists are not effective for Spam 2.0 filtering

The current prevention-based Spam 2.0 filtering methods are still susceptible to the abovementioned issues which need to be tackled effectively in order to eliminate Spam 2.0.

9.2.3 Problems with early-detection based spam 2.0 filtering methods

As mentioned in Chapter 1, early-detection methods prevent the distribution of spam. However, such methods are not detection-based since they do not investigate content to find spam patterns, nor do they apply costs on user actions to limit the access to the system. Early-detection methods typically look into the nature and behaviour of spam generation. By studying the behaviour of content distribution and examining behavioural features, early-detection methods are able to differentiate spam from legitimate content.

However, as mentioned in Chapter 2, the state-of-the-art, early-detection based spam filtering approaches are very fragile. As outlined in Chapter 3, the main socio-economic and technical problems with existing solutions are as follows:

1. Inability to identify Web 2.0 spam users
2. Machine learning-based methods are not effective for real-time detection
3. The nature of Spam 2.0 is not well understood

These three issues and weaknesses of the current early-detection based Spam 2.0 filtering methods have not been adequately addressed by current studies.

9.3 Dissertation Contributions

To address and analyse problems outlined in Chapter 3, this dissertation proposes the following solutions in the area of Spam 2.0 filtering. To evaluate the proposed solutions, three prototype systems have been developed.

- Characterising Spam 2.0, Spam 2.0 distribution model and generation approaches.
- Early-Detection based Spam 2.0 Filtering (EDSF) approach.
- One-the-Fly Spam 2.0 Filtering (OFSF) approach.

9.3.1 Study and evaluation of existing spam filtering research

This dissertation has provided an overview of the literature and has investigated the current efforts in the spam filtering field as well as evaluating the most successful anti-spam methods. A survey of spam filtering efforts on the web and Web 2.0 platforms including detection-based (i.e. content-based, linked based and rule-based), prevention-based and early-detection based approaches have been conducted. Although this research brings in a new concept of Spam 2.0 that has not been covered before, attempts that have been utilised in relation to the other types of spam (i.e. web spam) are also investigated to gain additional understanding.

Through study and evaluation of existing spam filtering approaches in the web spam area, it has became clear that the state-of-art methods whether they have focus on providing a more robust

webpage ranking algorithm or identification of web spam tactics in the search results. Technically, the methods are mainly toward link and detection-based approaches, which raises the following concerns:

1. Methods that strive to enhance the PageRank algorithm need significant amounts of resources. This makes them usable only for the large scale implementations (e.g. major search engines) and they are not efficient on a smaller scale.

2. Methods that strive to enhance the HITS algorithm cannot be used in offline mode. This might cause system performance issues during the system run time.

3. Both link and detection-based approaches are afterthoughts. The methods deal with the web spam problem after spam web pages have already infiltrated the web. Hence, they cannot prevent the web spam distribution from wasting resources on the web.

4. Early detection-based methods are not effectively designed to prevent web spam distribution. However, there is no study on prevention-based web spam filtering methods.

5. Hijacking is one of the major tactics that has recently been used by spammers. This tactic has not specifically been covered by state-of-the-art web spam filtering methods.

6. The majority of web spam filtering methods have addressed link-based tactics; however, content-based tactics have not received much attention.

Since the focus of this dissertation is on Spam 2.0, a comprehensive study has been conducted on existing Spam 2.0 filtering methods. Spam 2.0 is associated with existing web spam concerns as well as producing new problems. The overall problems with the current Spam 2.0 filtering methods are as follows:

1. Methods are platform-dependent. They cannot be extended to other platforms.

2. Detection-based methods are mainly content-based and can be bypassed by spammers.

3. Content-based methods are language-dependent. Hence, a solution might not be applicable to different languages.

4. Prevention-based methods are about to become outdated as computers become more powerful. The studies show that state-of-the-art prevention method such as CAPTCHA are weak against current spam tactics.

5. Early detection-based methods did not fully consider spambot identifications because they have been designed to identify only web crawlers in general.

Hence, one of the main contributions of this study was a survey of web spam and Spam 2.0 filtering approaches to highlight the unresolved issues and at the same time properly structure the knowledge in the domain of spam filtering.

9.3.2 Characterising spam 2.0, spam 2.0 distribution model and generation approaches

Current literature lacks a comprehensive understanding of Spam 2.0 and the way it is propagated on Web 2.0 platforms. Chapter 5 addresses these problems by providing a mechanism named HoneySpam 2.0 to monitor, track and store Spam 2.0. This information is later analysed and investigated to discover the characteristics of Spam 2.0 and its distribution model. HoneySpam 2.0 is tested against real-world Spam 2.0 data and its performance is compared with that of existing solutions.

HoneySpam 2.0 has three main parts as illustrated in Figure 9.1.

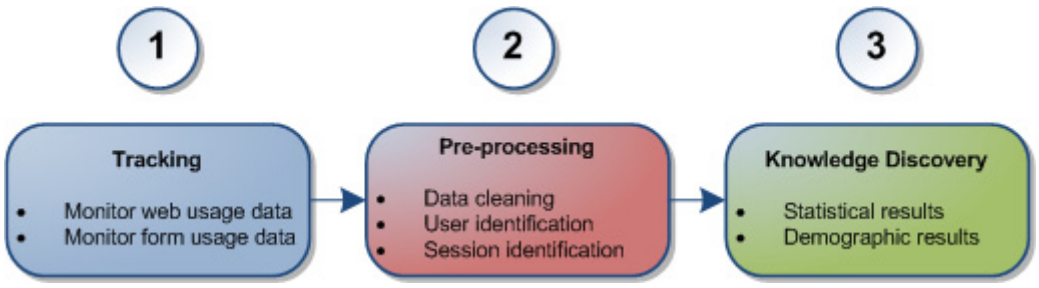


Figure 9.1: Three main parts of the HoneySpam 2.0 framework

Tracking presents a framework for monitoring web and form usage data. It provides input data for other parts of the system. The input data consists of navigation usage data (i.e. IP address, referrer URL, browser identity etc) and form usage data (i.e. mouse movement, keyboard actions, form load/unload, form's field focus/un-focus etc.).

Pre-processing includes tasks such as data cleansing, user identification, and session identification. In the data cleansing task, unnecessary data is removed. User identification is the task of associating usage data with a specific user. Finally, session identification refers to the splitting of usage data into different sessions. Each session contains information about usage data for website visits.

Knowledge discovery is the last part of the proposed solution. The output of the previous part of the system is used here to mine and characterise Spam 2.0. General data mining techniques are used to generate usage statistics, average amount of time spent on each page, return visit rate etc.

HoneySpam 2.0 has been executed on multiple web hosts including six commercial hosts and five free web hosts. The following observations have been made from spam user activities during general statistical analysis.

- Usage of search engines to find target websites
- Create numerous user accounts
- Low website webpage hits and revisit rates
- Distribute spam content in a short period of time
- No web form interaction

The in-depth analysis presented in Chapter 6 is based on the following factors in order to extract valuable information from accumulated data.

- Discovered spammer behaviour on Web 2.0 platforms
- Understood spammer intentions on Web 2.0 platforms
- Formulated spammer behaviour in order to develop discriminative feature sets
- Clustered spammer activities into groups based on features sets
- Characterised spammer behaviour in each cluster
- Developed a strategy to utilise feature sets for Spam 2.0 filtering

Three feature sets – Action Set, Action Time, and Action Frequency are presented to formulate spam users behaviour on web 2.0 platforms as well as a neural network clustering mechanism is used to cluster spammers' behaviour based on the three feature sets.

Three feature sets – Action Set, Action Time, and Action Frequency are presented to formulate spammer behaviour on web 2.0 platforms as well as a neural network clustering mechanisms are used to cluster spammer behaviour based on the three feature sets.

An Action is a user set of requested web objects in order to perform a certain task or purpose. Actions integrate user activity on web platforms into meaningful groups. Each group contains meaningful information about user tasks. An Action Set is a binary model that shows whether an Action is undertaken by a user. This feature set shows whether or not spam users perform different Actions among different transactions. Action Time is a feature vector to represent the time that is taken to perform each Action. This feature set shows how much time spam users would spend on performing each Action in different transactions. Action Frequency is a feature vector to represent the frequency of each Action. This feature set shows how frequent spammers perform each Action in different transactions. Actions are used to create Behavioural vectors to model users' web usage behaviour.

Behavioural vectors are high dimensional which makes them difficult to visualise. In order to understand this data, the Self-Organising Map (SOM) is used to reduce the dimension of data as well as clustering data into similar groups. SOM – a neural network clustering algorithm – is employed for clustering and visualisation of high dimensional data. The output of the SOM is a map which is a visual representation of input vector. Each SOM cluster provides insight into the nature of Spam 2.0 behaviour and their characteristics.

Nine experiments based on three transaction level and three feature sets have been done to profile Spam 2.0 behaviour. The overall results from each experiment and the characteristic of the centre node inside each cluster are as follows.

- Spam users focus on specific and limited Actions compared to real human users.
- Spam users use multiple user accounts to spread spam content, hide their identity and bypass restrictions.

- Spam users bypass filling in submission forms and directly submit the content to the Web server in order to efficiently spread spam.
- Spam users can be grouped into four different categories based on their actions – content submitters, profile editors, content viewers and mixed behaviour.
- Spam users change their IP address based on different Actions to hide their tracks.

The results of this study paved the way for the development of a Spam 2.0 filtering solution.

9.3.3 Early-Detection based Spam 2.0 Filtering (EDSF) approach

The early-detection based Spam 2.0 filtering methods attempt to identify spam behaviour prior to or just after spam content submission. Such methods investigate the behaviour of a spam submitter, e.g. automated tools, spambots etc.

The proposed EDSF method in this dissertation automatically distinguishes Spam 2.0 submission from genuine human content contribution inside Web 2.0 platforms. The proposed method studies, investigates and identifies Spam 2.0 in Web 2.0 platform. EDSF makes use of web usage navigation behaviour to build up a discriminative feature set including Action Set, Action Time, and Action Frequency. This feature set is later used in a machine learning classifier known as Support Vector Machine (SVM) for classification.

For the experiments used for this research, data was collected from the HoneySpam 2.0 framework as well as a live moderated forum. The performance of the EDSF is measured based on the F1-measure.

The EDSF achieved an average accuracy of 94.70%, which ranges from 93.18% for Action Time to 96.23% for Action Frequency. For the first experiment on the Action Set, the number of correct classified instances is 4,064 and the number of incorrect classified instances is 163. The number of correct classified instances for the second and third experiment is 3,939 and 4,068 respectively. The number of incorrect classified instances for the last two experiments is 288 and 159 respectively.

The overall performance of EDSF has been compared against four other state-of-the-art Spam 2.0 filtering methods. The overall F1-measure value of EDSF is higher than existing Spam 2.0 filtering method. It shows that the proposed method has better performance in regard to Spam 2.0 filtering.

The other advantage of EDSF is its detection strategy. EDSF is an early-detection based method, so it can detect spam users before content distribution

Given that there is no work in the literature on the classification of Spam 2.0 behaviour, the results of EDSF are the initial steps toward identifying Spam 2.0 by examining web usage behaviour.

9.3.4 On-the-Fly Spam 2.0 Filtering (OFSF) approach

The OFSF method presented in this dissertation automatically differentiates Spam 2.0 submissions from genuine human content inside Web 2.0 platforms. The proposed method studies, investigates and identifies Spam 2.0 on a Web 2.0 platform. In a similar fashion to EDSF, OFSF makes use of web usage navigation behaviour to establish discriminative feature sets. However, OFSF considers the sequence and order of user web navigation in order to build up the feature sets for classification. Unlike EDSF, OFSF uses a novel rule-based classifier to store, retrieve, and classify spam users separately from genuine human. OFSF makes it possible to spot spam users on-the-fly. While users are interacting with the Web 2.0 platform, OFSF is able to track and identify possible spam navigation patterns. To handle i.e. store, retrieve, update, and lookup navigation data OFSF uses a Trie data structure. Trie is in the form of tree data structure where each node stores an identifier to a user's web navigation Actions while the tree edges hold the order and sequence of Actions.

OFSF uses Action Strings (sequence of users Actions in a given transaction where each Action is identified by a text character). By placing each Action identifier in the specific Action order, an Action String and Trie data structure is generated. The main advantages are ease of updating, handling, and short access time. To classify spam user from real human users, a Trie is in the form of a tree structure where each node contains the value of the key with which it is associated.

For the experiments outlined in this dissertation, the same dataset that has been used in EDSF was also used for OFSF. F1-Measure, Matthews Correlation Coefficient (MCC) was used to measure the performance of OFSF. The overall average accuracy achieved with the OFSF experiments was 93% where the best accuracy is 96% on window size equal to or greater that 5 characters. The best F1 value that was achieved was 0.82 while the average F1 value was 0.75 on all the datasets. The average MCC value was 0.744 where the best MCC value was 0.808 for window size equal to or greater than 5 characters.

Overall, the average performance of OFSF compared with EDSF is lower. However, EDSF's better performance is achieved at the cost of time and computational resources. Additionally, on-the-fly identification of spam users in OFSF surpasses EDSF since the EDSF classifier cannot be trained with a partially completed input vector.

9.4 Future Works

The work that has been undertaken in this dissertation has been published in peer reviewed international conferences, journals and book chapters. Over the course of this research, 15 papers have been published in international conferences, journals and book chapters. Selected publications are attached in the Appendix. Although a significant amount of effort has been invested in this research, there is still scope for future work. The following projects are aims for the future works:

- Integrate HoneySpam 2.0 to collect data from various Web 2.0 platforms other than online forums. HoneySpam 2.0 can also be publicly available as a web service to accumulate more data from online web applications.

- Improve the performance of EDSF by investigating the following issues:

    - The nature of EDSF does not allow a distinction to be made between fair or spam Actions; hence, the EDSF can mark as spam users those genuine users with few Actions.

    - EDSF investigates the behaviour of the users only at the session level. Although a session level transaction provides a detailed profile of a user's visit, it does not provide an overall view of the user's behaviour.

    - Although Actions are good discriminative features for classification, some Actions might be similar for both genuine and spam users. For example, a genuine user which directly inputs a "creating new thread" link address might be wrongly classified as a spam user since spam users mainly aim for content contribution web pages rather than surfing through other web pages. This increases the chance of misclassifications. Therefore, further classification of Actions may be required.

- OFSF can be improved by considering the following:

- An adaptive mechanism to automatically update the classifier based on new unseen Action Strings.
- Inclusion of other usage behaviours into Action Strings such as time of each Action, existence of mouse / keyboard activity, etc. in order to differentiate between genuine human and spam Action Strings.
- A mechanism which will evolve as a site is updated and changed.

This work has coined the term ‘Spam 2.0’, provided insight into the nature of Spam 2.0 and proposed filtering mechanisms to combat this new and rapidly evolving problem.

Bibliography

Abernethy, J., Chapelle, O. & Castillo, C., 2010. Graph regularization methods for Web spam detection. *Machine Learning*, 81(2), pp.207-225.

Abram, H., Michael, W.G. & Richard, C.H., 2008. Reverse Engineering CAPTCHAs.

Adamic, L.A. et al., 2000. Power-Law Distribution of the World Wide Web. *Science*, 287(5461), p.2115a-.

AdelaideNow, 2011. Online scams cost Australians $1bn a year | Adelaide Now. Available at: http://www.adelaidenow.com.au/news/south-australia/online-scams-cost-australians-1bn-a-year/story-e6frea83-1226017977647 [Accessed March 23, 2011].

Akismet. 2011. About Akismet - The automattic spam killer. http://akismet.com/about/.

Alexandros, N. et al., 2006. Detecting spam web pages through content analysis.

Alpaydin, E., 2004. *Introduction to Machine Learning*, The MIT Press.

Andreolini, M. et al., 2005. HoneySpam: honeypots fighting spam at the source. In Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop (2005). Cambridge, MA: USENIX Association, p. 11.

Araujo, L. & Martinez-Romo, J., 2010. Web Spam Detection: New Classification Features Based on Qualified Link Analysis and Language Models. *Information Forensics and Security, IEEE Transactions on*, 5(3), pp.581-590.

Asano, Y., Tezuka, Y. & Nishizeki, T., 2007. Improvements of HITS Algorithms for Spam Links. In *Advances in Data and Web Management*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 479-490. Available at: http://dx.doi.org/10.1007/978-3-540-72524-4_50.

Back, A., 2002. *Hashcash - A Denial of Service Counter-Measure*, Available at: http://www.hashcash.org/ [Accessed December 23, 2010].

Baldi, Pierre, Soren Brunak, Yves Chauvin, Claus A. F. Andersen, and Henrik Nielsen. 2000. “Assessing the accuracy of prediction algorithms for classification: an overview.” *Bioinformatics* 16 (5): 412-424. doi:10.1093/bioinformatics/16.5.412.

Benczúr, A. et al., 2005. SpamRank - Fully Automatic Link Spam Detection Work in progress. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*.

Benczúr, A.A. et al., 2008. Web Spam: a Survey with Vision for the Archivist. In International Web Archiving Workshop (IWAW). Aaarhus, Denmark.

Benevenuto, F. et al., 2008. Identifying Video Spammers in Online Social Networks. In AIRWeb ’08.

Benevenuto, F. et al., 2009. Detecting spammers and content promoters in online video social networks. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. Boston, MA, USA: ACM, pp. 620-627.

Benevenuto, F. et al., 2010. Detecting Spammers on Twitter. In CEAS2010. Redmond, Washington.

Berners-Lee, T., 1945. RFC 1945 - Hypertext Transfer Protocol -- HTTP/1.0. Available at: http://tools.ietf.org/html/rfc1945 [Accessed February 9, 2011].

Berti-Equille, L. et al., 2009. Sailing the Information Ocean with Awareness of Currents: Discovery and Application of Source Dependence. In *the Conference on Innovative Data Systems Research*. CA, USA.

Bogers, A.M. & Bosch, A. van den, 2008. Using language models for spam detection in social bookmarking. In Proceedings of 2008 ECML/PKDD discovery challenge workshop. pp. 1-12.

Carvalho, A.L. da C. et al., 2006. Site level noise removal for search engines. In *Proceedings of the 15th international conference on World Wide Web*. Edinburgh, Scotland: ACM, pp. 73-82.

Caverlee, J. et al., 2009. A Parameterized Approach to Spam-Resilient Link Analysis of the Web. *Parallel and Distributed Systems, IEEE Transactions on*, 20(10), pp.1422-1438.

Chang, C., and C. Lin. 2001. LIBSVM: a library for support vector machines. Ed. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chellapilla, K. & Chickering, D., 2006. Improving Cloaking Detection using Search Query Popularity and Monetizability. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*. pp. 17-24.

Chellapilla, Kumar, and Patrice Simard. 2004. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In *NIPS*. citeulike-article-id:2719029 http://dblp.uni-trier.de/rec/bibtex/conf/nips/ChellapillaS04.

Cooley, R., Mobasher, B. & Srivastava, J., 1997. Web mining: information and pattern discovery on the World Wide Web. In Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on. pp. 558-567.

Cooley, R., Mobasher, B. & Srivastava, J., 1999. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1), pp.5-32.

Cooley, Robert, Mobasher, Bamshad & Srivastava, Jaideep, 1999. Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems, 1(1), pp.5-32.

Cormack, G.V., Hidalgo, J.M.G. & Sánz, E.P., 2007. Feature engineering for mobile (SMS) spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. Amsterdam, The Netherlands: ACM, pp. 871-872.

Cosentino, C., 2002. Advanced PHP for Web Professionals, Prentice Hall.

Daigle, L., 2004. RFC 3912 - WHOIS Protocol Specification. Available at: http://tools.ietf.org/html/rfc3912 [Accessed February 9, 2011].

Defensio. 2011. Defensio · Security for the Social Web. http://defensio.com/.

Dellschaft, K. & Staab, S., 2010. On Differences in the Tagging Behavior of Spammers and Regular Users. In *Proceedings of the Web Science Conference 2010*.

Desai, Apurva. 2008. *Computer Graphics*. PHI Learning.

dmoz, 2011. ODP - Open Directory Project. Available at: http://www.dmoz.org/ [Accessed June 7, 2011].

DuBois, P., 2006. MySQL Cookbook, Second Edition, O'Reilly Media.

Edward, Fredkin. 1960. "Trie memory." *Commun. ACM* 3 (9): 490-499. doi:http://doi.acm.org/10.1145/367390.367400.

Fahlman, S.E., 2002. Selling interrupt rights: A way to control unwanted e-mail and telephone calls [Technical forum].

Fetterly, D., Manasse, M. & Najork, M., 2004. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*. Paris, France: ACM, pp. 1-6.

Fielding, R. et al., 1999. Hypertext Transfer Protocol -- HTTP/1.1 - 9 Method Definitions [Accessed Online Dec 2009]. *http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html*.

Flanagan, D., 2006. JavaScript: The Definitive Guide, Fifth Edition, O'Reilly Media.

Fogaras, D. & Rácz, B., 2004. Towards Scaling Fully Personalized PageRank. In *Algorithms and Models for the Web-Graph*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 105-117. Available at: http://dx.doi.org/10.1007/978-3-540-30216-2_9.

Gao, H. et al., 2010. Detecting and characterizing social spam campaigns. In *Proceedings of the 17th ACM conference on Computer and communications security*. Chicago, Illinois, USA: ACM, pp. 681-683.

Geddes, B., 2010. Advanced Google AdWords, Sybex.

Grier, C. et al., 2010. @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*. Chicago, Illinois, USA: ACM, pp. 27-37.

Grier, Chris, Kurt Thomas, Vern Paxson, and Michael Zhang. 2010. @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, 27-37. Chicago, Illinois, USA: ACM.

Gutmans, A., Bakken, S. & Rethans, D., 2004. PHP 5 power programming, Prentice Hall.

Gyöngyi, Z. & Garcia-Molina, H., 2005. Spam: It's Not Just for Inboxes Anymore. *Computer*, 38(10), pp.28-34.

Gyöngyi, Z. & Garcia-Molina, H., 2005. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*. Chiba, Japan.

Gyöngyi, Z., Garcia-Molina, Hector & Pedersen, J., 2004. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*. Toronto, Canada: VLDB Endowment, pp. 576-587. Available at: http://portal.acm.org/citation.cfm?id=1316689.1316740# [Accessed March 8, 2010].

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. “The WEKA data mining software: an update.” *SIGKDD Explor. Newsl.* 11 (1): 10-18.

Harrenstien, K., 1982. RFC 812 - NICNAME/WHOIS. Available at: http://tools.ietf.org/html/rfc812 [Accessed February 9, 2011].

Hayati, P. & Potdar, V., 2008. Evaluation of Spam Detection and Prevention Frameworks for Email and Image Spam - A State of Art. In The 2nd International Workshop on Applications of Information Integration in Digital Ecosystems (AIIDE 2008). Linz, Austria.

Hayati, P. & Potdar, V., 2009. Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods. In 7th IEEE International Conference on Industrial Informatics. Cardiff, Wales.

Hayati, P. et al., 2009. HoneySpam 2.0: Profiling Web Spambot Behaviour. In *12th International Conference on Principles of Practise in Multi-Agent Systems*. Nagoya, Japan: Lecture Notes in Artificial Intelligence, pp. 335-344.

Hayati, P. et al., 2010. Definition of Spam 2.0: New Spamming Boom. In *IEEE International Conference on Digital Ecosystems and Technologies (DEST 2010)*. Dubai, UAE: IEEE.

Hayati, P., Potdar, V., Chai, K., et al., 2010. Web Spambot Detection Based on Web Navigation Behaviour. In *24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010)*. Perth, Western Australia: IEEE.

Hayati, P., Potdar, V., Sarenche, S., et al., 2010. Definition of Spam 2.0: New Spamming Boom. In *IEEE International Conference on Digital Ecosystems and Technologies (DEST 2010)*. Dubai, UAE: IEEE.

Hayati, Pedram, Kevin Chai, Alex Talevski, and Vidyasagar Potdar. 2010. Behaviour-Based Web Spambot Detection by Utilising Action Time and Action Frequency. In *The 2010 International Conference on Computational Science and Applications (ICCSA 2010)*. Fukuoka, Japan: Lecture Notes in Computer Science (LNCS), Springer.

Hayati, Pedram, Vidyasagar Potdar, Kevin Chai, and Alex Talevski. 2010. Web Spambot Detection Based on Web Navigation Behaviour. In *24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010)*. Perth, Western Australia: IEEE.

Hayati, Pedram, Vidyasagar Potdar, W. F. Smyth, and Alex Talevski. 2010. Rule-Based Web Spambot Detection Using Action Strings. In *The Seventh Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS 2010) (Under Review)*, ed. 2010. Redmond, Washington.

Heikkinen, E., 2011. Pligg CMS | Open Source Content Management System. Available at: http://pligg.com/ [Accessed June 7, 2011].

Hindle, A, M. W. Godfrey, and R.C. Holt. 2008. “Reverse Engineering CAPTCHAs.” Proceedings of the 2008 15th Working Conference on Reverse Engineering - Volume 00. doi:http://dx.doi.org/10.1109/WCRE.2008.35.

Holdener III, A.T., 2008. Ajax: The Definitive Guide, O’Reilly Media.

Horstkotte, D. et al., 2004. Guidelines on Prevention, Diagnosis and Treatment of Infective Endocarditis Executive Summary. *European Heart Journal*, 25(3), pp.267-276.

Houck, Chad. 2010. Decoding reCAPTCHA DEF CON 18 Hacking Conference.

Huang, C., Jiang, Q. & Zhang, Yan, 2010. Detecting Comment Spam through Content Analysis. In *Web-Age Information Management*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 222-233. Available at: http://dx.doi.org/10.1007/978-3-642-16720-1_23.

Hunt, C., 2002. TCP/IP Network Administration, O'Reilly Media.

Ion, A. et al., 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages.

Irani, D. et al., 2010. Study of TrendStuffing on Twitter through Text Classification. In CEAS2010. Redmond, Washington.

Jacobs, K., Frye, C. & Frye, D., 2008. EXCEL 2007 Charts Mage Easy, McGraw-Hill Osborne Media.

Jindal, N. & Bing, L., 2007. Analyzing and Detecting Review Spam. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. pp. 547-552.

Jun Bi, Jianping Wu & Wenmao Zhang, 2008. A Trust and Reputation based Anti-SPIM Method. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. INFOCOM 2008. The 27th Conference on Computer Communications. IEEE. pp. 2485-2493. Available at: 10.1109/INFOCOM.2008.319.

Junod, J., 1997. Servers to spam: drop dead, Computers and Security. *Elsevier*, 16, pp.623-623.

Kleinberg, J.M., 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5), pp.604-632.

Kohavi, and Provost. 1998. "Glossary of Terms." *Machine Learning* 30 (2) (February 1): 271-274.

Kohonen, T. 1990. "The self-organizing map." *Proceedings of the IEEE* 78 (9): 1464-1480.

Kolari, P., Java, A. & Finin, T., 2006. Characterizing the Splogosphere. In *Proceedings of the 3rd Annual Workshop on the Weblogging Ecosystem*. Available at: citeulike-article-id:2914188.

Kolari, Pranam, Finin, Tim & Joshi, A., 2006. SVMs for the blogosphere: Blog identification and splog detection. In *AAAI Spring Symposium on Computational Approaches to Analysing Weblogs*. Stanford University, California. Available at: citeulike-article-id:2913705.

Koster, M., 1993. Guidelines for Robot Writers [Accessed Online Dec 2009]. *http://www.robotstxt.org/guidelines.html*.

Koster, M., 1994. A Standard for Robot Exclusion [Accessed Online Dec 2009]. *http://www.robotstxt.org/orig.html*.

Koster, M., 1995. Robots in the Web: threat or treat? [Accessed Online Dec 2009]. *http://www.robotstxt.org/threat-or-treat.html*.

Koutrika, G. et al., 2008. Combating spam in tagging systems: An evaluation. *ACM Trans. Web*, 2(4), pp.1-34.

Krause, B. et al., 2008. The anti-social tagger: detecting spam in social bookmarking systems. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*. Beijing, China: ACM, pp. 61-68.

Le, Zhang, Zhu Jingbo, and Yao Tianshun. 2004. “An evaluation of statistical spam filtering techniques.” *ACM Transactions on Asian Language Information Processing (TALIP)* 3 (4): 243-269. doi:http://doi.acm.org/10.1145/1039621.1039625.

Lee, K., Caverlee, James & Webb, S., 2010. Uncovering social spammers: social honeypots + machine learning. In Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval. Geneva, Switzerland: ACM, pp. 435-442.

Levine, J.R., 2005. Experiences with greylisting. In Conference on Email and Anti-Spam.

Lim, E.-P. et al., 2010. Detecting Product Review Spammers using Rating Behaviors. In The 19th ACM International Conference on Information and Knowledge Management. Toronto, Canada. Available at: http://ink.library.smu.edu.sg/sis_research/623/ [Accessed November 22, 2010].

Liu, K., Fang, B. & Zhang, Yu, 2009. Detecting tag spam in social tagging systems with collaborative knowledge. In *Proceedings of the 6th international conference on Fuzzy systems and knowledge discovery - Volume 7*. Tianjin, China: IEEE Press, pp. 427-431.

Liu, Yuting et al., 2008. BrowseRank: letting web users vote for page importance. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. Singapore, Singapore: ACM, pp. 451-458.

Live-Spam-Zeitgeist, 2009. Some Stats, Akismet.

Matthews, B. W. 1975. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme.” *Biochim Biophys Acta* 405 (2): 442-451.

May, M., 2005. Inaccessibility of CAPTCHA. *W3C Working Group*. Available at: http://www.w3.org/TR/turingtest/ [Accessed December 23, 2010].

May, Matt. 2005. Inaccessibility of CAPTCHA. *W3C Working Group*. http://www.w3.org/TR/turingtest/.

Mediawiki, 2011. MediaWiki. Available at: http://www.mediawiki.org/wiki/MediaWiki [Accessed June 7, 2011].

Mertz, D., 2004. Charming Python: Beat spam using hashcash.

Metaxas, P. & Destefano, J., 2005. Web Spam, Propaganda and Trust. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*.

Michalski, R.S., Carbonell, J.G. & Mitchell, T.M., 1985. *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.

Miles, R. & Hamilton, K., 2006. Learning UML 2.0, O’Reilly Media. Available at: http://oreilly.com/catalog/9780596009823.

Molinaro, A., 2005. SQL Cookbook, O’Reilly Media.

Narisawa, K. et al., 2007. Unsupervised Spam Detection Based on String Alienness Measures. In *Discovery Science*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 161-172. Available at: http://dx.doi.org/10.1007/978-3-540-75488-6_16.

net-security.org, 2008. Latest spam statistics. *http://www.net-security.org/secworld.php?id=6056*. Available at: http://www.net-security.org/secworld.php?id=6056.

Nitin, J. & Bing, L., 2008. Opinion spam and analysis.

Ogbuji, U., 2008. Real Web 2.0: Battling Web spam.

Oreilly, T., 2007. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, 1, p.17.

Page, L. et al., 1998. *The PageRank Citation Ranking: Bringing Order to the Web*, Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768.

Park, K. et al., 2006. Securing Web Service by Automatic Robot Detection. In *USENIX 2006 Annual Technical Conference Refereed Paper*.

Paul, H., Georgia, K. & Hector, G.-M., 2007. Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges. *IEEE Internet Computing*, 11(6), pp.36-45.

Perkins, A., 2001. The Classification of Search Engine Spam. Available at: http://www.silverdisc.co.uk/articles/spam-classification [Accessed November 4, 2010].

Pfleeger, S.L. & Bloom, G., 2005. Canning SPAM: Proposed solutions to unwanted email. *Security & Privacy, IEEE*, 3(2), pp.40-47.

phpBB, 2011. phpBB • Free and Open Source Forum Software. Available at: http://www.phpbb.com/ [Accessed June 7, 2011].

Pilone, D. & Pitman, N., 2005. UML 2.0 in a Nutshell, O'Reilly Media. Available at: http://oreilly.com/catalog/9780596007959.

Quinlan, J.R., 1992. *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Quittek, J. et al., 2008. On Spam over Internet Telephony (SPIT) Prevention. *Communications Magazine, IEEE*, 46(8), pp.80-86.

R. Fielding et al., 1996. Hypertext Transfer Protocol - HTTP/1.1. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.3801.

Rijsbergen, C.J.V., 1979. *Information retrieval*, Butterworths.

Sculley, D., and M. Wachman Gabriel. 2007. "Relaxed online SVMs for spam filtering." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. doi:http://doi.acm.org/10.1145/1277741.1277813.

SMF Team, 2011. Simple Machines. Available at: http://www.simplemachines.org/ [Accessed June 7, 2011].

Smith, R.W., 2002. Advanced Linux Networking, Addison-Wesley Professional.

Spiliopoulou, M. et al., 2003. A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. INFORMS JOURNAL ON COMPUTING, 15(2), pp.171-190.

Spitzner, L., 2002. *Honeypots tracking hackers*, Addison-Wesley Professional. Available at: http://www.informit.com/store/product.aspx?isbn=9780321108951.

Steinbach, M., Karypis, G. & Kumar, V, 2000. A comparison of document clustering techniques. In KDD workshop on text mining. Available at: http://www.cs.cmu.edu/~dunja/KDDpapers/Steinbach_IR.pdf.

Steven J. J. Tedjamulia. 2005. Motivating Content Contributions to Online Communities: Toward a More Comprehensive Theory. In *Hawaii International Conference on System Sciences*, 7:193b. January 3. http://doi.ieeecomputersociety.org/10.1109/HICSS.2005.444.

Takemura, T. & Ebara, H., 2008. Spam Mail Reduces Economic Effects. In *Digital Society, 2008 Second International Conference on the*. pp. 20-24.

Tan, P.-N. & Kumar, V., 2002. Discovery of Web Robot Sessions Based on their Navigational Patterns. *Data Mining and Knowledge Discovery*, 6(1), pp.9-35.

TypePad AntiSpam. 2011. TypePad AntiSpam. http://antispam.typepad.com/.

Uemura, T., Ikeda, D. & Arimura, H., 2008. Unsupervised Spam Detection by Document Complexity Estimation. In *Discovery Science*. pp. 319-331. Available at: http://dx.doi.org/10.1007/978-3-540-88411-8_30.

Urvoy, T. et al., 2008. Tracking Web spam with HTML style similarities. *ACM Trans. Web*, 2(1), pp.1-28.

von Ahn, Luis, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. "reCAPTCHA: Human-Based Character Recognition via Web Security Measures." *Science* 321 (5895): 1465-1468.

von Ahn, Luis, Manuel Blum, Nicholas Hopper, and John Langford. 2003. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology — EUROCRYPT 2003*, 646-646. http://dx.doi.org/10.1007/3-540-39200-9_18.

Wang, A., 2010. Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach. In *Data and Applications Security and Privacy XXIV*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 335-342. Available at: http://dx.doi.org/10.1007/978-3-642-13739-6_25.

Wang, A.H., 2010. Don't follow me: Spam detection in twitter. In *Int'l Conference on Security and Cryptography (SECRYPT)*.

Webb, S., Caverlee, J. & Pu, C., 2008. Social Honeypots: Making Friends with a Spammer Near You. In Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008). Mountain View, CA.

Webb, Steve, Caverlee, James & Pu, Calton, 2008. Predicting web spam with HTTP session information. In *Proceeding of the 17th ACM conference on Information and knowledge management*. Napa Valley, California, USA: ACM, pp. 339-348.

Whitworth, B. & Whitworth, E., 2004. Spam and the social-technical gap. *Computer*, 37(10), pp.38-45.

Wordpress, 2011. Blog Tool and Publishing Platform. Available at: http://mu.wordpress.org/ [Accessed June 7, 2011].

Workathome, 2009. Work from home online ad placing work pay per posting.

Wu, B. & Davison, B.D., 2005a. Cloaking and Redirection: A Preliminary Study. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.3390.

Wu, B. & Davison, B.D., 2005b. Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. Chiba, Japan: ACM, pp. 820-829.

Wu, B. & Davison, B.D., 2006. Detecting semantic cloaking on the web. In *Proceedings of the 15th international conference on World Wide Web*. Edinburgh, Scotland: ACM, pp. 819-828.

Xiaoxin, Y., Jiawei, H. & Philip, S.Y., 2007. Truth discovery with multiple conflicting information providers on the web.

Yahoo!, 2009. How to Prevent Your Site or Certain Subdirectories From Being Crawled [Accessed Online Dec 2009]. *http://help.yahoo.com/l/us/yahoo/search/webcrawler/*.

Yan, X. & Su, X.G., 2009. *Linear Regression Analysis: Theory and Computing*, World Scientific Publishing Company.

Yi-Min Wang & Ming Ma, 2007. Strider Search Ranger: Towards an Autonomic Anti-Spam Search Engine. In *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*. Autonomic Computing, 2007. ICAC '07. Fourth International Conference on. p. 32. Available at: 10.1109/ICAC.2007.38.

Yi-Min, W. et al., 2007. Spam double-funnel: connecting web spammers with advertisers. In Proceedings of the 16th international conference on World Wide Web. New York, NY, USA.

Yiqun, L. et al., 2008. Identifying web spam with user behavior analysis. In Proceedings of the 4th international workshop on Adversarial information retrieval on the web. New York, NY, USA.

Yu, H. et al., 2009. Web Spam Identification with User Browsing Graph. In *Information Retrieval Technology*. pp. 38-49. Available at: http://dx.doi.org/10.1007/978-3-642-04769-5_4.

Zakas, N.C., McPeak, J. & Fawcett, J., 2006. Professional Ajax, Wrox.

Zhijun Liu et al., 2005. Detecting and filtering instant messaging spam - a global and personalized approach. In *Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on*. Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on. pp. 19-24. Available at: 10.1109/NPSEC.2005.1532048.

Zhou, H. & Dang, X., 2007. A new generalized similarity-based topic distillation algorithm. *Wuhan University Journal of Natural Sciences*, 12(5), pp.789-792.

Zinman, A. & Donath, J., 2007. Is Britney Spears Spam?. In Fourth Conference on Email and Anti-Spam. Mountain View, CA.

Appendix I

Early-Detection based Spam 2.0 Filtering (EDSF) classification results:

Action Set

```
=== Run information ===

Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5
-M 40.0 -C 1.0 -E 0.0010 -P 0.1
Relation:     Man
Instances:    4227
Attributes:   35
              Index
              View Board
              View Topic
              Start new topic
              View the profile of
              Set Search Parameter
              Search Results
              Preview
              Profile - Notificati
              Profile - Account Re
              Modify message
              Post reply
              action=quotefast
              Send message
              action=spellcheck
              action=findmember
              Personal Messages In
              Viewing Members 1 to
              Recent Posts
              View Lastest Post
              View Help
              Profile - Forum Prof
              Profile - Look and L
              Choose a theme...
              Remnants Forum Passw
              Recent Unread Topics
              action=jsmodify
              Personal Message Opt
              Updated Topics
              Remnants Forum - Sta
              action=stats
              All Unread Topics
              Post new poll
              Login
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 0.38 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4064               96.1438 %
Incorrectly Classified Instances       163                3.8562 %
Kappa statistic                          0.7856
```

```
Mean absolute error                      0.0386
Root mean squared error                  0.1964
Relative absolute error                 18.4413 %
Root relative squared error             60.7532 %
Total Number of Instances             4227

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.999         0.323      0.958     1          0.979      0.838
Bot
                 0.677     0          0.997     0.677     0.806      0.838    Man
Weighted Avg.    0.961     0.285      0.963     0.961     0.958      0.838

=== Confusion Matrix ===

    a    b   <-- classified as
 3725    1 |    a = Bot
  162  339 |    b = Man
```

Action Time

```
=== Run information ===

Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5
-M 40.0 -C 1.0 -E 0.0010 -P 0.1
Relation:     Man
Instances:    4227
Attributes:   35
              Index
              View Board
              View Topic
              Start new topic
              View the profile of
              Set Search Parameter
              Search Results
              Preview
              Profile - Notificati
              Profile - Account Re
              Modify message
              Post reply
              action=quotefast
              Send message
              action=spellcheck
              action=findmember
              Personal Messages In
              Viewing Members 1 to
              Recent Posts
              View Lastest Post
              View Help
              Profile - Forum Prof
              Profile - Look and L
              Choose a theme...
              Remnants Forum Passw
              Recent Unread Topics
              action=jsmodify
              Personal Message Opt
              Updated Topics
              Remnants Forum - Sta
              action=stats
              All Unread Topics
              Post new poll
              Login
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===
```

```
LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 1.67 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3939               93.1867 %
Incorrectly Classified Instances       288                6.8133 %
Kappa statistic                          0.639
Mean absolute error                      0.0681
Root mean squared error                  0.261
Relative absolute error                 32.5834 %
Root relative squared error             80.7554 %
Total Number of Instances             4227

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.976     0.399      0.948     0.976     0.962      0.789    Bot
                 0.601     0.024      0.774     0.601     0.676      0.789    Man
Weighted Avg.    0.932     0.355      0.927     0.932     0.928      0.789

=== Confusion Matrix ===

    a    b   <-- classified as
 3638   88 |    a = Bot
  200  301 |    b = Man
```

Action Frequency

```
=== Run information ===

Scheme:       weka.classifiers.functions.LibSVM -S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5
-M 40.0 -C 1.0 -E 0.0010 -P 0.1
Relation:     Man
Instances:    4227
Attributes:   35
              Index
              View Board
              View Topic
              Start new topic
              View the profile of
              Set Search Parameter
              Search Results
              Preview
              Profile - Notificati
              Profile - Account Re
              Modify message
              Post reply
              action=quotefast
              Send message
              action=spellcheck
              action=findmember
              Personal Messages In
              Viewing Members 1 to
              Recent Posts
              View Lastest Post
              View Help
              Profile - Forum Prof
              Profile - Look and L
              Choose a theme...
              Remnants Forum Passw
              Recent Unread Topics
              action=jsmodify
              Personal Message Opt
```

```
              Updated Topics
              Remnants Forum - Sta
              action=stats
              All Unread Topics
              Post new poll
              Login
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 0.38 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4068               96.2385 %
Incorrectly Classified Instances       159                3.7615 %
Kappa statistic                          0.795
Mean absolute error                      0.0376
Root mean squared error                  0.1939
Relative absolute error                 17.9887 %
Root relative squared error             60.0032 %
Total Number of Instances             4227

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.998     0.299      0.961     0.998     0.979      0.849    Bot
                 0.701     0.002      0.975     0.701     0.815      0.849    Man
Weighted Avg.    0.962     0.264      0.963     0.962     0.96       0.849

=== Confusion Matrix ===

    a    b   <-- classified as
 3717    9 |    a = Bot
  150  351 |    b = Man
```

Appendix II

Selected Publications

- Hayati, P. et al., 2011. Characterisation of Web Spambots using Self Organising Maps. International Journal of Computer Systems Science and Engineering, vol. 1, pp.69-78.
- Hayati, Pedram, Vidyasagar Potdar, W. F. Smyth, and Alex Talevski. 2010. Rule-Based Web Spambot Detection Using Action Strings. In The Seventh Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS 2010) (Under Review), ed. 2010. Redmond, Washington.
- Hayati, Pedram, Vidyasagar Potdar, Kevin Chai, and Alex Talevski. 2010. Web Spambot Detection Based on Web Navigation Behaviour. In 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010). Perth, Western Australia: IEEE.
- Hayati, P. et al., 2010. Behaviour-Based Web Spambot Detection by Utilising Action Time and Action Frequency. In The 2010 International Conference on Computational Science and Applications (ICCSA 2010). Fukuoka, Japan: Lecture Notes in Computer Science (LNCS), Springer.
- Hayati, P. et al., 2009. HoneySpam 2.0: Profiling Web Spambot Behaviour. In 12th International Conference on Principles of Practise in Multi-Agent Systems. Nagoya, Japan: Lecture Notes in Artificial Intelligence, pp. 335-344.

Comput Syst Sci & Eng (2011) 1: 69–85

International Journal of
Computer Systems
Science & Engineering

Characterisation of Web Spambots using Self Organising Maps

Pedram Hayati, Vidyasagar Potdar, Alex Talevski and Kevin Chai

Anti Spam Research Lab, Digital Ecosystems & Business Intelligence (DEBI) Institute, Curtin University of Technology, Perth, Australia. Email: p.hayati@curtin.edu.au, v.potdar@curtin.edu.au, a.talevksi@curtin.edu.au, k.chai@curtin.edu.au

The growth of spam in Web 2.0 environments not only reduces the quality and trust of the content but it also degrades the quality of search engine results. By means of web spambots, spammers are able to distribute spam content more efficiently to more targeted websites. Current anti-spam filtering solutions have not studied web spambots thoroughly and the characterisation of spambots remains an open area of research. In order to fill this research gap, this paper utilises Kohonen's Self-Organising Map (SOM) to characterise web spambots. We analyse web usage data to profile web spambots based on three novel set of features i.e. action set, action frequency and action time. Our experimental results uncovered important characteristics of web spambots that 1) they focus on specific and limited actions compared with humans 2) they use multiple user accounts to spread spam content, hide their identity and bypass restrictions, 3) they bypass filling in submission forms and directly submit the content to the Web server in order to efficiently spread spam, 4) they can be categorise into 4 different categories based on their actions – content submitters, profile editors, content viewers and mixed behaviour, 5) they change their IP address based on different action to hide their tracks. Our results are promising and they suggest that our technique is capable of identifying spam in Web 2.0 applications.

Keywords: *web spambots, spam 2.0, spam detection, behaviour analysis, web usage mining*

1. INTRODUCTION

Web 2.0 brings freedom to the web environment by allowing users to participate in content creation & management. However, this ability is also granted to users that distribute spam content (spammers). Spammers have spread spam content on Web 2.0 applications including forums, blogs, wikis, social bookmarking websites etc. [1]

We refer to this new spamming technique as *Spam 2.0* which is defined as the propagation of unsolicited, anonymous, mass content to infiltrate legitimate Web 2.0 applications. Examples of Spam 2.0 would include posting promotional threads in online discussion boards, manipulating wiki pages and creating fake user profiles in social networking websites [1].

Recent studies report an alarming situation that over 75% of blogs are spam blogs (splogs) [2] and the amount of comment spam has doubled in the 2009 compare with last year [3].

State-of-art anti-spam filters either employ content-based approaches or out-link and in-link based approaches for spam detection. Zinman et al.(2007) use variety of content-based features in a machine learning classifier to classify social spam [4]. Nitin et al (2008) extract 34 content-based features from online opinions to build the spam classifier [5].

An unsupervised content-based spam detection method proposed by Uemura et al. [6], employs an entropy-like feature called document complexity to detect spam content in blog and forums. Webb et al. (2008) monitored the interaction of social spammers in an online community to characterise spam profiles [7]. However, an in depth study at identifying the origin and source of spam content in Web 2.0 platforms i.e. a study on characterising web spambots is missing in current literature.

Web spambot (i.e. *spambot*) is a type of *web robot* or *Internet robot* that spreads spam content in Web 2.0 applications [8]. Spammers automate the process of spreading spam content in Web 2.0 applications (spam 2.0) by means of *spambots*. Spambots can be programmed to automatically crawl the web, find Web 2.0 websites, register a user account and spread spam con-

tent. They can be deployed in any system ranging from genuine users computers (as a part of *botnets*) to servers.

Spambots make it easier to quickly distribute spam content among a large number of websites without spammer intervention. They can also be programmed to be *application-specific* or *website-specific*. The former targets special web applications such as forums, blogs, wikis, etc. The latter is designed to target a specific Web 2.0 website such as *Amazon*, *Youtube*, *MySpace*, etc.

It is worth mentioning that spambots are different from *email spambots*. Email spambots are designed to harvest email addresses from webpages and distribute spam content through emails. Web spambots however are active in Web 2.0 applications and can mimic human user behaviour [1].

Spambots waste network and server resources, pollute content in Web 2.0 applications in addition to increasing the need to filter and manage unnecessary content. Additionally spambots can place a legitimate websites in a danger of being blacklisted by search engines. Therefore, characterising spambots is the important step toward controlling and eliminating Spam 2.0.

In this paper we investigate spambot behaviour by employing Self-Organizing Maps (SOM). SOM organises multi-dimensional data into a two-dimensional map. We feed the SOM with three feature sets and generate a map for each feature set. Each map reveals different characteristics of spambot behaviour. We run 9 experiments based on these three feature sets to characterize spambot behaviour. The main contributions of this paper are as follow.

- Utilising self-organising maps to profile spambot behaviour.
- Proposing three feature sets *action set*¸ *action time*, *action frequency* to formulate spambot behaviour.
- Evaluating the performance of our proposed framework with real world data.

The rest of paper is structured as follow. In section 2 we present insights into the problems caused by spambots and importance of eliminating them in addition to providing background knowledge. Section 3 illustrates our proposed framework for characterising spambot behaviour. Section 4 presents our experimental results based on 9 different experiments. We formulate our conclusion in Section 5.

2. BACKGROUND

Spam is defined as mass, unsolicited and commercial content [9]. From simple text-based email messages to spam in Internet telephony systems, the purpose of spamming is the same i.e. to attract users to view spam content that generates advertising and revenue [10].

For achieving such purposes, spammers look at any type of communication used by humans and employ various methods to attract their attention. Spam is currently prolific because it is comparatively cheaper and easier to spread spam content rather than to detect spam [11]. As the Internet evolves spam techniques will also evolve and branch into different types of media. Figure 1 illustrates the different domains that are currently targeted by spammers.

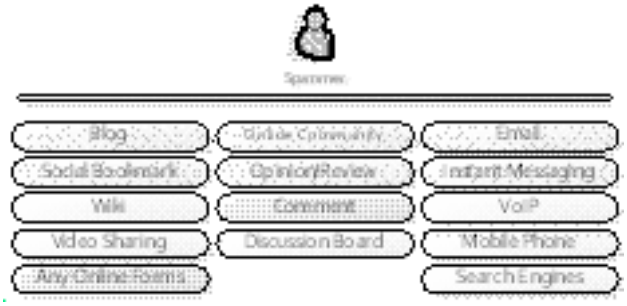


Figure 1 Different types of domain that have been targeted by spammers.

The main disadvantages of spam are as follows:

- *Wastes network bandwidth and storage / memory space*: Spam takes up valuable resources such as hard-disk space, Internet quota for spreading junk content.
- *Frustrates users:* For instance, spam contains fake information along with a link to spam-related websites. Spam occasionally contains adult-related content, which can offend users.
- *Misleads search engine spiders:* Spam content on the Web abuses Search Engine Optimisation (SEO) techniques to get improved and underserved search rankings. It manipulates search engine results against a set of keywords and builds up their page rank scores by creating many links to their website. Hence, poor quality and junk content receive more attention through higher rankings than genuine and high quality content.

2.1 Spambot Countermeasures

To deal with spambots, some *anti-robot* techniques are being used by a number of websites. In this section we recall some of the common techniques and explain their limitations.

2.1.1 IP Address

A simple way to detect web robot is to examine IP address. A list of known web robot IP addresses can be retrieved from some sources[1]. However, it is hard to maintain an up-to-date list of IP addresses since new web robots are being deployed as well as some spambots can be deployed on legitimate hosts with legitimate IP address. Hence, this technique is an inefficient way of trying to stop spambots.

2.1.2 Robots.txt

Known as *Robot Exclusion Protocol* is a text file normally inside root directory of websites. It contains list of access restriction places (e.g. webpages, directories, etc) for the website that forbid web robots from accessing them [12]. According to this protocol, web robots should examine robots.txt file whenever they visit a

[1] IP Addresses of Search Engine Spiders: http:/www.iplists.com/

website. Figure 2 is an example of in the robots.txt file. It forbids all web robots from accessing whole website.

```
User-agent: *
Disallow: /
```

Figure 2 Sample entry in robots.txt file. It keeps out all web robots.

However, pursuing Robot Exclusion Protocol is voluntary and many web robots do not follow this protocol [13]. As spambots are tasked to distribute promotional and junk content to as many websites as possible, it is doubtful that they would observe the robot rules stated in the robots.txt file. Therefore, this technique is ineffective at dealing with spambots.

2.1.3 User-agent

User-agent is a field inside a *HTTP Request Header* to the web server that identifies the client application. A web robot should declare its identity to the web server by specifying this field [14]. For instance the user-agent for a Yahoo! web crawler is *Slurp* [15]. By simply examining the user-agent field, web administrators are able to restrict access for some specific web robots. However, spambots often hide their identity or rename their user-agent to a name that is not restricted [8]. Therefore this technique is ineffective at restricting spambots.

2.1.4 Head Request

The response of web server on the *HTTP Head request* is header information without a message body. Web robots use *head request* to check the validity of hyperlinks, accessibility and recent modifications on the requested webpage [16]. Therefore, large amount of *head requests* in the incoming traffic can show web robots activity. However, this heuristic is not effective at detecting spambots as they normally request the message body rather than the header.

2.1.5 Referrer

Referrer is a field inside HTTP request that contains a link to the webpage a client followed to reach to the current requested webpage. For instance if a user reaches to www.example/page2.html from www.example.com page, the referrer filed is www.example.com. Ethical web robots normally do not assign any value to referrer field so they can be differentiated from human users. However, spambots often assign a value to referrer field to mimic a human user and can also provide fake links to conceal their navigation [17].

2.1.6 Flood Control

Flood control is a technique to limit the number of requests a client can send within a specified time interval. The idea is to restrict the number of submission requests to the server by spambots [18]. For example, once user creates a new thread in a forum they may have to wait 10 minutes before creating another thread. This technique may slow down automatic submissions by spambots but it also causes inconvenience for genuine human users. Additionally, such a technique can not stop spambots but only delay their submission process. We discovered from our experiment that spambots often create multiple user accounts and would therefore bypass this technique.

2.1.7 Nonce

Nonce is technique to stop automated form submission. Nonce is a random generated set of characters that are placed on a webpage with a form [18]. When the user submits the form, the nonce is sent the server. If the nonce does not exist in the form submission, it reveals that form was not loaded by the client and indicates the possibility of an automated attack by a spambot. This technique only ensures that submitted content originates from a web form loaded by the client but can not stop spambots that submit content through the website forms.

2.1.8 Form variation

This technique involves varying objects inside a form upon webpage request [18]. For example, the name of input fields within a form changes for each user webpage request. The idea of this technique is similar to nonce, in that it wants to ensure a server form generated is used during submission. However, like nonce, it can not block spambots that use the website forms.

2.1.9 Hashcash

The idea behind hashcash is to put increased cost of submission in client side in order to make spreading spam more costly [19]. This technique involves the sender calculating a stamp for submitted content. The calculation of the stamp is difficult and time-consuming but comparatively cheap and fast for receiver to verify. Similar to flood control, hashcash can only slow down but not stop spambot activity.

2.1.10 Completely Automated Public Turing test to tell Computers and Human Apart (CAPTCHA)

CAPTCHA is the most popular anti-robot technique adopted by many websites. It is a challenge response technique usually in format of a distorted image of letters and numbers [20]. Users are asked to infer a CAPTCHA image and type its letters in a form. However, CAPTCHA places inconvenience on users as it wastes their time, causes distraction, is unpleasant and at times scary. A number of recent works have reported approaches to defeat CAPTCHAs automatically by using computer programs [21–23]. CAPTCHA's drawbacks include the following:

1. Decrease user convenience and increase complexity of human computer interaction.

2. As programs become better at deciphering CAPTCHA, the image may become increasingly difficult for humans to decipher.

3. As computers get more powerful, they will be able to decipher CAPTCHA better than humans.

Therefore, CAPTCHA is a short term solution that may not prove effective in the future. Spambots armed with anti-CAPTCHA tools are able to bypass this restriction and spread spam content easily while the futile user inconvenience remains.

2.2 Web Usage Data

User navigation through websites can be tracked and logged. This knowledge can be stored and it is referred as web usage data [24]. Such knowledge can be mined to extract valuable information such as user navigation and click-stream. Generally, web usage data includes information about client IP Address, requested object (e.g. webpage), method of request (e.g. POST, GET), time of request and web browser type (e.g. Internet Explorer, Mozilla Firefox, etc). We evaluate web usage data to study spambots behaviour in Web 2.0 platform.

3. FRAMEWORK – CHARACTERISATION OF SPAMBOT BEHAVIOUR USING KOHONEN'S SELF ORGANIZING MAPS

3.1 Preliminary Concepts

In this paper we are going to employ SOM to characterise spambot behaviour. So before we start describing the actual solution, we would first cover the basics of SOM.

3.1.1 Self Organising Map (SOM)

The SOM is a type of neural network based on competitive learning. SOM can be employ for clustering and visualisation of high dimensional data. The output of the SOM is a map which is visual representation of input vector. Maps can be generated in two or three dimensions but two-dimensional maps are more in common.

SOM has two phases, which are the training and mapping phases. In the training phase, an input vector in form of a one-dimensional array (sometimes 2 dimensional) is presented to the map. Next, the weights of the components on the map (called nodes) which are closer to the input vector gain more strength for neighbouring nodes. The node which has closet distance to the input vector is called *Best Matching Unit* (BMU). Therefore, in the mapping phase, similar input vectors are in same region and can be grouped together as a cluster.

3.1.2 SOM algorithm

SOM starts by initialising each node's weight vector, w_i, on the map by assigning a small random variable. Next, an input vector x is presented to the map and distance between all nodes on the map and x is calculated as

$$d_j = \|x_i - w_j\| = \sqrt{\sum_{k=0}^{|w|} (x_{ik} - w_{kj})^2} \qquad (1)$$

The minimum d_j is selected as BMU. After finding BMU, SOM updates w_i for BMU and its neighbours nodes according to E.q 2 so they get closer to input vector.

$$w_i(t+1) = w_i(t) + \alpha(t) h_{ci}(t)[x - w_i(t)] \qquad (2)$$

where t is the time-step, $h_{ci}(t)$ is a *Gaussian* neighbourhood kernel and decreases with time $t\alpha(t) = \alpha(0)\frac{(1-t)}{T}$ is linear learning rate, T is training length (neighbourhood size less than number of nodes in one dimension of map).

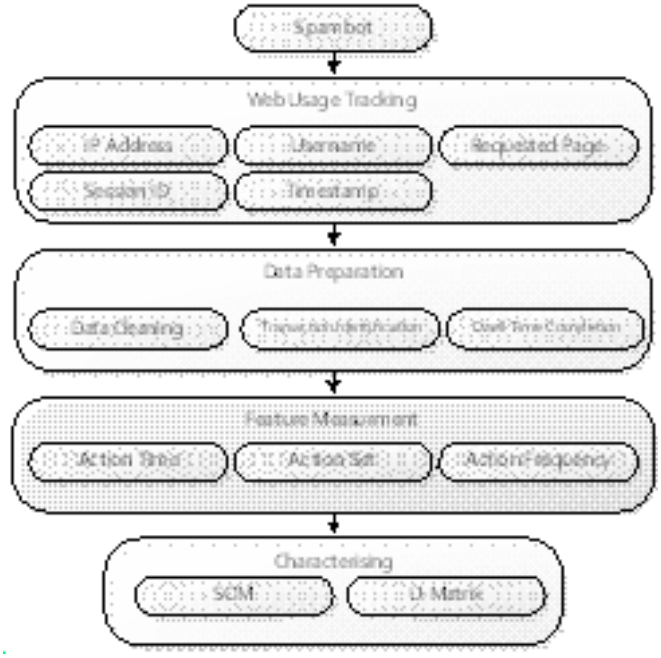


Figure 3 Spambot Characterising Framework.

3.1.3 Unified distance matrix (U-Matrix)

U-matrix visualises the distance between adjacent nodes in SOM. It can be employed to determine the number of clusters in SOM as well as how close they are to each other. In this paper, we utilise Kohonen's SOM to characterise spambots by monitoring their web usage data. In other words, we use SOM and the U-matrix as a visualisation tool to cluster different spambot web usage behaviour.

3.2 Overview

Our proposed framework for profiling spambots has 4 major modules as illustrated in Figure 3. In order to study spambot behaviour, we use the data collected from our previous work, HoneySpam 2.0 [8], which contains spambot web usage data. However, this data requires preparation for it to be used for spambot characterisation. Hence we propose three new feature sets – *action time*, *action set*, and *action frequency*. Our main intention to choose feature sets which are:

1. *Generic*: so they can be extend to other form of Web 2.0 platforms
2. *Differentiable:* so it can distinguish spambot behaviour from humans.

We define *action* as a set of user activity in a system to achieve a certain goal. For instance, in a forum, a user can navigate to the registration page, complete the registration form and click the submit button in order to create a new user account. This procedure can be formulated as the *register a new account* action.

Action set refers to set of actions that have been performed. *Action time* refers to the amount of time (dwell time) spent on a particular action. *Action frequency* refers to the number of times a user performs a particular action. We provide a formal description of each feature set in Section 3.4.

In order to characterise spambot behaviour, we employ SOM to organise each feature set on a map and visualise them by using the U-matrix. By studying the centre node inside each cluster we attempt to understand spambot behaviour from our dataset.

3.3 Web usage tracker

This module tracks spambot interaction with system. It stores the *IP address, username, requested webpage URL, session identity,* and *timestamp* of each request a spambot has made to the website. This information makes it possible to track spambot behaviour for each browsing session.

Conventionally, web usage navigation tracking is done through web server logs that described in Section 2. However, these logs do not specify usernames and the session identifier for each request. Therefore, we use our own web usage tracker that we developed in our previous work, *HoneySpam 2.0* [8]. Our tracker is designed to monitor spambots within an online forum void of any human activity. Therefore, data collected in this module consist of solely spambot web usage data.

3.4 Data preparation

This module comprises of three components which include data cleaning, transaction identification and dwell time completion.

Data cleaning

This component removes irrelevant web usage data such as:

- Data related to researchers who monitor the forum.
- Data related to visitors who did not create a user account.
- Data related to crawlers and other Web robots that are not spambots.

Transaction Identification

This component involves activities needed to construct meaningful clusters of user navigation data [24]. We group web usage data into three levels of abstraction – IP, User and Session. The highest level of abstraction is IP and each IP address can be used by multiple users. The middle level is the user level and each user can perform multiple browsing sessions. Finally, the lowest level is the session level, which contains detail information of the user navigation data in each website visit.

In our proposed framework, we define a transaction for all three level of abstraction to study the characteristic of spambot from each level. Figure 4 illustrates the association relationship between the IP, user and session levels.

Dwell time completion

Dwell time is an amount of time a user spends viewing a particular webpage in a session. As described in Section 3.1, web usage tracking module records the timestamp for each request. Hence, dwell time can be define as

$$d'_i = t_{i+1} - t_i \ \ where\ 1 \leqslant i \leqslant |S| \tag{3}$$

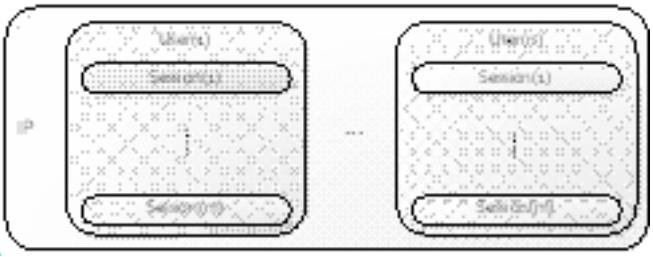


Figure 4 Association relation among IP, User and Session transitions.

d'_i is dwell time for ith requested webpage in session S at time t;

However, according to E.q. 3. it is not possible to calculate the last visited page in a session. For example, a user navigates to the last page on the website then closes their web browser or navigates to different website. Hence we calculate the average dwell time spent on other webpages in the same session as the dwell time for the last page.

3.5 Feature Measurement

The feature measurement module formulates web usage data to be used in the SOM to reveal spambots characteristics. It formats the data into three different feature sets which are action set, action time and action frequency.

Definition 1: **Action Set ($\overrightarrow{aS}$)**

Given a set of webpages $W = \{w_1, w_2, ..., w_{|W|}\}$, A is defined as a set of *Actions*, such that

$$A = \{a_i \mid a_i \subset W\} = \{\{w_l, ..., w_k\}\} \quad 1 \leqslant l, k \leqslant |W| \tag{4}$$

Respectively s_i is defined as

$$s_i = \{a_j\} \quad 1 \leqslant i \leqslant |T|; \quad 1 \leqslant j \leqslant |A| \tag{5}$$

s_i refers to a set of actions performed in transaction i and T is total number of transactions.

In order to build the input vector we assign each action, a_i, as a feature. Hence, we represent an action set as a bit vector

$$\overrightarrow{aS} = \left\langle v^i_1, ..., v^i_{|A|} \right\rangle \tag{6}$$

where

$$v^i_j = \begin{cases} 1 & a_j \in s_i \\ 0 & otherwise \end{cases} \tag{7}$$

Definition 2: Action Frequency ($\overrightarrow{aF} = \langle h^i_1, ..., h^i_{|A|} \rangle$)

Action frequency is a vector where h^i_j is the frequency of j^{th} action in s_i. otherwise it is zero.

Definition 3: Action Time ($\overrightarrow{aT} = \left\langle d^i_1, ..., d^i_{|A|} \right\rangle$)

We define action time as a vector where

$$d^i_j = \begin{cases} \frac{\sum_{k \in a_j} d'_k}{h^i_j} & a_j \in s_i \\ 0 & otherwise \end{cases} \tag{8}$$

d^i_j is a dwell time for action a_j in s_i which is equal to total amount of time spend on each webpage inside a_j. In cases that a_j occurs more once, we divide d^i_j by the action frequency,h^i_j to calculate the average dwell time. Table 1 shows a sample action time input vector.

Table 1 Sample Dwell Time Input Vector for Class1 Transactions.

	a_1	a_2	…	$a_{\|A\|}$
d_1	5	4	…	0
d_2	0	3.5	…	0
…	…	…	…	…
$d_{\|T\|}$	0	2	…	1

Table 2 All Possible Combinations of Features

	$\overrightarrow{aT}$	$\overrightarrow{aF}$	$\overrightarrow{aS}$
Session (S)	(S, $\overrightarrow{aT}$)	(S, $\overrightarrow{aF}$)	(S, $\overrightarrow{aS}$)
User (U)	(U, $\overrightarrow{aT}$)	(U, $\overrightarrow{aF}$)	(U, $\overrightarrow{aS}$)
IP (I)	(I, $\overrightarrow{aT}$)	(I, $\overrightarrow{aF}$)	(I, $\overrightarrow{aS}$)

Where S, U and I are set of all session, users and IP addresses.

3.6 Characterising

We present 9 combinations of the three feature sets and three level of abstraction as present in Table 2 to be used in our experiments with SOM.

We used u-matrix to project result of SOM and study the cluster. We choose centre node of each cluster to describe the characteristic of that particular cluster.

4. EXPERIMENTAL SETUP

4.1 Data Collection

For the purpose of this experiment we used the dataset that we collected in [8]. Our data consist of 19,348 records from an operational online forum.

4.2 Data Pre-processing

In our dataset, 11,039 records were generated from spambots while the rest contained irrelevant information as mentioned in Section 3.3 that we removed.

We group the data into three transaction levels which are session, user and IP address levels. The result of this task produced 3,726 distinct sessions, 1,067 spambot users, and 871 unique IP addresses. Figure 5 represents spambot action frequency for each action for the three transaction levels. To construct our input vectors, we extract 11 actions from our dataset. The details of each action are presented in Table 3. Accordingly we make $\overrightarrow{aS}$, $\overrightarrow{aF}$, and $\overrightarrow{aT}$ based on each action.

4.3 Experiments

We run our experiment based on the 9 feature sets discussed in Section 3.5 and we summarise the result of each experiment in their corresponding transaction level. We visualise SOM data in the U-matrix. In the following section we explain results from each experiment and the characteristic of the centre node inside each cluster.

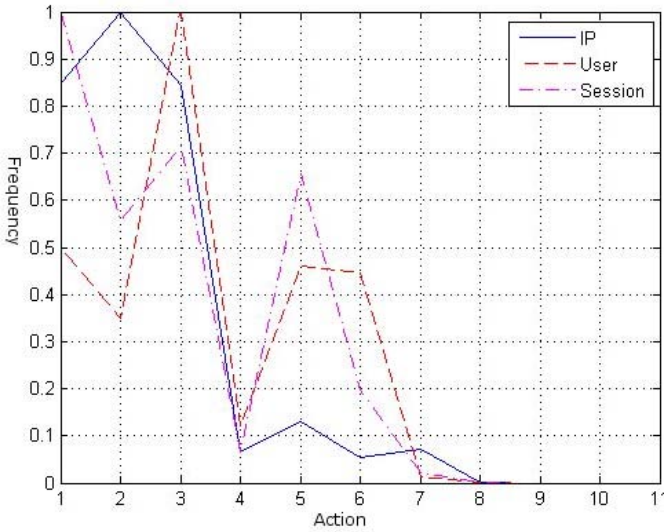


Figure 5 Spambots action frequency per each action for different transaction level. X axis indicates action A to K. Y axis shows the frequency of each action.

Table 3 Summery of Actions by Spambots.

Action Index	Description
A	View root page
B	View topics
C	Start new topic
D	View topic (view their own created topic)
E	View user profile information
F	Edit user information
G	Start new poll
H	User authentication (Login)
I	Reply to topic
J	Recently submitted topics
K	View overall forum statistic

5. EXPERIMENTAL RESULTS

5.1 Experiment 1: Session level result

Based on three experiments on session level it reveals the detailed behaviour of each spambot every time they visited our system. Figure 6 illustrates u-matrix for (S, $\overrightarrow{aF}$). The dark line represents large distances between nodes. The light node areas represent nodes that are close to each other. The dark lines separate the map into 3 major clusters. The details of each cluster are shown in Table 4.

Three major clusters which we label as *Content Submitters*, *Profile Editors* and *Mixture*.

- Cluster 1 (Content Submitter). Spambots in this cluster perform the *Starting New Topic* action. These bots did not perform any other action in this level hence the dwell time for their action cannot be calculated. This behaviour reveals that spambots do not navigate through website. They are programmed to directly submit the content. Additionally,

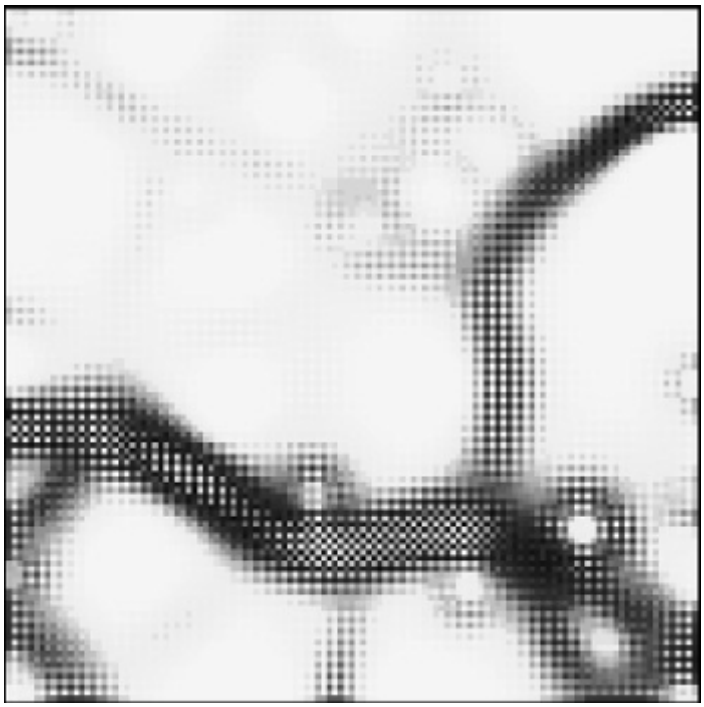


Figure 6 Result of U-matrix for session and frequency of actions. The dark lines are where the distance between nodes is large. In the light area, nodes are close to each other, which represent a cluster. The dark line separates the map into 3 major clusters.

Table 4 Major Characteristic of Spambots in the Session Level.

	Cluster 1	Cluster 2	Cluster 3
Dwell time	N/A	2,3	4,6,7
Frequency	1	2,3	4,9
Actions	C, A	AE, BB, BCB	AEFF, AEFFABCBD

in their action set there is no request for action C, hence it shows they do not interact with server form in order to submit their content. We believe that simple rules such as blocking direct content submission can prevent this abnormal behaviour in the website and stop spambots.

- Cluster 2 (Profile Editor). The goal of spambots in this cluster is to edit their own profile page. Profile page consist of various information such as *name*, *email address*, *homepage URL*, *signature*, etc. Spambots navigate from action A (view root page) to action F (edit user information) with a frequency of one for each action. The dwell time for their action is between 2-3 seconds. It points to the fact that spambots did not use the website forms to submit their profile details, since their dwell time is quite low. By modifying the profile information spambots are able to create a link farm and spread spam content. For instance they specify a link inside *profile signature field* to their other campaigns / websites.

- Cluster 3 (Mixture). Spambots in this cluster performed both above mentioned set of actions. They navigate through following action sequence (A,E,F,F,A,B,C,B,D). The average frequency for each action is 2, which means they perform the above sequence two times in each session. The average dwell time is between 4 – 7 seconds. This behaviour again shows spambots did not use form to submit content or modify their profile, since this sequence cannot feasibly be performed in 4 – 7 seconds. Additionally they navigate to view their submitted content in order to increase the view count of their topic, as well as to possibly ensure their submitted content is published.

Table 5 Major Characteristic of Spambots in the User Level.

	Cluster 1	Cluster 2	Cluster 3
Dwell time	N/A	2.3	2.5
Frequency	1	8	11
Actions	C, CC	AEFFABCB	AAEFABCBBD

5.2 Experiment 2: User level results

User level tracking gives knowledge about characteristics of each spambot for their total active time in the forum. It can show whether or not their behaviour has changed over time. The results of three different experiments in the user level are:

- Cluster 1 (Content Submitter). Spambots use their user account to only submit spam content directly to the website similar to the discussion in Section 5.1. However spambots use their username once since there is only one session belongs to each user in this cluster. The dwell time is zero (unknown as there is only 1 page request), the frequency of the action is either one or two and they only perform action C.

- Cluster 2 (Content Submitter and Profile Editor). Spambots start their action navigation by visiting the root page, modifying profile page and then submit spam content. Their average dwell time is 2.5 seconds for each action with a total frequency of 8 actions per user level. Their navigation sequence is as follows (A,E,F,F,A,B,C,B). Once the actions are performed, the spambots did not log in and use the user account again.

- Cluster 3 (Mixture). In this cluster, the average dwell time of spambots was 7 seconds for navigating through (A,A,E,F,F,A,B,C,B,B,D) with a frequency of 11. They mix both profile editing and content submission behaviour. They viewed their own created topic at the end of their navigation set. Same as previous clusters, spambots just have one session for each user account in this cluster.

Investigation of spambot in user level reveals that once they created a user name they just use it once and they do not change their behaviour during their life time. Table 5 presents detail of each cluster. Figure 7 illustrates U-matrix map for (U, $\overrightarrow{aF}$). It shows majority of spambot at the user level have similar characteristic as the map reveals dominated by light areas.

5.3 Experiment 3: IP level results

IP level presents a high level view of spambot behaviour. The experiment resulted in the discovery of three major clusters which include:

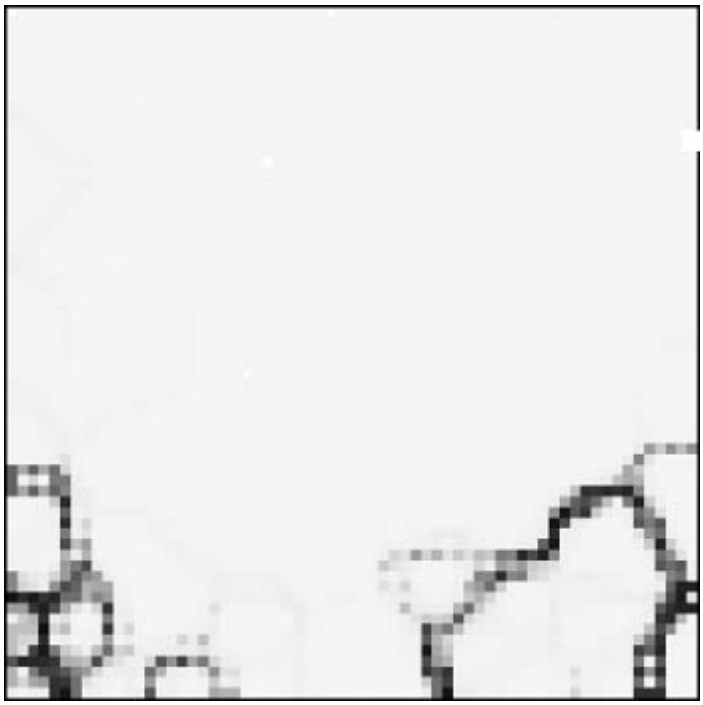


Figure 7 U-matrix representation of user level and action frequency. White squires on the map show the amount of nodes that are close together. Larger squares represent more nodes.

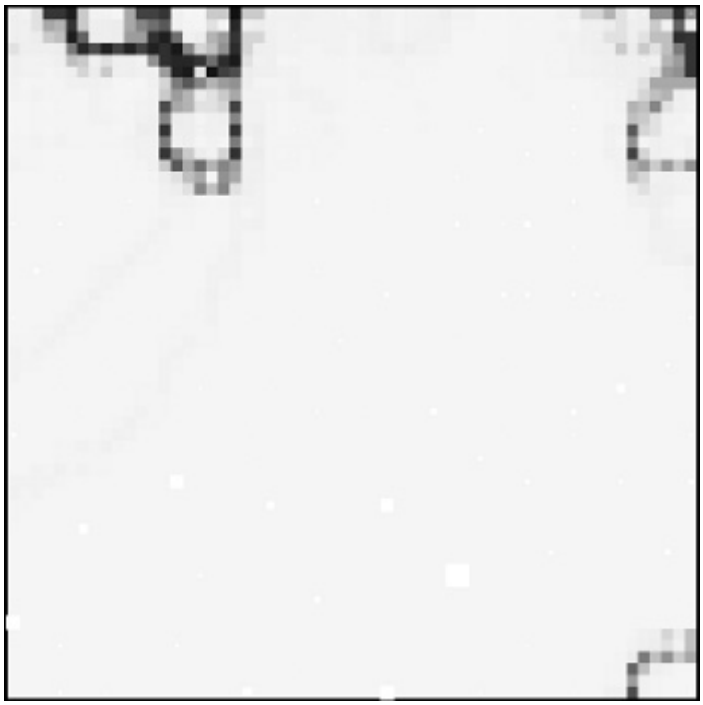


Figure 8 U-Matrix for IP level and action frequency.

Table 6 Major Characteristic of Spambots in the IP Level.

	Cluster 1	Cluster 2	Cluster 3
Frequency	8	11	100+
Actions	AEFFABCB	D…	AEFABC…

- Cluster 1 (Mix). This cluster consists of 4 – 5 frequency for each action set. Spambots in this cluster navigate through the root webpage, modify their profile and submit spam content. Finally they end their navigation by viewing their own created content. The amount of frequency shows that they created multiple accounts under one IP address name to perform their actions.
- Cluster 2 (Content Viewer). We discovered this strange cluster of spambots. Spambots in this cluster only navigate to action *D (viewing their own topic)*. This behaviour was not seen in the previous levels. Further investigation revealed that spambots change their IP address once they submit the spam content. Therefore some spambots use multiple IP address for one user.
- Cluster 3 (Big Hit Mix). Action frequency in this cluster is more than 100 for submitting spam content or to modify profiles. This cluster has the similar characteristic to cluster 1. However the higher action frequencies show that spambots use multiple username to perform the same actions.

The experiment of IP address dwell time map does not result in clear clusters. The reason maybe due to change of IP address by spambots at the end of navigation which result in unknown dwell time as well as multiple username per IP address that result in various dwell time. Table 6 presents details of each experiment at IP level. Figure 8 represents u-matrix map of IP level and action frequency.

6. DISCUSSION & LIMITATION

The U-matrix visualisation in some of our experiments produced a number of minor clusters. While we investigated these minor clusters as well, they did not result into distinct behaviour in useful for spambot characterisation. For example, a cluster was identified for spambots who perform action *A* (visiting the root page). This action is general and does not reveal any interesting characteristic. Hence we did not report such minor clusters in our experiments.

Self-organising map and neural networks in general do not allow much user interpretability in understanding why nodes belong to each cluster. It can be viewed as a black box algorithm which is fed some inputs and generates the output. Hence one challenge in future work can be investigation of differences among clusters.

The data we collected for this work can be extended in future to evaluate spambots in other platforms such as blogs, wikis and other types of Web 2.0 applications.

7. RELATED WORKS

There has been enormous amount of work dedicated to spam detection and filtering in recent years. This section will review the important and related areas in the existing literature.

In the area of web usage mining and web robot detection, Tan et al. [25] proposed a rule-based framework to discover search engine crawlers and camouflaged web robots. They utilised navigation patterns such as session length, set of visited webpages, and requested method type (e.g. GET, POST, etc).

Park et al. [17] presented a malicious web robot detection method based on type of *HTTP request* and existence of *mouse movement.* However, the focus of both these research was on general web robot detection rather than spambot detection.

In Honeyspam 2.0 [8] we proposed our web tracking module to track spambot data. The focus of this work was to develop a framework to gather spambot data rather than characterising it.

Jan et al. [26]'s work showed a *proactive* approach to track current follow of spam messages inside *botnets*. They monitor email spambot controllers to gather recent spam messages and use them inside spam-filtering technique to block such messages.

Yiquen et al. [27] and Yu et al. [28] employ user web access data to classify spam webpages from legitimate webpages. The main assumption in their framework is to rely on user web access data.

Based on all the existing work in the areas of spambot detection, we are confident that no one has so far undertaken in depth research in the area of characterising spambot behaviour. In this paper, we have moved in this direction in order to provide a stepping stone for the development of next generation spam filtering technology.

8. CONCLUSION

In this paper, we presented a number of characteristics of spambot behaviour within forums (a Web 2.0 platform) by utilising Kohonen's self-organising maps. This study forms the basis for research into next generation spam filtering techniques. We propose a framework to profile spambot behaviour. We employed web usage data and proposed three unique feature sets – action set, action frequency and action time to study spambots. Our main intention is to propose feature sets that are generic so it is applicable to other Web 2.0 platforms and differentiable to distinguish spambots and humans based on usage behaviour.

We grouped the data based on three level of abstraction – session, user and IP address and we conducted 9 experiments. The main observations are as follows:

- Spambots focus on particular actions such as submitting content, updating profiles detail, etc.
- pambots do not reuse a single user account as they register new user accounts frequently because they do not want their user account to get blacklisted.
- Some spambots directly post spam content to the website and they do not interact with input forms.
- Spambots are designed to repeat limited set of tasks and not a large range of tasks.
- Spambots can be categorised into four different categories based on four different goals i.e. content submitters, profile editors, content viewers and mixed behaviour.
- Some spambots change their IP address once they complete an action and are preparing on performing another action.

Our promising results suggest that our technique is capable for application in identifying spambot for Web 2.0 applications. We plan to extend this work in the future by proposing techniques to distinguish spambot and human users by evaluating their web usage data.

REFERENCES

1. P. Hayati and V. Potdar, "Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods " in *7th IEEE International Conference on Industrial Informatics* Cardiff, Wales, 2009.
2. P. Kolari, A. Java, and T. Finin, "Characterizing the Splogosphere," in *Proceedings of the 3rd Annual Workshop on the Weblogging Ecosystem*, 2006.
3. Live-Spam-Zeitgeist, "Some Stats, Akismet," [Accessed online by May 2009] http://akismet.com/stats/, 2009.
4. A. Zinman and J. Donath, "Is Britney Spears spam," in *Fourth Conference on Email and Anti-Spam* Mountain View, California, 2007.
5. J. Nitin and L. Bing, "Opinion spam and analysis," in *Proceedings of the international conference on Web search and web data mining* Palo Alto, California, USA: ACM, 2008.
6. T. Uemura, D. Ikeda, and H. Arimura, "Unsupervised Spam Detection by Document Complexity Estimation," in *Discovery Science*, 2008, pp. 319-331.
7. S. Webb, J. Caverlee, and C. Pu, "Social Honeypots: Making Friends with a Spammer Near You," in *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA, 2008.
8. P. Hayati, K. Chai, V. Potdar, and A. Talevski, "HoneySpam 2.0: Profiling Web Spambot Behaviour," in *12th International Conference on Principles of Practise in Multi-Agent Systems*, Nagoya, Japan, 2009, pp. 335-344.
9. Z. Le, Z. Jingbo, and Y. Tianshun, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, pp. 243-269, 2004.
10. S. Cobb, "The Economics of Spam," [Accessed: 9 Aug 09] EPrivacyGroup Website, http://www.eprivacygroup.com, 2003.
11. B. Whitworth and E. Whitworth, "Spam and the social-technical gap," *Computer*, vol. 37, pp. 38-45, 2004.
12. M. Koster, "A Standard for Robot Exclusion [Accessed Online Dec 2009]," in *http://www.robotstxt.org/orig.html*, 1994.
13. M. Koster, "Robots in the Web: threat or treat? [Accessed Online Dec 2009]," in *http://www.robotstxt.org/threat-or-treat.html*, 1995.
14. M. Koster, "Guidelines for Robot Writers [Accessed Online Dec 2009]," in *http://www.robotstxt.org/guidelines.html*, 1993.
15. Yahoo!, "How to Prevent Your Site or Certain Subdirectories From Being Crawled [Accessed Online Dec 2009]," in *http://help.yahoo.com/l/us/yahoo/search/webcrawler/*, 2009.
16. R. Fielding, U. Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1 - 9 Method Definitions [Accessed Online Dec 2009]," in *http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html*, 1999.
17. K. Park, V. S. Pai, K.-W. Lee, and S. Calo, "Securing Web Service by Automatic Robot Detection," *USENIX 2006 Annual Technical Conference Refereed Paper*, 2006.
18. U. Ogbuji, "Real Web 2.0: Battling Web spam," [Accessed: 3 Aug 09] http://www.ibm.com/developerworks/web/library/wa-realweb10/, 2008.
19. D. Mertz, "Charming Python: Beat spam using hashcash," [Accessed: 3 Aug 09] http://www.ibm.com/developerworks/linux/library/l-hashcash.html, 2004.
20. L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in *Advances in Cryptology — EUROCRYPT 2003*, 2003, pp. 646-646.
21. H. Abram, W. G. Michael, and C. H. Richard, "Reverse Engineering CAPTCHAs," in *Proceedings of the 2008 15th Working Conference on Reverse Engineering - Volume 00*: IEEE Computer Society, 2008.

22. K. Chellapilla and P. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)," in *NIPS*, 2004.
23. G. Mori and J. Malik, "Recognizing objects in adversarial clutter: breaking a visual CAPTCHA," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. I-134-I-141 vol.1.
24. R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, 1997, pp. 558-567.
25. P.-N. Tan and V. Kumar, "Discovery of Web Robot Sessions Based on their Navigational Patterns," *Data Mining and Knowledge Discovery*, vol. 6, pp. 9-35, 2002.
26. G. Jan, bel, H. Thorsten, and T. Philipp, "Towards Proactive Spam Filtering (Extended Abstract)," in *Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* Como, Italy: Springer-Verlag, 2009.
27. L. Yiqun, C. Rongwei, Z. Min, M. Shaoping, and R. Liyun, "Identifying web spam with user behavior analysis," in *Proceedings of the 4th international workshop on Adversarial information retrieval on the web* Beijing, China: ACM, 2008.
28. H. Yu, Y. Liu, M. Zhang, L. Ru, and S. Ma, "Web Spam Identification with User Browsing Graph," in *Information Retrieval Technology*, 2009, pp. 38-49.

Rule-Based On-the-fly Web Spambot Detection Using Action Strings

Pedram Hayati
Anti Spam Research Lab
Digital Ecosystems & Business Intelligence Institute, Curtin Business School, Curtin University, Perth, Western Australia, Australia
p.hayati@curtin.edu.au

Vidyasagar Potdar
Anti Spam Research Lab
Digital Ecosystems & Business Intelligence Institute, Curtin Business School, Curtin University, Perth, Western Australia, Australia
v.potdar@curtin.edu.au

Alex Talevski
Anti Spam Research Lab
Digital Ecosystems & Business Intelligence Institute, Curtin Business School, Curtin University, Perth, Western Australia, Australia
a.talevski@curtin.edu.au

William F. Smyth
Algorithms Research Group,
Department of Computing & Software
McMaster University, Hamilton, Ontario, Canada L8S 4K1
smyth@mcmaster.ca[1]

ABSTRACT

Web spambots are a new type of internet robot that spread spam content through Web 2.0 applications like online discussion boards, blogs, wikis, social networking platforms etc. These robots are intelligently designed to act like humans in order to fool safeguards and other users. Such spam content not only wastes valuable resources and time but also may mislead users with unsolicited content. Spam content typically intends to misinform users (scams), generate traffic, make sales (marketing/advertising), and occasionally compromise parties, people or systems by spreading spyware or malwares.

Current countermeasures do not effectively identify and prevent web spambots. Proactive measures to deter spambots from entering a site are limited to question / response scenarios. The remaining efforts then focus on spam content identification as a passive activity. Spammers have evolved their techniques to bypass existing anti-spam filters.

In this paper, we describe a rule-based web usage behaviour action string that can be analysed using Trie data structures to detect web spambots. Our experimental results show the proposed system is successful for on-the-fly classification of web spambots hence eliminating spam in web 2.0 applications.

1. INTRODUCTION

A fake profile in an online communities, an unsolicited comment in blog, a commercial unwelcome thread in an online discussion boards etc are example of new form of spamming techniques called spam 2.0 [1, 2]. Spam 2.0 offers a far more attractive proposition for spammers as compared to traditional spam specifically email spam.

Spammers can discover web 2.0 applications and use automated tools to distribute spam information that is targeted at a demographic of their choice with very little resistance. A single spam 2.0 attack may reach many targeted and domain specific users and online messages typically cannot be deleted by regular users and persist until an administrator deals with them often impacting many users in the meantime.

Spam 2.0 posts also have a parasitic nature. They may exist on legitimate and often official websites. If such information persists, the trust in such pages is diminished, spam is effectively promoted by trusted sources, many users can be mislead or lead to scams and computer malware and such legitimate sites may be blacklisted which then deprives all others of legitimate content. As a result of the success and impact rates of spam 2.0, it is far more popular amongst spammers and has far greater negative socio-economic impact.

Such spamming techniques promote the use of Web Spambots (or simply spambots): a web crawler that navigates the World Wide Web with the sole purpose of planting unsolicited content on external websites. To date, as Web 2.0 platforms are getting more prevalent, spam 2.0 are becoming more and more widespread. In order to avoid being spuriously used in this way (and to eliminate the clutter and wasted time created by spambots), websites seek tools to recognize and deter spambots as they arrive. Usually the tools used for this purpose are based on challenge-response techniques, such as Completely Automated Public Turing test to

CEAS 2010 – Seventh annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, July 13-14, 2010, Redmond, Washington, US

[1] The work of the third author was supported in part by the Natural Sciences & Engineering Research Council of Canada.

tell Computers and Human Apart (CAPTCHA), a well-known technique typically in the form of an image used to block web crawlers access to a website [3]. However such techniques are about to expire as it decreases human users' convenience and computers are getting more and more powerful day by day [2, 4, 5].

In this paper we propose a rule-based combinatorial approach that distinguishes automatically between spambots and regular human users based on their usage patterns:

- Introduce the idea of using web usage behaviour to investigate spambots behaviour on the Web,
- Propose a new concept – Action Strings, a feature set extracted from web usage behaviour to model spam users vs. human users behaviour,
- Illustrate a novel rule-based classifier using Trie structure [6-8] for fast and on-the-fly spambot detection,

Here we describe the experience with a prototype system developed by us. In the next section we discuss about two major concepts – web usage data and trie structure that we used in our proposed system

2. BACKGROUND

2.1 Web Usage Data

Users browsing websites navigate through web objects such as web pages, image, files, documents etc. Such data can be recorded and later mined to extract valuable information [9]. This tracking data is referred to as web usage data. Web usage data has been widely employed in many studies to understand visitor browsing patterns, make personalised websites, improve web performance and to implement decision support and recommendation systems [10] [11]. Web usage data may contain the following fields:

- The visited URL,
- The referrer URL,
- Timestamp,
- IP address,
- Browser/Operating System identity.

In our work, we associate two additional attributes to each entry – a unique session identity (session ID) and a user account name of a user who makes a request. The former makes it possible to track a user while the user is visiting a website, while the latter includes their username in each record to make it possible to track user activities over a period of time.

2.2 Trie

A trie structure is a way to store and retrieve information. Its main advantages over other techniques are: ease of updating and handling, shorter access time, and removing redundancies in the data structure [6]. A trie is in the form of a tree structure where each node contains the value of the key it is associated with.

In this paper we construct a trie structure which consists of human and spambot behaviour patterns. The trie gives fast on-the-fly pattern matching ability and we use that for spambot pattern detection.

3. PROBLEM

To set the scene for this paper, we begin with a brief overview of the problems in spam 2.0 and spambot detection. A number of solutions have been proposed to classify spam in web 2.0 environments such as: opinion spam detection [12], spam in social networks [13-15], spam in video sharing websites [1, 16]. Most of these solutions have focussed on one particular form of spam. By extracting features from the content of spam 2.0, such techniques try to make spam content separate from legitimate content. However most of them did not come up with satisfactory classification results. Hence, this leads us to investigate different approaches to identify spam 2.0.

This research continues in line with our previous work on combating spam 2.0 [2]. Our main idea is to actively investigate spambot – as an origin of spam 2.0 instead of seeking to discriminate attributes inside spam content.

The concept of spambot identification has not been investigated thoroughly. Hence, we first provide a formal definition for this problem.

Similar to the spam classification problem [17] spambot detection can be formalised as a binary classification problem as follows

$$U = \{u_1, u_2, ..., u_{|U|}\} \tag{1}$$

Where U is set of users visiting a website, u_i is the i^{th} user in a set.

$$C = \{c_h, c_s\} \tag{2}$$

Where C refers to the overall set of users, c_h refers to human user class and c_s refers to spambot user class. the binary classification $\phi(u_i, c_j)$ is

$$\phi(u_i, c_j) : U \times C \rightarrow \{0,1\} \tag{3}$$

Where

$$\phi(u_i, c_j) = \begin{cases} 1 & u_i \in c_s \\ 0 & otherwise \end{cases} \tag{4}$$

Since in spambot detection problem each u_i belongs to one and only one class, the decision function can be simplified as $\phi(u_i)_{spam} : U \rightarrow \{0,1\}$.

4. HUMAN BEHAVIOUR VS. SPAMBOT BEHAVIOUR

The main assumption of our proposed method is that human web usage behaviour is intrinsically different from spambot behaviour. The reason is that spambots have different intentions. Spambots visit the website mainly to spread spam content rather than to consume the content. Hence by investigating and mining web usage data it is possible to distinguish spambots from human users. This viewpoint is different from previous studies which have focused on content and meta-content based features [1].

123.123.123.123 - - [10/Jul/2009:00:19:25 +0800] "GET /forum/index.php?action=post;board=1 HTTP/1.0" 200 7852 "http://example.com/forum/index.php?action=post;board=1" "Opera/9.0 (Windows NT 5.1; U; en)"

123.123.123.123 - - [10/Jul/2009:00:19:30 +0800] "GET /forum/index.php?board=1 HTTP/1.0" 200 6248 "http://example.com/forum/index.php?board=1" "Opera/9.0 (Windows NT 5.1; U; en)"

(a) Spambot HTTP header requests

111.111.111.111 - - [15/Aug/2009:07:05:10 +0800] "GET /forum/index.php HTTP/1.0" 200 60 "http://example.com/" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

111.111.111.111 - - [15/Aug/2009:07:06:12 +0800] "GET /forum/index.php?board=1 HTTP/1.0" 200 6248 "http://example.com/forum/index.php" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

111.111.111.111 - - [15/Aug/2009:07:08:32 +0800] "GET /forum/index.php?action=post;board=1 HTTP/1.0" 200 7852 "http://example.com/forum/index.php?board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

111.111.111.111 - - [15/Aug/2009:07:10:27 +0800] "GET /forum/index.php?topic=1397 HTTP/1.0" 200 524 "http://example.com/forum/index.php?action=post;board=1" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET CLR 3.5.30729)"

(b) Human HTTP header requests

Figure 1. Series of spambot http requests (a) vs. human http requests (b) in an online forum.

Figure 1 illustrates a simple series of HTTP requests sent from both humans and spambots to submit new content to an online forum. Each entry consists of an IP address, timestamp, request method, the requested URL, the response status code, the referrer URL and browser identity. There are some obvious differences between spambot and human requests. For example, the human's sequence of requests is more expected or on average, spambot sessions last for some seconds while human sessions last for 5 minutes etc. This factor shows that spambots follow pre-programmed and optimised procedures. Such key differences in web usage behaviour can be utilised to distinguish humans and spambots.

5. ACTION STRINGS

In order to model web usage data into a behavioural model, we propose an action as a set of user efforts to achieve certain purposes (E.q. 5). For example, in an online forum, in order to create a new user account, the user needs to navigate to the registration page, complete the registration form and submit this information. This procedure can be formulated as "Register a new user account" action. Actions abstract web usage data. Actions can be a suitable discriminative feature to model user behaviour and can also be extendible to many other Web 2.0 platforms.

Given a set of webpages $W = \{w_1, w_2, ..., w_{|W|}\}$, A is defined as a set of Actions, such that

$$A = \{a_i \mid a_i \subset W\} = \{\{w_l, ..., w_k\}\} \quad 1 \le l, k \le |W| \tag{5}$$

Respectively s_i is defined as

$$s_i = (a_j) \quad 1 \le i \le |T|; \quad 1 \le j \le |A| \tag{6}$$

s_i refers to a sequence of actions called *Action Strings*, performed in session i and T is total number of sessions.

6. RULE-BASED ON-THE-FLY SPAMBOT DETECTION METHOD

Figure 2 provides an overview of the monitoring strategy in algorithmic form. Each user (u) has multiple sessions and each session contains a series of performed actions (s_i). For every new incoming u our algorithm goes down through the trie to find matching nodes. If the probability for that specific action string (1- $P_H(s_i)$) is higher than a given threshold, our algorithm classifies the incoming u as a spambot.

for every new user *u* do
    location ← root of trie
    for every action *i* do
        location ← (*location*, *i*)
        $P_H(s_i)$ ← eval(*location*)
        if 1− $P_H(s_i)$ > *threshold* then
            zap(*u*)
    until *u* exits

Figure 2. Overview of user monitoring strategy

6.1 Framework

We propose a rule-based on-the-fly spambot detection framework by using the trie data structure. Our framework consists of five steps as follows:

6.1.1. Step1: Monitoring web usage data. Track web usage data and the navigation stream. For each incoming HTTP

request, IP address, visited URL, referrer URL, browser identity, timestamp, session ID and user account name are tracked.

6.1.2. Step2: Data cleaning and preparation. Once data is aggregated through Step 1, it needs to be cleansed of irrelevant information such as visitors who did not create a user account in the system. Next, data needs to be grouped into meaningful user navigation clusters. This task is known as Transaction Identification [9]. As discussed in Section 2.1, we accompany each tracking record with a unique session ID. Each session ID can assist us to track a user's navigation. In our proposed framework we group data based on session IDs. Therefore, requests made by a user in a single visit are clearly identifiable. For example, a user may visit a system three times in a day, each time our proposed framework assigns a unique session ID to the user. Hence, it is possible to track user requests in each visit. Moreover, defining transactions at the session level can be utilised for fast and on-the-fly classification as well as to provide in-depth insights into the user behaviour.

6.1.3. Step3: Action extractions and formulation. Table 1 provides a summary of different actions that can be performed by users in our system. Each action – defined in Section 5 – comes along with an index key that is used to formulate a user's interaction type. Such action lists can be extended to other web 2.0 applications and can be defined beforehand for a specific platform. Upon new incoming traffic, our framework assigns associated actions. Later, by putting each action index key together, action strings can be build up as defined in Section 5. Action strings grow over time as the user performs actions, as well as preserving the order of the action sequence. As mentioned in Section 4, such a sequence can be utilised as a discriminative feature to distinguish humans and spambots

Table1. Action Index Key and Action Lists

Action Key	Description
A	View root page
B	View topics
C	Start new topic
D	View topic (view their own created topic)
E	View user profile information
F	Edit user information
G	Start new poll
H	User authentication (Login)
I	Reply to topic
J	View recently submitted topics
K	View statistic
L	Spell checking
M	Send private message
N	Search
O	Get new topics
P	Update topic
Q	Preview topic

6.1.4. Step4: Building up a trie data structure. Once the action strings for both human and spambots have been created, our framework builds a trie data structure based on each action string. Each trie edge contains an action key index and each node contains the probability of a specific action string being either human or spambot. This probability is computed through E.q 7 and E.q. 8.

$$P_H(s_i) = \frac{f_H(s_i)}{f_H(s_i) + f_B(s_i)} \quad (7)$$

$$P_B(s_i) = \frac{f_B(s_i)}{f_H(s_i) + f_B(s_i)} \quad (8)$$

where s_i is an action string, $f_H(s_i)$ is the frequency of human action strings that have prefix s_i. $f_B(s_i)$ is the frequency of spambots with prefix s_i, $P_H(s_i)$ is the probability of s_i being human and $P_B(s_i)$ of its being spambot, respectively.

Zero probability means that that particular action string has never occurred before. For example Figure 3 illustrates a trie data structure for a set of {ABC, ABCD, ABDE, ABC} action strings for a human and {ABD, ABD, ABDE} action strings for a spambot.

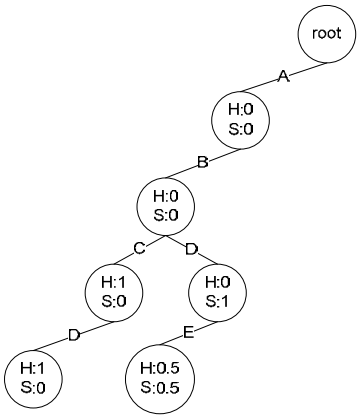


Figure 3. Simple trie data structure for set of {ABC, ABCD, ABDE, ABC} for a human and {ABD, ABD, ABDE} for a spambot. S and H represent probability of action string belonging to spambots and humans respectively.

6.1.5. Step5: Classification. New incoming action string s_i is validated through trie structure that has been built in Step4. After validation s_i can fall into two categories – Match and Not-Matched. In the former case, our framework looks at $P_H(s_i)$ and if $1 - P_H(s_i) > Threshold$, s_i would be classified as spambot. In the latter situation no decision is made; this is a focus of our future work.

6.2 Performance Measurement

We used Matthews Correlation Coefficient (MCC) method to measure the performance of our proposed framework [18]. MCC is one of the best performance measurement methods of binary classifications especially when the data among two classes of data is not balanced [19]. It considers true and false positives and returns a value between -1 and +1. If the return value is closer to +1 the classification result is better and the decision can be

considered to have greater certainty. However, if the result value is close to 0 it shows the output of the framework is similar to random prediction. A result value closer to -1 shows a strong inverse ability of the classifier. MCC is defined as follows;

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (9)$$

In Eq. 9, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives.

7. EXPERIMENTAL RESULTS & DISCUSSION

7.1 Dataset

According to our understanding, there is no publicly available collection, which contains both human and spambot web usage data for web 2.0 platform. Hence, we use the same dataset that we collected in our previous study [2]. We collected human user data from a human moderated online forum and our spambot data from our HoneySpam 2.0 project [2], which runs same online forum application with the same configurations as human forum. Each entry in our dataset contains visited URL link, referrer URL link, timestamp, browser identity, and IP address; and it is associated with session ID and forum username. Our dataset contains 16594 entries consisting of 11039 spambots records and 5555 human records. In the action extraction and formulation step we come up with 34 individual actions. We used 2/3 of the data for feeding the trie in Building up a trie data structure step and use the remaining 1/3 for evaluation purpose.

7.2 On-The-Fly Detection

Our proposed framework has a real-time detection or on the fly detection feature. The system creates action strings as they happen, i.e. based on the user webpage behaviour. The aim is to identify the spam behaviour pattern in the least amount of actions and then flag that transaction as spambot. We make a window over test action strings, run our classifier and increase the window's size by one character for the next run. Such situations can simulate real world practices where user action strings grow over the time.

7.3 Results

We make five random datasets (*DS1* to *DS5*) and run our framework based on each dataset. The window size ranges from 2 to 10 characters (the length of the longest action string in our dataset is 10). Additionally we vary the threshold from -0.05 to 0.05. We measure the performance of our framework by using MCC for each experiment.

Figure 4 illustrates the accuracy of our proposed system for different window sizes on our five random datasets. The best accuracy of 96% was achieved on DS1 with window size equal to or larger than 4 (Table 3), while overall accuracy is 93% over all datasets. Table 2 summarises average accuracy for our 5 random datasets.

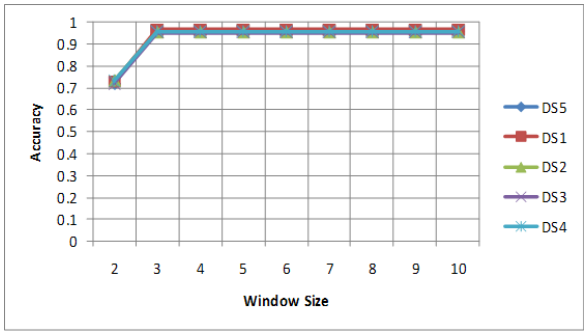


Figure 4. Accuracy of our proposed system for different windows size on 5 random datasets. Threshold maintains at zero for all experiments.

Table2. Average Accuracy of datasets

	DS1	DS2	DS3	DS4	DS5
Accuracy	0.939	0.930	0.926	0.933	0.934

Table3. Accuracy of different windows sizes on our datasets

	DS1	DS2	DS3	DS4	DS5
Len 2	0.727	0.734	0.719	0.732	0.721
Len 3	0.965	0.955	0.952	0.958	0.960
Len 4	0.965	0.955	0.952	0.958	0.960
Len 5+	0.966	0.955	0.952	0.958	0.960

Figure 5 presents MCC value for different windows sizes. Overall average MCC on all datasets is 0.744. Table 4 shows the value of MCC on different datasets. The best MCC value 0.802 was achieved on DS1 for window size equal to or larger than 5 (Table 5).

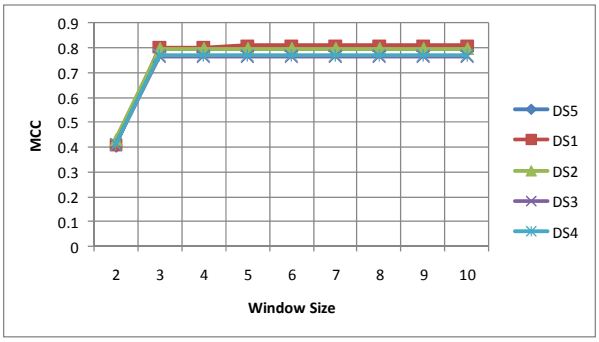


Figure 5. MCC values of different window size on 5 random datasets.

Table4. Average MCC of datasets

	DS1	DS2	DS3	DS4	DS5
MCC	0.762	0.754	0.725	0.729	0.749

Table5. MCC value of different windows size on our datasets

	DS1	DS2	DS3	DS4	DS5
Len 2	0.406	0.430	0.412	0.410	0.405
Len 3	0.802	0.795	0.764	0.769	0.792
Len 4	0.802	0.795	0.764	0.769	0.792
Len 5+	0.808	0.795	0.764	0.769	0.792

Figure 6 illustrates the performance of our proposed system based on different threshold at different window sizes (*Len2* to *Len6*).

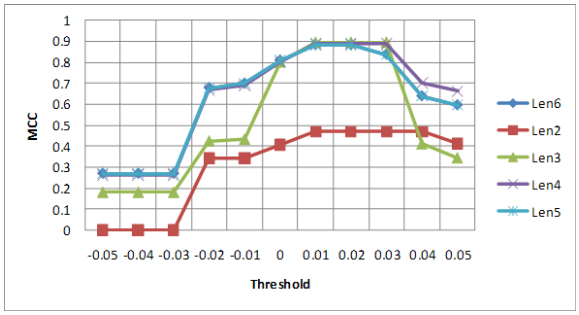


Figure 6. Performance of system based on different thresholds for different window size (Len2 to Len6)

The best MCC value of 0.88 is achieved for thresholds between 0 and 0.01 for window size equal to or larger than 4 (*Len4* to up).

The performance of our proposed system becomes better as window size grows. This shows that our system can predict better as user uses the system over time. However, the performance remains the same after some windows size. The reason is that our datasets are randomly selected and not all the action string combinations are included during Step 4, *Building up a trie data structure*. Hence some of the action strings are not included in trie structure and our system may not be able to match incoming unseen action string. The same behaviour can be seen in the accuracy of our results (Fig. 4), where accuracy remains the same after a certain window size (window size 4).

In Figure 6 it is obvious that moving the threshold toward negative values increases the number of false-negatives while moving toward positive values increases the number of false positives. Therefore, MCC on such values is low.

8. RELATED WORKS

Although the topic of spam has been investigated extensively, to the best of our knowledge research into spambot detection on Web 2.0 applications and spam 2.0 is quite young and has not received comprehensive attention. In this section we review some of the important literature in this area.

Tan et al. [20] propose a web robot session identification method based on their navigational patterns. The main assumption in their proposed system is that web robot navigational patterns such as session length and set of visited webpages (width and depth of visited webpages) is different from those of humans. The aim of their study is on unknown and camouflaged web robots and web crawlers. Park et al. [21] provide a method for malicious web robot detection based on types of requests for web objects (e.g. Cascading Style Sheet files, image files) and existence of mouse/keyboard activity. However, both the above-mentioned studies did not focus on spambot detection in web 2.0 applications where a spambot can mimic human user behaviour.

A proactive spam filtering approach has been proposed by Göbel et al. [22]. Their proposed framework includes interaction with spam botnet controllers which can provide the latest spam messages. Later, it can present a template for current spam runs to improve spam filtering techniques.

Jindal and Liu [12] study opinion spam in review-gathering websites. They propose a machine learning approach based on 36 content-based features to differentiate opinion spam from legitimate opinion. Zinman and Donath [13] attempted to create a model to distinguish spam profiles from legitimate ones in Social Networking Services. Their machine learning based method uses content-based features to do the classification. Benevenuto et al. [16] provide a mechanism to identify video spammers in online social network by means of a Support Vector Machine classifier against content-based features. Heymann et al. [15] survey spam filtering techniques on the social web and evaluate a spam filtering technique on a social tagging system. Most of the above studies focus on one particular type of spam and are limited to the content attributes of that particular domain. Moreover, they do not study the source of the spam problem, i. e. the spambot.

Yu et al.[23] and Yiqun et al. [24] categorise spam webpages from legitimate webpages by employing user web access logs. Their framework relies on user web access logs as a trusted source for classifying webpages.

Last but not least, in HoneySpam 2.0 [2] we proposed our web tracking framework to track spambot data. We develop a framework for accumulating spambot web usage data rather than for detecting spambots.

9. CONCLUSION AND FUTURE WORKS

Research in the area of web spambot detection in Web 2.0 platform is quite young. Most of the current studies have focussed on one particular type of spam rather than a general solution to block spammers. We aim to detect web spambots as a source of spam problems on the Web 2.0 platform. Hence our solution can be extended to other web applications. This paper provides a rule-based on-the-fly web spambot detection method. Our method is based on web usage behaviour. We extract discriminative features called action strings from web usage data to classify spambot vs. human. We propose action as a set of user efforts to achieve certain purposes and action strings as a sequence of actions for a particular user in a transaction. In order to make a real-time and on-the-fly classification method we build a trie data structure based on action strings.

We evaluate our method against an online forum and achieved average accuracy of 93% on spambot detection. We measure the performance of our system by using MCC. The average MCC value of 0.744 is achieved on our 5 randomly selected datasets.

As future work, we intend to improve our detection method and develop an adaptive mechanism to make a real-time update to the system. Additionally, we will evaluate our proposed method against other Web 2.0 platforms such as social networking websites, wikis, and blogs.

10. REFERENCES

[1] P. Hayati and V. Potdar, "Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods " in *7th IEEE International Conference on Industrial Informatics* Cardiff, Wales, 2009.

[2] P. Hayati, K. Chai, V. Potdar, and A. Talevski, "HoneySpam 2.0: Profiling Web Spambot Behaviour," in *12th International Conference on Principles of Practise in Multi-Agent Systems*, Nagoya, Japan, 2009, pp. 335-344.

[3] A. Luis von, B. Manuel, and L. John, "Telling humans and computers apart automatically," *Commun. ACM*, vol. 47, pp. 56-60, 2004.
[4] K. Chellapilla and P. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)," in *NIPS*, 2004.
[5] Y. Jeff and A. Ahmad Salah El, "Usability of CAPTCHAs or usability issues in CAPTCHA design," in *Proceedings of the 4th symposium on Usable privacy and security* Pittsburgh, Pennsylvania: ACM, 2008.
[6] F. Edward, "Trie memory," *Commun. ACM*, vol. 3, pp. 490-499, 1960.
[7] R. M. Donald, "PATRICIA\—Practical Algorithm To Retrieve Information Coded in Alphanumeric," *J. ACM*, vol. 15, pp. 514-534, 1968.
[8] M. Kurt, "Compressed tries," *Commun. ACM*, vol. 19, pp. 409-415, 1976.
[9] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, 1997, pp. 558-567.
[10] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the World Wide Web," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 11, pp. 94-107, 1999.
[11] J. Dean and M. R. Henzinger, "Finding related pages in the World Wide Web," *Computer Networks*, vol. 31, pp. 1467-1479, 1999.
[12] J. Nitin and L. Bing, "Opinion spam and analysis," in *Proceedings of the international conference on Web search and web data mining* Palo Alto, California, USA: ACM, 2008.
[13] A. Zinman and J. Donath, "Is Britney Spears spam," in *Fourth Conference on Email and Anti-Spam* Mountain View, California, 2007.
[14] S. Webb, J. Caverlee, and C. Pu, "Social Honeypots: Making Friends with a Spammer Near You," in *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA, 2008.
[15] H. Paul, K. Georgia, and G.-M. Hector, "Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges," *IEEE Internet Computing*, vol. 11, pp. 36-45, 2007.
[16] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, C. Zhang, and K. Ross, "Identifying Video Spammers in Online Social Networks," in *AIRWeb '08* Beijing, China, 2008.
[17] Z. Le, Z. Jingbo, and Y. Tianshun, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, pp. 243-269, 2004.
[18] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochim Biophys Acta*, vol. 405, pp. 442-451, 1975.
[19] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, pp. 412-424, May 1, 2000 2000.
[20] P.-N. Tan and V. Kumar, "Discovery of Web Robot Sessions Based on their Navigational Patterns," *Data Mining and Knowledge Discovery*, vol. 6, pp. 9-35, 2002.
[21] K. Park, V. S. Pai, K.-W. Lee, and S. Calo, "Securing Web Service by Automatic Robot Detection," *USENIX 2006 Annual Technical Conference Refereed Paper*, 2006.
[22] G. Jan, bel, H. Thorsten, and T. Philipp, "Towards Proactive Spam Filtering (Extended Abstract)," in *Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* Como, Italy: Springer-Verlag, 2009.
[23] H. Yu, Y. Liu, M. Zhang, L. Ru, and S. Ma, "Web Spam Identification with User Browsing Graph," in *Information Retrieval Technology*, 2009, pp. 38-49.
[24] L. Yiqun, C. Rongwei, Z. Min, M. Shaoping, and R. Liyun, "Identifying web spam with user behavior analysis," in *Proceedings of the 4th international workshop on Adversarial information retrieval on the web* Beijing, China: ACM, 2008

Web Spambot Detection Based on Web Navigation Behaviour

Pedram Hayati, Vidyasagar Potdar, Kevin Chai, Alex Talevski
Anti-Spam Research Lab (ASRL)
Digital Ecosystem and Business Intelligence Institute
Curtin University, Perth, Western Australia
{pedram.hayati, kevin.chai}@postgrad.curtin.edu.au
{v.potdar, a.talevski}@curtin.edu.au

***Abstract*— Web robots have been widely used for various beneficial and malicious activities. Web spambots are a type of web robot that spreads spam content throughout the web by typically targeting Web 2.0 applications. They are intelligently designed to replicate human behaviour in order to bypass system checks. Spam content not only wastes valuable resources but can also mislead users to unsolicited websites and award undeserved search engine rankings to spammers' campaign websites. While most of the research in anti-spam filtering focuses on the identification of spam content on the web, only a few have investigated the origin of spam content, hence identification and detection of web spambots still remains an open area of research. In this paper, we describe an automated supervised machine learning solution which utilises web navigation behaviour to detect web spambots. We propose a new feature set (referred to as an action set) as a representation of user behaviour to differentiate web spambots from human users. Our experimental results show that our solution achieves a 96.24% accuracy in classifying web spambots.**

Keywords— Web spambot detection, Web 2.0 spam, spam 2.0, user behaviour

I. INTRODUCTION

Spammers do not only deploy their own spam webpages (known as *Web spam*) but they spread spam content over Web 2.0 applications such as online communities, wikis, social bookmarking, online discussion boards etc. [1]. Web 2.0 collaboration platforms like online discussion forums, wikis, blogs, etc. are misused by spammers to distribute spam content. This new spamming technique is called *Spam 2.0* [1]. Examples of Spam 2.0 would include spammers posting promotional threads in online discussion boards, manipulating *wiki* pages, creating fake and attractive user profiles in online community websites etc [1].

According to live reports by [2], the amount of comment spam on the Web has doubled within the past year. To date, spammers exploit new tools and techniques to achieve their purposes. An example of such a tool is a *Web spambot* (simply *spambot*), which is a type of web robot designed to spread spam content on behalf of spammers [3]. Spambots are able to crawl the web, create user accounts and contribute in collaboration platforms by spreading spam content [4]. Spambots do not only waste useful resources but also put the legitimate website in danger of being blacklisted, hence identifying and detecting spambots still remain to be an open area of research. It should be noted that Web spambots are different from spambots which are designed to harvest email address from webpages. For the sake of simplicity here we refer to web spambot as spambot.

Current countermeasures which solely focus on detection and prevention of spam content are not suitable enough to be used in a Web 2.0 environment [1]. For example, most of the content-based methods in email spam [5] or web spam techniques such as link-based detection [6], Trustrank [7], etc are not applicable in Web 2.0 environments since unlike web spam, Spam 2.0 content involves spammers contributing into legitimate website [1].

In this paper, we present a novel method to detect spambots inside Web 2.0 platforms (Spam 2.0) based on web usage navigation behaviour. The main contributions of this paper are to:

- Present a framework to detect spambots by analysing web usage data and evaluate its feasibility in combating the distribution of Spam 2.0.
- Propose an action set as a new feature set for spambot detection.
- Evaluate the performance of our proposed framework with real world data.

We make use of web usage navigation behaviour to build up our feature set and train our Support Vector Machine (SVM) classifier. Our preliminary results show that our framework is capable of detecting spambot with 96.24% accuracy.

The rest of paper is structured as follow.

- Section II gives an overview of the problems in spambot detection along with the problem definition.
- Section III presents our proposed framework for spambot detection.
- Data preparation and our experimental results are discussed in Section IV.
- We conclude the paper in Section V along with our future works.

II. PROBLEM

Spambots mimic human behaviour in order to spread spam content. To hinder spambots activities, most websites adopt *Completely Automated Public Turing test to tell Computers and Human Apart* (CAPTCHA) which is a popular challenge-response technique to differentiate web robots from humans [8]. However, CAPTCHA is not a suitable solution for stopping spambots and it inconveniences human users. Existing research shows that by making use of machine learning algorithm even CAPTCHA based techniques can be deciphered by programming code [9-11]. Other filtering techniques are content based i.e. focusing on spam content classification rather than spambot detection [1, 3]. The formal definition of the spambot detection problem is discussed in following section.

A. Problem Definition

The problem of spambot detection is a binary classification problem that is similar to the spam classification problem described in [12]. Suppose

$$D = \{u_1, u_2, ..., u_{|U|}\} \tag{1}$$

where,

D is a dataset of users visiting a website

u_i is the i^{th} user

$$C = \{c_h, c_s\} \tag{2}$$

where,

C refers overall set of users

c_h refers to human user class

c_s refers to spambot user class

Then the decision function is

$$\phi(u_i, c_j) : D \times C \rightarrow \{0,1\} \tag{3}$$

$\phi(u_i, c_j)$ is a binary classification function, where

$$\phi(u_i, c_j) = \begin{cases} 1 & u_i \in c_s \\ 0 & otherwise \end{cases} \tag{4}$$

In spambot detection each u_i belongs to one and only one class so, the classification function can be simplified as $\phi(u_i)_{spam} : D \rightarrow \{0,1\}$.

III. PROPOSED SOLUTION

A. Solution and Overview

While most of the research in anti-spam filtering has a focused on identification of spam content on the web, only a few have investigated the source of the spam problem [1, 4, 13-15]. One way of identifying the source of spam is to study spambot behaviour. In this paper our fundamental assumption is that spambot behaviour is intrinsically different from those of humans. In order to test this assumption, we make use of web usage data from both spambots and humans. Web usage data contains information regarding the way web users navigate websites and can be implicitly collected while a user browses the website. However, it is necessary to convert web usage data in a format that

- is discriminative and reliable feature set that differentiates spambot behaviour from humans
- can be extended to other platforms of Web 2.0 applications such as wikis, blogs, online communities etc.

Hence, we propose a new feature set called an *action set* to formulate web usage data. We define an *Action* as a set of user requested webpages to achieve a certain goal. For example, in an online forum, a user navigates to a specific board then goes to the *New Thread* page to start a new topic. This user navigation can be formulated as *submit new content* action. Table I presents some example of actions for different Web 2.0 platforms. We provide a formal description of action set in Section 3.2. Our results show that the use of action sets is an effective representation of user behaviour and can be successfully used to classify spambot and human users.

TABLE I. EXAMPLES OF USER ACTIONS IN WEB 2.0 PLATFORMS INCLUDING ONLINE DISCUSSION BOARDS (I.E. FORUMS), BLOGS, WIKIS, ONLINE COMMUNITIES

Platform	Action
Online Discussion Boards	Post a topic, Reply to a topic, Contact other users, etc.
Blogs (comment)	Read posts, Read others comment, Post a comment, etc.
Wikis	Start new topic, Edit topic, Search, etc.
Online Communities	Adding new friend, Sending message, Writing comments, etc.

B. Framework

Our proposed framework consists of 3 main modules – Tracker, Extractor, and Classifier as shown in Fig. 1. Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.

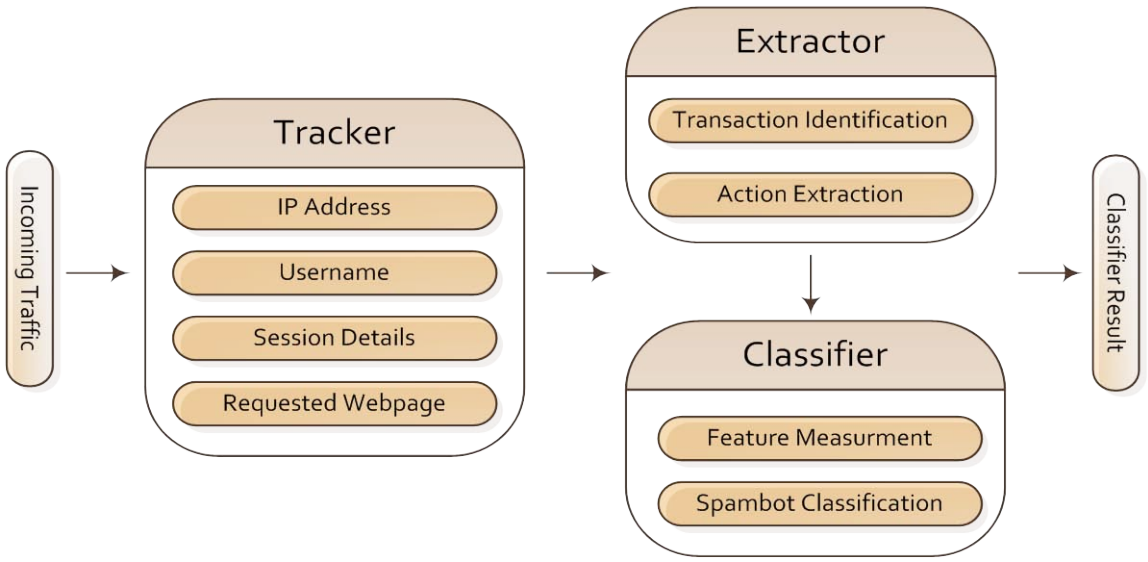


Figure 1. Architecture of Proposed Framework

*1) Incoming Traffic :*Represents a new user who enters the system via a web interface like the index page of a discussion board.

2) Tracker : This is the entry module for all incoming traffic and tracks *IP addresses*, *Usernames*, *Session Details*, and *Requested Webpages*. It starts by reading *HTTP Request* headers and extracts each of above attributes. A unique session is assigned for individual browsing session, so it is possible to track the navigation on each occasion when the user visits the website. The tracker module stores the gathered data along with corresponding username for each record.

3) Extractor : This is the second module of the proposed framework and involves two main activities which are transaction identification and action extraction.

a) Transaction Identification : Transaction identification involves creating a meaningful group of user requests [16]. In our framework, we group user requests based on three levels of abstraction. These levels range from the highest to the lowest level of abstraction (Fig. 2). At the highest level of abstraction, we have *IP address*, followed by *user* and at the lowest level of abstraction we have *session*.

The IP address which is at the highest level can contain more than one user. Inherently, at the user level, each user group can contain more than one session. The session level is the lowest level of abstraction and it contains information about user navigation data for each website visit. The user level of abstraction defines the total active time in a website spent by a user. The IP level can illustrate the behaviour of a specific host during the total amount of time that a host is connected to the website.

Figure 2. Levels of Web Usage Data Abstraction

In our proposed solution we chose the session abstraction level for the following reasons:

- session level can be built and analysed quickly while other levels need more tracking time to get a complete view of user behaviour.
- session level provides more in-depth information about user behaviour when compared with the other levels of abstraction.

Hence we define a transaction as a set of webpages that a user requests in each browsing session.

b) Action Extraction : Action extraction is the next activity in the Extractor module. Given a set of webpages $W = \{w_1, w_2, \ldots, w_{|W|}\}$ A is defined as a set of *Actions*, such that

$$A = \{a_l \mid a_l \subset W\} = \{\{w_l, \ldots, w_k\}\} \quad 1 \le l, k \le |W| \quad (5)$$

Respectively s_i is defined as

$$s_i = \{a_j\} \quad 1 \le i \le |U|; \quad 1 \le j \le |A| \quad (6)$$

s_i refers to a set of actions performed by user i .

4) Classifier : Classifier module is the third module in the proposed framework and involves two main activities, which are feature measurement and spambot classification.

a) Feature Measurement : Our framework extracts and measures features to build an input vector for spambot classification. In order to build the input vector, we consider each action (a_i) as a feature in our system. Suppose, there are n different actions, we represent input vector, $\vec{s_i'}$, as a bit vector (Eq. 7).

$$\vec{s_i'} = \{a_1^i, a_2^i, ..., a_n^i\} \tag{7}$$

where,

$$a_j^i = \begin{cases} 1 & a_j \in s_i \\ 0 & otherwise \end{cases} \tag{8}$$

b) Spambot Classification : In order to perform spambot classification, we used SVM on our extracted and measured features. SVM is selected as it has solid theoretical foundation, can handle high dimensional data better than most classifiers, supports non-linearity and can perform classification more accurately than other classifiers in applications involving relatively high dimensional data as in our experiment [17]. We utilise popular SVM tool known as LibSVM [18] for our spambot classification experiment.

5) Classifier Result : The result of the SVM classifier is the classification of user sessions into two classes i.e. spambots or humans.

The following section provides a detailed view of our algorithm along with the pseudo code.

C. Algorithm

Table II provides steps of our proposed framework for spambot classification. Our algorithm starts with a loop for each session, t_i . For each requested webpage in t_i , if the webpage is associated to a particular action, our algorithm looks for the next requested webpage in the session. If a group of webpages forms an action, a' , the system adds a' to the set of action performed by user k . Next, for each set of actions , s_k , the framework uses the classifier to mark the session as a spambot or human session and classifies the corresponding user k who made s_k as a spambot or human user.

TABLE II. PROPOSED METHOD ALGORITHM

1. Let T refer to the set of sessions.
2. for each $t_i \in T$ do
3. create new action a' .
4. for each $w_j \in t_i$ do
5. if $w_j \in a$ then
6. add w_j to a' .
7. if $a' \in A$ then
8. add a' to s_k .
9. remove all w_j in a' from t_i .
10. if $t_i . length > 0$ then
11. create new action a' .
12. else
13. continue to new w_i .
14. end.
15. end.
16. for each s_i do
17. if $\phi(s_k)_{spam} = 1$ then
18. mark s_i as spambot session and i as spambot user.
19. else
20. mark s_i as human session and i as human user.
21. end.

IV. EXPERIMENTAL RESULT

A. Data Preparation

Two major tasks in web data mining are *sessionsiation* and *user identification* from web server logs [21]. However we do not need to conduct these tasks as we track human and spambot navigation directly through the forum for each user account and for each of their sessions.

The spambot data used for our experiment is collected from our previous work [3] over a period of 60 days. Additionally, we host an online discussion forum for a human community that is under human moderation. We removed forum specific data such as the domain name, web server IP address, etc from both datasets. We grouped user navigation records based on sessions and extracted a set of actions from each dataset along with user accounts. We merge both datasets into a single dataset which contains the username and set of actions that belong to the particular user. We split the dataset into a training set (2/3) and a test set (1/3). Table III present a summery of our dataset (Table IV).

TABLE III. SUMMERY OF COLLECTED DATA

Data	Frequency
# of human records	5555
# of spambot records	11039
# of total sessions	4227 (training: 2818, test: 1409)
# of actions	34

As shown Table 5, there are 34 action navigations (34 columns) used as 34 feature and 2818 session (2818 rows) to train the SVM classifier.

TABLE IV. TRAINING DATA FORMAT USES IN THE SVM CLASSIFIER

	a_1	a_2	...	a_{34}
t_1	1	0	…	1
t_2	0	0	…	1
…	…	…	…	…
t_{2818}	0	1	…	0

B. *Performance Measurement*

We used *Matthew Correlation Coefficient (MCC)* method to measure the performance of our proposed framework [19]. MCC is one of the best performance measurement methods of binary classifications especially when the data among two classes of data are not balanced [20]. It considers true and false positives and returns a value between -1 and +1. If the return value is closer to +1 the classification result is better and the decision can be considered to have greater certainty. However, if the result value is close to 0 it shows the output of the framework is similar to random prediction. A result value closer to -1 shows a strong inverse ability of the classifier. MCC is defined as follows;

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (9)$$

In Eq. 9, TP is number of true positives, TN is number of true negatives, FP is number of false positives and FN in number of false negatives.

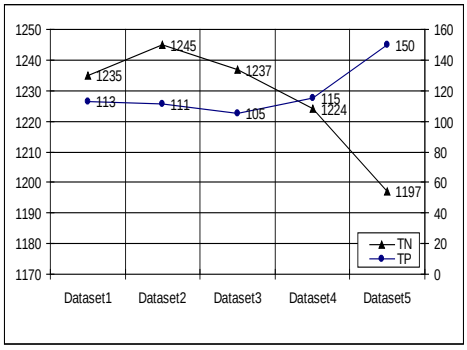
C. *Results*

We run our experiment on 5 randomly split training and test datasets. For each dataset we measure the accuracy and MCC value. The average accuracy 95.55% was achieved which is range from 95.03% on dataset 4 to 96.24% on dataset 2 as shown in Table V.

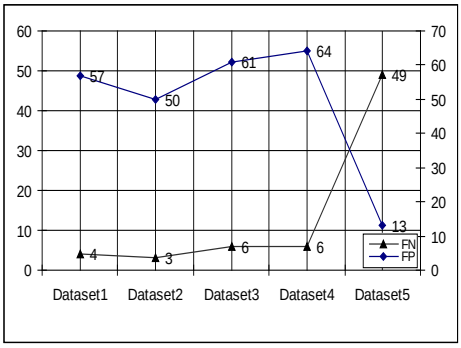
TABLE V. EXPERIMENT RESULT ON 5 RANDOM DATASETS

	Accuracy	**MCC**
Dataset 1	95.67%	0.780
Dataset 2	96.24%	0.801
Dataset 3	95.24%	0.751
Dataset 4	95.03%	0.757
Dataset 5	95.60%	0.809

Fig. 3 shows a comparison among the number of true negatives (correctly detected spambots), true positives (correctly detected humans), false negatives (spambots classified as humans) and false positives (humans classified as spambots) in each dataset. Although the highest accuracy was achieved by dataset 2, its MCC value is slightly lower than the MCC value corresponding to dataset 5. The reason is the number of false positives in dataset 5 is a quarter of those of dataset 2 as shown in Figure 2b.



(a)



(b)

Figure 3. **(a)** Left x Axis, True Negatives (TN), Right x Axis, True Positives (TP). **(b)** Left x Axis, False Megatives (FN) and Right x Axis, False Positives (FP)

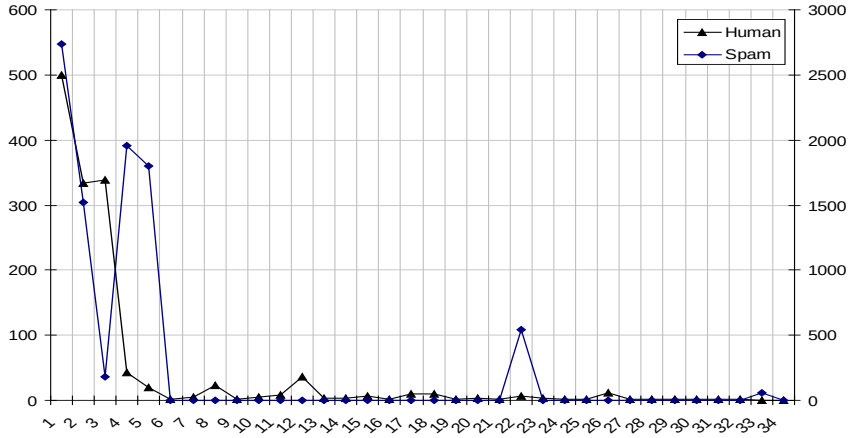


Figure 4. Frequency of Humans and Spambots. Left x Axis, Humans, Right x Axis, Spambots. Y Axis Represents Actions.

Figure 4 illustrates action frequencies of spambots and humans. A closer look at Figure 4 reveals that spambots are more active in action 5 and 6. These actions belong to *posting new thread* in online discussion forums. On the other hand humans are more active in actions 3 and 4 which belong to *read forums topics*.

V. Related Work

The identification of web robots has been of interest to the research community as robot navigation on websites add noise and interferes with web usage mining of human navigation. Tan et al. [22] presented a framework to detect search engine crawlers that are camouflaged and previously unseen web robots. In their proposed framework, they use navigation patterns such as the session length, depth and width of webpage coverage and request methods. Park et al. [23] proposed a malicious web robot detection method based on the request type in *HTTP* headers along with *mouse movement*. However the focuses of these studies were not on spambot detection.

In the area of email spambots, Göbel et al. [24] introduced a *proactive* approach to monitor current spam messages inside *botnets*. They interact with email spambot controllers to collect latest email spam messages and generate *templates* and employ them inside spam filtering techniques.

In the web spam domain, Yu [25] and Liu [26] proposed a framework based on user web access data to classify spam and legitimate webpages. Their framework is based on the assumption that user navigation behaviour on spam webpages is different from legitimate webpages. However, from our study we show that one cannot assume that navigation behaviour being evaluated is from a human user but could also be from a spambot.

VI. Conclusion and Future Work

While most of the research in the anti-spam filtering concentrates on the identification of spam content, we understand that the work we have proposed in this research is innovative by focusing on spambot identification to manage spam rather than analysing spam content. The importance of this approach (i.e. detecting spambots rather than spam content) is that it is a completely new approach to identify spam content and can be extended to other Web 2.0 platforms rather than only forums.

We proposed a novel framework to detect spambots inside Web 2.0 applications, which lead us to Spam 2.0 detection. We proposed a new feature set i.e. action navigations, to detect spambots. We validated our framework against an online forum and achieved 96.24% accuracy using the MCC method. In the future we will extend this work on a much larger dataset and improve our feature selection process.

References

[1] Hayati, P., Potdar, V.: Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods 7th IEEE International Conference on Industrial Informatics, Cardiff, Wales (2009)

[2] Live-Spam-Zeitgeist: Some Stats, Akismet. [Accessed online by May 2009] http://akismet.com/stats/ (2009)

[3] Hayati, P., Chai, K., Potdar, V., Talevski, A.: HoneySpam 2.0: Profiling Web Spambot Behaviour. 12th International Conference on Principles of Practise in Multi-Agent Systems, Vol. 5925. Lecture Notes in Artificial Intelligence, Nagoya, Japan (2009) 335-344

[4] Webb, S., Caverlee, J., Pu, C.: Social Honeypots: Making Friends with a Spammer Near You. Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008), Mountain View, CA (2008)

[5] Hayati, P., Potdar, V.: Evaluation of Spam Detection and Prevention Frameworks for Email and Image Spam - A State of Art. The 2nd International Workshop on Applications of Information Integration in Digital Ecosystems (AIIDE 2008), Linz, Austria (2008)

[6] Qingqing, G., Torsten, S.: Improving web spam classifiers using link structure. Proceedings of the 3rd international workshop on Adversarial information retrieval on the web. ACM, Banff, Alberta, Canada (2007)

[7] Zolt, n, G., ngyi, Hector, G.-M., Jan, P.: Combating web spam with trustrank. Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB Endowment, Toronto, Canada (2004)

[8] von Ahn, L., Blum, M., Hopper, N., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. Advances in Cryptology — EUROCRYPT 2003 (2003) 646-646

[9] Abram, H., Michael, W.G., Richard, C.H.: Reverse Engineering CAPTCHAs. Proceedings of the 2008 15th Working Conference on Reverse Engineering - Volume 00. IEEE Computer Society (2008)

[10] Baird, H.S., Bentley, J.L.: Implicit CAPTCHAs. Proceedings SPIE/IS&T Conference on Document Recognition and Retrieval XII (DR&R2005), San Jose, CA (2005)

[11] Chellapilla, K., Simard, P.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). NIPS (2004)

[12] Le, Z., Jingbo, Z., Tianshun, Y.: An evaluation of statistical spam filtering techniques. ACM Transactions on Asian Language Information Processing (TALIP) 3 (2004) 243-269

[13] Nitin, J., Bing, L.: Opinion spam and analysis. Proceedings of the international conference on Web search and web data mining. ACM, Palo Alto, California, USA (2008)

[14] Zinman, A., Donath, J.: Is Britney Spears spam. Fourth Conference on Email and Anti-Spam, Mountain View, California (2007)

[15] Uemura, T., Ikeda, D., Arimura, H.: Unsupervised Spam Detection by Document Complexity Estimation. Discovery Science (2008) 319-331

[16] Cooley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems 1 (1999) 5-32

[17] Liu, B.: Web Data Mining. Springer Berlin Heidelberg (2007)

[18] Chang, C., Lin, C.: LIBSVM: a library for support vector machines. In: http://www.csie.ntu.edu.tw/~cjlin/libsvm, S.a.a. (ed.): (2001)

[19] Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta 405 (1975) 442-451

[20] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16 (2000) 412-424

[21] Cooley, R., Mobasher, B., Srivastava, J.: Web mining: information and pattern discovery on the World Wide Web. Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on (1997) 558-567

[22] Tan, P.-N., Kumar, V.: Discovery of Web Robot Sessions Based on their Navigational Patterns. Data Mining and Knowledge Discovery 6 (2002) 9-35

[23] Park, K., Pai, V.S., Lee, K.-W., Calo, S.: Securing Web Service by Automatic Robot Detection. USENIX 2006 Annual Technical Conference Refereed Paper (2006)

[24] Jan, G., bel, Thorsten, H., Philipp, T.: Towards Proactive Spam Filtering (Extended Abstract). Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer-Verlag, Como, Italy (2009)

[25] Yu, H., Liu, Y., Zhang, M., Ru, L., Ma, S.: Web Spam Identification with User Browsing Graph. Information Retrieval Technology (2009) 38-49

[26] Yiqun, L., Rongwei, C., Min, Z., Shaoping, M., Liyun, R.: Identifying web spam with user behavior analysis. Proceedings of the 4th international workshop on Adversarial information retrieval on the web. ACM, Beijing, China (2008)

Hayati, P., Chai, K., Potdar, V., Talevski, A.: HoneySpam 2.0: Profiling Web Spambot Behaviour. 12th International Conference on Principles of Practise in Multi-Agent Systems, Vol. 5925. Lecture Notes in Artificial Intelligence, Nagoya, Japan (2009) 335-344

Hayati, P., Potdar, V.: Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods 7th IEEE International Conference on Industrial Informatics, Cardiff, Wales (2009)

Hayati, P., Potdar, V.: Spammer and Hacker, Two Old Friends. 3rd IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST 2009), Istanbul, Turkey (2009)

Hayati, P., Potdar, V.: Evaluation of Spam Detection and Prevention Frameworks for Email and Image Spam - A State of Art. The 2nd International Workshop on Applications of Information Integration in Digital Ecosystems (AIIDE 2008), Linz, Austria (2008)

Behaviour-Based Web Spambot Detection by Utilising Action Time and Action Frequency

Pedram Hayati, Kevin Chai, Vidyasagar Potdar, Alex Talevski

Anti-Spam Research Lab (ASRL)
Digital Ecosystem and Business Intelligence Institute
Curtin University, Perth, Western Australia
<pedram.hayati, kevin.chai>@postgrad.curtin.edu.au
<v.potdar, a.talevski>@curtin.edu.au

Abstract. Web spam is an escalating problem that wastes valuable resources, misleads people and can manipulate search engines in achieving undeserved search rankings to promote spam content. Spammers have extensively used Web robots to distribute spam content within Web 2.0 platforms. We referred to these web robots as spambots that are capable of performing human tasks such as registering user accounts as well as browsing and posting content. Conventional content-based and link-based techniques are not effective in detecting and preventing web spambots as their focus is on spam content identification rather than spambot detection. We extend our previous research by proposing two action-based features sets known as *action time* and *action frequency* for spambot detection. We evaluate our new framework against a real dataset containing spambots and human users and achieve an average classification accuracy of 94.70%.

Keywords: Web spambot detection, Web 2.0 spam, spam 2.0, user behaviour

1 Introduction

Web spam is a growing problem that wastes resources, misleads people and can trick search engines algorithms to gain unfair search result rankings [1]. As a result, spam can decrease the quality and reliability of the content in the World Wide Web (WWW). As new web technologies emerge, new spamming techniques have also emerged to misuse these technologies [2]. For instance, collaborative Web 2.0 websites have been targeted by *Spam 2.0* techniques. Examples of Spam 2.0 techniques would include creating fake and attractive user profiles in social networking websites, posting promotional content in forums and uploading advertisement comments within blogs.

While similar to traditional spam, Spam 2.0 poses some additional problems. Spam content can be added to legitimate websites and therefore influence the quality of content within the website. A website that contains spam content can lose its

popularity among visitors as well as being blacklisted for hosting unsolicited content if the website providers are unable to effectively manage spam.

A Spam 2.0 technique that has been used extensively is *Web Spambots* or *Spam Web Robots*(which we refer to as *spambots*). Web robots are automated agents that can perform a variety of tasks such as link checking, page indexing and performing vulnerability assessment of targets [3]. However spambots are specifically designed and employed to perform malicious tasks i.e. to spread spam content in Web 2.0 platforms [4]. They are able to perform human-users tasks on the web such as registering user accounts, searching/submitting content and to navigate through websites. In order to counter the Spam 2.0 problem from its source, we focus our research efforts on spambot detection.

Many countermeasures have been used to prevent general web robots from the website [5-7]. However, such solutions are not sophisticated enough to deal with evolving spambots and existing literature lacks specific work dedicated to spambot detections within Web 2.0 platforms.

The study performed in this paper continues from our previous work on spambot detection [4] and presents a new method to detect spambot based on web usage navigation behaviour. The main focuses and contributions of this paper are to:

- Propose a behaviour based detection framework to detect spambots on the Web.
- Present two new feature sets that formulate spambot web usage behaviour.
- Evaluate the performance of our proposed framework with real data.

We extract feature sets from web usage data to formulate web usage behaviour of spambots and use Support Vector Machine (SVM) as a classifier to distinguish between human users and spambots. Our result is promising and shows a 94.70% average accuracy in spambot classification.

2 Spambot Detection

As previously discussed, one area of research to counter the Spam 2.0 problem is spambot detection. The main advantage of such an approach is to stop spam at the source so spambots do not continue to waste resources and mislead users. Additionally, spammers have shown that they use variety of techniques to bypass content-based spam filters (e.g. word-salad [8], Naïve Bayas poisoning [9]) hence spambot detection can be effective solution for the current situation.

The aim for spambot detection is to classify spambot user from human users while they are surfing a website. Some practical solutions such as *Completely Automated Public Turing test to tell Computers and Human Apart(CAPTCHA)[5]*, *HashCode[6]*¸ *Noune[10]*, *Form Variation [10]*, *Flood Control [10]* have been proposed to either prevent or slow down spambots activity within a website. Additionally the increasing amounts of Spam 2.0 and recent works prove that such techniques are not effective enough for spambot detection [11].

Behaviour-based spam detection has more capabilities to detect new spamming patterns as well as early detection and adaptation to legitimate and spam behaviour [12]. In this work we propose behaviour-based spambot detection method based on web usage data.

2.1 Problem Definition

We can formulate spambot detection problem in to a binary classification problem similar to the spam classification problem describe in [13]:

$$D = \{u_1, u_2, ..., u_{|U|}\} \tag{1}$$

where,
D is a dataset of users visiting a website
u_i is the i^{th} user

$$C = \{c_h, c_s\} \tag{2}$$

where,
C refers overall set of users
c_h refers to human user class
c_s refers to spambot user class

Then the decision function is

$$\phi(u_i, c_j) : D \times C \rightarrow \{0,1\} \tag{3}$$

$\phi(u_i, c_j)$ is a binary classification function, where

$$\phi(u_i, c_j) = \begin{cases} 1 & u_i \in c_s \\ 0 & otherwise \end{cases} \tag{4}$$

In spambot detection each u_i belongs to one and only one class so, the classification function can be simplified as $\phi(u_i)_{spam} : D \rightarrow \{0,1\}$.

3 Behaviour-Based Spambot Detection

3.1 Solution Overview

Our fundamental assumption is that spambots behave differently to human users within Web 2.0 applications. Hence by evaluating web usage data of spambots and human users, we believe we can identify the spambots. Web usage data can be implicitly gathered while users and spambots surf though websites . However, web usage data by itself is not effective in distinguishing spambot and human users.

Additional features need to be evaluated with web usage data in Web 2.0 applications for effective spambot detection. Therefore, we investigate two new feature sets called *Action Time* and *Action Frequency* in study spambot behaviour. An *Action* can be defined as a user set of requested web objects in order to perform a certain task or purpose. For instance, a user can navigate to the registration page in an online forum, fill in the required fields and press on submit button in order to register a new user account. This procedure can be formulated as "*Registering a user account*" action.

Actions can be a suitable discriminative feature to model user behaviour within forums but can also be extendible to many other Web 2.0 platforms. For instance, the "*Registering a user account*" action is performed in numerous Web 2.0 platforms, as users often need to create an account in order to read and write content.

In this work we make a use of *action time* and *action frequency* to formulate web usage behaviour. *Action time* is amount of time spend on doing a particular action. For instance, in "*Registering a new user account*" action, *action time* is the amount of time user spends navigating to account registration page, completing the web form and submitting the inputted information. Similarly, *action frequency*¸ is the frequency of doing one certain action. Additionally, if a user registers two accounts, their "*Registering a new user account*" *action frequency* is two. Section 3.2 provides a formal explanation of *action time* and *action frequency*.

It is possible to classify spambots from human user once *action time* and *action frequency* are extracted from web usage data and feed into the SVM classifier.

3.2 Framework

Our proposed framework consists of 4 main modules, which include *web usage tracking*, *data preparation*, *feature measurement* and *classification*as shown in Figure 1.

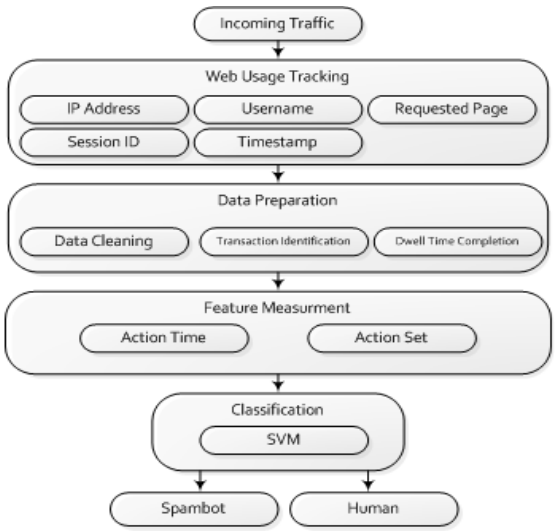


Fig. 1. Behaviour-Based Spambot detection framework

Incoming Traffic

Incoming traffic shows a user entering the website through a web interface such as the homepage of an online forum.

Web Usage Tracking

This module records web usage data including the user's *IP address*, *username*, *requested webpage URL*, *session identity*, and *timestamp*. The username and session ID makes it possible to track each user when he/she visits the system.

Conventionally, web usage navigation tracking is done through web server logs[14]. However, these logs can not specify usernames and sessions of each request. Hence we employ our own web usage tracking system developed in our previous work, *HoneySpam 2.0*[4] in order to collect web usage data.

Data Preparation

This module includes three components, which are *data cleaning*, *transaction identification* and *dwell time completion*.

Data cleaning

This component removes irrelevant web usage data from:

- Researchers who monitor the forum
- Visitors who did not create a user account
- Crawlers and other Web robots that are not spambots

Transaction Identification

This component performs tasks needed to make meaningful clusters of user navigation data[15]. We group web usage data into three levels of abstraction, which include *IP*, *User* and *Session*. The highest level of abstraction is IP and each IP address in web usage data consist multiple users. The middle level is the user level and each user can have multiple browsing sessions. Finally, the lowest level is the session level which contains detailed information of how the user behaved for each website visit.

In our proposed solution we performed spambot detection at the session level for the following reasons:

- The session level can be built and analysed quickly while other levels need more tracking time to get a complete view of user behaviour.
- The session level provides more in-depth information about user behaviour when compared with the other levels of abstraction.

Hence we define a transaction as a set of webpages that a user requests in each browsing session and extract features accordingly.

Dwell time completion

Dwell time is defined as an amount of time user spend on specific webpage in his/her navigation sequence. It can be calculated by looking at each record timestamp. Dwell time is defined as:

$$d_i' = t_{i+1} - t_i \ \ where \ 1 \le i \le |S| \qquad (5)$$

d_i' is dwell time for i[th] requested webpage in session S at time t;

In E.q.5. it is not possible to calculate dwell time for the last visited page in a session. For example, a user navigates to the last page on the website then closes his/her web browser. Hence we consider the average dwell time spent on other webpages in the same session as the dwell time for the last page.

Feature Measurement

This module extracts and measures our proposed feature sets of *action time* and *action frequency* from web usage data.

Definition 1: Set of actions (s_i)

Given a set of webpages $W = \{w_1, w_2, ..., w_{|W|}\}$, A is defined as a set of *Actions*, such that

$$A = \{a_i \mid a_i \subset W\} = \{\{w_l, ..., w_k\}\} \quad 1 \le l, k \le |W| \qquad (4)$$

Respectively s_i is defined as

$$s_i = \{a_j\} \quad 1 \le i \le |T|; \quad 1 \le j \le |A| \qquad (5)$$

s_i refers to a set of actions performed in transaction i and T is total number of transactions.

Definition 2: Action Frequency ($\overrightarrow{aF} = \langle h_1^i, ..., h_{|A|}^i \rangle$)

Action frequency ($\overrightarrow{aF}$) is a vector where h_j^i is the frequency of j th action in s_i. otherwise it is zero.

Definition 3: Action Time ($\overrightarrow{aT} = \langle d_1^i, ..., d_{|A|}^i \rangle$)

We define action time as a vector where

$$d_j^i = \begin{cases} \dfrac{\sum_{k \in a_j} d_k'}{h_j^i} & a_j \in s_i \\ 0 \quad otherwise \end{cases} \qquad (7)$$

d_j^i is a dwell time for action a_j in s_i which is equal to total amount of time spend on each webpage inside a_j. In cases that a_j occurs more than once, we divide d_j^i by the action frequency, h_j^i to calculate the average dwell time.

Classification

We employ Support Vector Machine (SVM) as our machine learning classifier. Support Vector Machine (SVM) is a machine learning algorithm designed to be robust for classification especially binary classification [16]. SVM trains by n data points or features $\{(x_1, y_1), (x_2, y_2), ...(x_n, y_n))$ and each feature comes along with class label (y_i). As mentioned in previous section there are two classes {*human,spambot}* in spambot detection, which we assign numerical value -1 and +1 to each class accordingly. SVM then tries to find an optimum hyperplane to separate two classes and maximising the margin between each class. A decision function on new data point x is define as $\phi(x) = \text{sgn}(w, x + b)$ where *w* is weight vector and *b* is bias term.

3.3 Performance Measurement

We utilised *F-Score* to measure the performance of our classification results [17]. F-Score is defined E.q. 8.

$$F = 2\frac{\text{Re}\,call \times \text{Pr}\,ecision}{\text{Re}\,call + \text{Pr}\,esision} \qquad (8)$$

where

$$\text{Re}\,call = \frac{|Detected\ Spambots|}{|Spambots|}$$

$$\text{Pr}\,ecision = \frac{|Detected\ Spambots|}{|Spambots + Humans|}$$

In the next section we discuss about experimental result of our work

4 Experimental Results

4.1 Data Set

We collected our spambot data from our previous work [4] over a period of a month. We combine this data with human data collected from an online forum with same configuration as spambot host. We removed domain specific information from both datasets. Next we combined these two dataset for experimentation. Table 1 illustrates a summary of our collected data.

Table1. Summary of collected data

Data	Frequency
# of human records	5555
# of spambot records	11039
# of total sessions	4227
# of actions	34

In feature measurement module we come up with 34 individual actions. We extract *action time* and *action frequency* from our dataset and use them separately in our classifier.

4.2 Results

We run 2 experiments on our dataset based on each feature set. We achieved an average accuracy of 94.70%, which ranges from 93.18% for *action time* to 96.23% for *action frequency*. Table 2 and Table 3 summarise the result from each experiment along with ratio of true-positives(TP) (the number of correctly classified spambots) and false-positives (FP) (number of incorrectly classified human users).

Table2. Summary of experimental results on *action time* (aT) feature set

C	TP	FP	Precision	Recall	*F*
c_h	0.976	0.399	0.948	0.976	0.962
c_s	0.601	0.024	0.774	0.601	0.676
Average	0.932	0.355	0.927	0.932	0.928

Table3. Summary of experimental results on *action frequency* (aF) feature set

C	TP	FP	Precision	Recall	*F*
c_h	0.998	0.299	0.961	0.998	0.979
c_s	0.701	0.002	0.975	0.701	0.815
Average	0.962	0.264	0.963	0.962	0.960

It is clear that *action frequency* is a slightly better classification feature to classify spambot from human users. Spambots tend to repeat certain tasks more often when compared with humans that perform a larger variety of tasks rather than focusing on specific tasks. The result of our work shows that *action time* and *action frequency* are good feature for spambot detection and therefore Spam 2.0 prevention.

5 Related Works

There has been extensive research focused on spam management and spam filtering. However, there has been little work dedicated to Spam 2.0 and spambot detection.

In the web robot detection, Tan et al. [3] propose a framework to detect unseen and camouflaged web robots. They use navigation pattern, session length and width as well as the depth of webpage coverage to detect web robots. Park et al. [7] present a malicious web robot detection method based on HTTP headers and mouse movement. However none of these works have studied spambots in Web 2.0 applications.

Yiquen et al.[18] and Yu et al. [19] utilise user web access logs to classify web spam from legitimate webpages. However the focus of their work is different from ours as they rely on user web access log as a trusted source for web spam classification. However, in this work we show that web usage logs can be obtained from both humans and spambots and such as distinction should be made.

In our previous work on HoneySpam 2.0 [4], we propose a web tracking system to track spambot data. The dataset collected in HoneySpam 2.0 is used in this work.

6 Conclusion

To the best of our knowledge, our research from [4] and this paper is the first work focused on spambot detection while conventional research has been focused on spam content detection. In this paper, we extended our previous work in spambot detection in Web 2.0 platforms by evaluating two new feature sets known as action time and action frequency. These feature sets offer a new perspective in examining web usage data collected from both spambots and human users. Our proposed framework was validated against an online forum and achieved an average accuracy of 94.70% and evaluated the performance of our framework using F-score. Future work will be focused on evaluating more feature set or combination of feature set, decrease ratio of false-positives as well as extending our work on other web 2.0 platforms to classify spambots from human users.

References

[1] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*, Chiba, Japan, 2005.

[2] P. Hayati and V. Potdar, "Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods " in *7th IEEE International Conference on Industrial Informatics* Cardiff, Wales, 2009.

[3] P.-N. Tan and V. Kumar, "Discovery of Web Robot Sessions Based on their Navigational Patterns," *Data Mining and Knowledge Discovery*, vol. 6, pp. 9-35, 2002.

[4] P. Hayati, K. Chai, V. Potdar, and A. Talevski, "HoneySpam 2.0: Profiling Web Spambot Behaviour," in *12th International Conference on Principles of Practise in Multi-Agent Systems*, Nagoya, Japan, 2009, pp. 335-344.

[5] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in *Advances in Cryptology — EUROCRYPT 2003*, 2003, pp. 646-646.
[6] D. Mertz, "Charming Python: Beat spam using hashcash," [Accessed: 3 Aug 09] http://www.ibm.com/developerworks/linux/library/l-hashcash.html, 2004.
[7] K. Park, V. S. Pai, K.-W. Lee, and S. Calo, "Securing Web Service by Automatic Robot Detection," *USENIX 2006 Annual Technical Conference Refereed Paper*, 2006.
[8] T. Uemura, D. Ikeda, and H. Arimura, "Unsupervised Spam Detection by Document Complexity Estimation," in *Discovery Science*, 2008, pp. 319-331.
[9] S. Sarafijanovic and J.-Y. Le Boudec, "Artificial Immune System for Collaborative Spam Filtering," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, 2008, pp. 39-51.
[10] U. Ogbuji, "Real Web 2.0: Battling Web spam," [Accessed: 3 Aug 09] http://www.ibm.com/developerworks/web/library/wa-realweb10/, 2008.
[11] H. Abram, W. G. Michael, and C. H. Richard, "Reverse Engineering CAPTCHAs," in *Proceedings of the 2008 15th Working Conference on Reverse Engineering - Volume 00*: IEEE Computer Society, 2008.
[12] J. S. Salvatore, H. Shlomo, H. Chia-Wei, L. Wei-Jen, N. Olivier, and W. Ke, "Behavior-based modeling and its application to Email analysis," *ACM Trans. Internet Technol.*, vol. 6, pp. 187-221, 2006.
[13] Z. Le, Z. Jingbo, and Y. Tianshun, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, pp. 243-269, 2004.
[14] R. Cooley, B. Mobasher, and J. Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns," *Knowledge and Information Systems*, vol. 1, pp. 5-32, 1999.
[15] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, 1997, pp. 558-567.
[16] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," S. a. a. http://www.csie.ntu.edu.tw/~cjlin/libsvm, Ed., 2001.
[17] C. J. V. Rijsbergen, *Information retrieval*: Butterworths, 1979.
[18] L. Yiqun, C. Rongwei, Z. Min, M. Shaoping, and R. Liyun, "Identifying web spam with user behavior analysis," in *Proceedings of the 4th international workshop on Adversarial information retrieval on the web* Beijing, China: ACM, 2008.
[19] H. Yu, Y. Liu, M. Zhang, L. Ru, and S. Ma, "Web Spam Identification with User Browsing Graph," in *Information Retrieval Technology*, 2009, pp. 38-49.

HoneySpam 2.0: Profiling Web Spambot Behaviour

Pedram Hayati*, Kevin Chai, Vidyasagar Potdar, and Alex Talevski

Digital Ecosystem and Business Intelligence Institute, Curtin University,
Perth, Western Australia
{Pedram.Hayati,Kevin.Chai}@postgrad.curtin.edu.au
{v.potdar,a.talevski}@curtin.edu.au

Abstract. Internet bots have been widely used for various beneficial and malicious activities on the web. In this paper we provide new insights into a new kind of bot termed as *web spambot* which is primarily used for spreading spam content on the web. To gain insights into web spambots, we developed a tool (HoneySpam 2.0) to track their behaviour. This paper presents two main contributions, firstly it describes the design of HoneySpam 2.0 and secondly we outline the experimental results that characterise web spambot behaviour. By profiling web spambots, we provide the foundation for identifying such bots and preventing and filtering web spam content.

Keywords: honeyspam, web spambot, web robot detection, spam detection, web usage mining.

1 Introduction

Web content which provides false or unsolicited web information is defined as Web Spam [1]. Spam content is prolific on the web due to the development and widespread adoption of Web 2.0 as spammers no longer need to purchase, host and promote their own domains. Spammers can now use legitimate websites to host spam content such as fake eye-catching profiles in social networking websites, fake promotional reviews, responses to threads in online forums with unsolicited content and manipulated wiki pages etc. This spamming technique which is different from previous Web spamming techniques is referred to as *Web 2.0 Spam* or *Spam 2.0* [2]. Figure 1 illustrates Spam 2.0 and its relation to other spamming campaigns.

Little is known about the amount of Spam 2.0 on the Internet. At the time of writing this paper, *Live Spam Zeitgeist* has shown over a 50% increase in the amount of spam messages (e.g. spam comments in blogging tools) since 2008 [3]. This report showed a dramatic increase in the amount of successful Spam 2.0 attacks and the inadequacy of existing countermeasure tools [2].

In this paper, we conduct an effective study on Spam 2.0 and the behaviour of web spambots. Web spambots are a new kind of Internet robot which is primarily used for spreading spam content on the web. To gain insights into this kind of bot we have developed a tool (HoneySpam 2.0) to track their behaviour.

* Pedram Hayati is PhD student at Curtin University.

J.-J. Yang et al. (Eds.): PRIMA 2009, LNAI 5925, pp. 335–344, 2009.

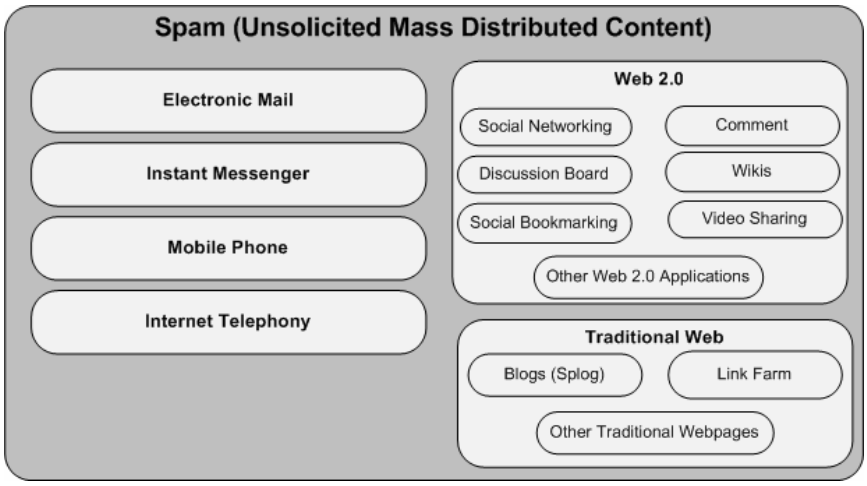


Fig. 1. Spam 2.0 among other spamming campaigns

HoneySpam 2.0 draws on the idea of honeypots (a technique for tracking cyber attackers). It implicitly tracks web usage data which includes detailed monitoring of click streams, pages navigation, forms, mouse movements, keyboard actions and page scrolling. Our results illustrate that HoneySpam 2.0 is effective in profiling web spambots. HoneySpam 2.0 is a tracking solution that can be integrated in any web application ranging from blogging tools to social networking applications. To the best of our knowledge, there is no existing literature that studies web spambots from a web usage perspective. We believe that this work is the first step in finding the attributes of web spambots and is a roadmap for future research.

This paper is organised as follows; in Section 2, we provide taxonomy on Spam 2.0 and web spambots, Section 3 explains the HoneySpam 2.0 architecture. Next, Section 4 discusses our experimental results. Section 5 explains previous work. This is followed by conclusions and future works in Section 6.

2 Taxonomy of Spam 2.0 and Web Spambots

Along with its web applications, Web 2.0 provides a collaboration platform for flexible web content management. Spam 2.0 has existed since web 2.0 concepts have appeared online. Spammers misuse/use this functionality to distribute spam content. The act of spamming can be done through two channels: manual and automated. The former refers to using human operations such as hiring cheap labour to manually distribute spam content [4] (e.g. A sample job advertisement for spam content publishing can be found in [5]). The latter is achieved by exploiting software or scripts which can be in the form of client-side software or web robots. Web or Internet robots are programming scripts or automated agents that are designed for specific tasks such as crawling webpages and parsing URLs without human interaction [6].

Unfortunately, web robots can be exploited for malicious activities such as checking web servers for vulnerabilities, harvesting email addresses from webpages

as well as performing Denial-of-Service attacks (DoS) [7]. Recently, some web robots have been used for distributing spam content on web applications through posting promotional comments, placing ads in online forums, submitting links through trackbacks etc. [2]. We call this type of web robots as *web spambots*. Web spambots can be designed to be *application specific* which target specific web applications like Wordpress blogging tools, phpBB, Mediawiki or *website specific* infecting websites like Amazon, MySpace, Facebook etc.

It should be noted that web spambots are different from spambots. According to [7] spambots are Internet robots that crawl webpages to harvest email addresses in order to send spam emails. However web spambots act completely differently from spambots. Web spambots can crawl the web or discover new victims via search engines, register new user accounts and submit spam content in Web 2.0 applications. Their main task is to distribute spam content on to the web. They can reside on different machines (e.g. infected user computers, free or commercial web hosting sites). Web spambots can be programmed once and used many times. The motivations behind such spamming activities are [2]:

- Deceiving search engines to rank spam and junk content higher. For instance, the more spam content added to different websites linking back to the spammer's websites, will increase their website's ranking.
- Misleading users to view unsolicited contents such as webpages, link-farms, ad-hosted website, phishing websites, etc.

By employing automated web spambots, spammers are able to get more traffic on their campaigns. This highlights the importance of web spambot detection as a possible way to eliminate spam 2.0. In the next section we discuss current techniques to discover web spambots along with our proposed solution.

2.1 Web Spambot Countermeasure Techniques

One of the most popular countermeasures to differentiate web robots (include web spambots) from humans is *Completely Automated Public Turing test to tell Computers and Human Apart* (CAPTCHA) [8]. It is a type of challenge-response test usually in the form of a distorted image of numbers and letters, which humans have to infer and type in to a web form. However, CAPTCHA is inconvenient for users as it wastes their time, causes distraction, is unpleasant and at times scary. A number of recent works have reported approaches to defeat CAPTCHAs automatically by using computer programs [9-11]. CAPTCHA's drawbacks includes the following:

1. Decrease user convenience and increase complexity of human computer interaction.
2. As programs become better at deciphering CAPTCHA, the image may become difficult for humans to decipher.
3. As computers get more powerful, they will be able to decipher CAPTCHA better than humans.

Therefore, CAPTCHA is only a short term solution that is about to expire. Web spambots armed with anti-CAPTCHA tools are able to bypass this restriction and spread spam content easily while the futile user inconvenience remains.

Other countermeasures for combating automated request are by using *Hashcash*, *Nonce* and *Form variation* [12, 13]. The idea behind *Hashcash* is that the sender has to calculate a stamp for the submitted content [13]. The calculation of the stamp is difficult and time-consuming for sender but comparatively cheap and fast for receiver to verify. Although *Hashcash* cannot stop automated spamming, it can slow down the spamming process. On the other hand, using *nonce* and *Form variation* techniques can only guarantee that users use their submission forms rather than posting directly.

However, we believe that one effective way to detect web spambot is by looking at web spambot behaviour through its page navigations, click-streams, mouse interactions, keyboard actions and form filling. We put forward the notion that their web usage behaviour is intrinsically different from humans. The focus of this work is on profiling web spambot behaviour. We proposed HoneySpam 2.0 to detect, track and monitor web spambots. In following section we explain our proposed method.

3 HoneySpam 2.0 Architecture

HoneySpam 2.0 is a stand-alone tool that is designed to be integrated with any web application with the aim to track and analyse web spambot behaviour. HoneySpam 2.0 is based upon the idea of a honeypot where a vulnerable server is deployed to attract cyber attackers in order to study their behaviour. HoneySpam 2.0 is designed to study the web spambots which are considered to be attackers in the spamming scenario.

HoneySpam 2.0 consists of two main components: Navigation Tracking Component (NTC) & Form Tracking Component (FTC). Figure 2 illustrates the detailed Model-View-Control (MVC) architecture of HoneySpam 2.0 inside a web application.

3.1 Navigation Tracking Component

This component is designed to track page navigation patterns and header information of incoming traffic. The following information is captured by this component: the time of request, the IP address, the browser identity and the referrer URL.

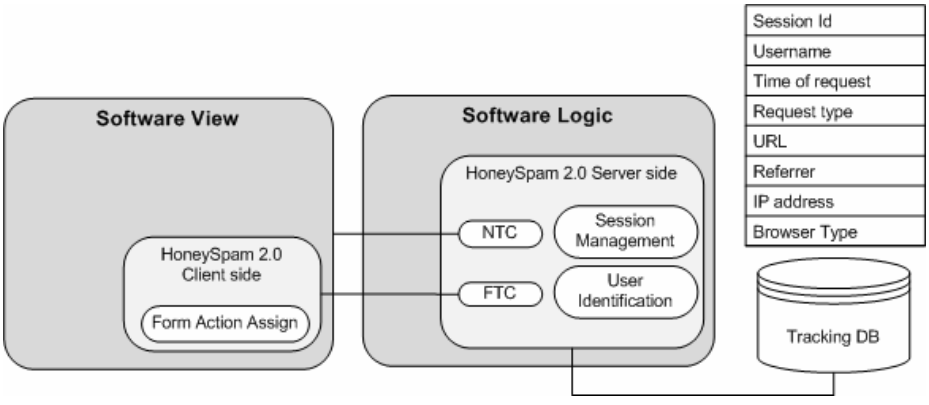


Fig. 2. HoneySpam 2.0 Architecture

NTC is also designed to automatically differentiate individual visit sessions. Session information is captured to identify user's web usage behaviour during each visit. NTC stores the captured data along with session identity (session id) into the tracking database, which is used during an analysis and profiling stage.

3.2 Form Tracking Component

Other than the NTC, we felt the need to develop additional functionality that could provide us insights into how web spambots use web forms. It is a common belief that web spambots do not interact with web forms, however, our study aims to test this here. We developed a form tracking component to specifically study form input behaviour. This component captures and stores the following form related actions such as:

- ⇒ Mouse clicks and movements
- ⇒ Keyboard actions
- ⇒ Form field focus and un-focus
- ⇒ Form load and submission
- ⇒ Page navigation

For each of the above mentioned actions we also captured header information e.g. IP address, browser identity, referrer link session id etc. We decided to store header information since we wanted to ensure that form action events are originating from the same session with the same header information.

Additionally, both components check whether requests are initiated from registered users or visitors. If it was initiated from a registered user then it stores the username along with other web usage information for that request. This was done to differentiate registered web users usage data and visitor usage data, which can be used for further analysis. Visitors are termed as guests in our database. Also, most web applications require the registration of a valid username before further interaction with the system.

3.3 Deploying HoneySpam 2.0

Deploying HoneySpam 2.0 involves implementing both client side and server side code. The client side code, assigns FTC actions to web forms so once a user triggers an action it would be captured, transmitted to FTC server side code and would be recorded in the database. On the server side, NTC tracks the users navigation behaviour based on incoming traffic requests. In the next section, we explain our experimental setting by outlining where we integrated HoneySpam 2.0 and discuss our results and key observations.

4 Experimental Results

The HoneySpam 2.0 tool was employed onto 1 commercial and 5 free web hosting servers. The web applications provided on these servers include online discussion forums and social bookmarking. We did not add any initial content to any of these applications and placed a message (in the form of an image) to notify human users not

to use these applications. Additionally, as described in Section 4.4 our FTC component did not capture any form activity records which ensures that there were no human activities inside our hosted campaigns.All of the content added to our web applications is added by web spambots. The data presented in this paper has been collected for a period of 60 days[1] and is summarised in Table 1.

Table 1. Summary of Data for Honeyspam 2.0 showing the total number of bots, total number of posts, average posts per day etc

Summary of Data for Honey Spam	#
No. of tracking records	11481
No. of web spambots' tracking records	6794
Total No. of registered bots	643
Total No. of unique used IP addresses	314
Total No. of posts	617
Average Posts per day	8.27
Average Registered web spambots per day	7.91
Average online web spambot per day	2.72

In terms of content, 60% of spam content was adult-related, 13% consisted of an assorted category of content (e.g. free software, keyword spam) and 9% of content was related to drug (pharmaceutical) and movies (Fig 3a). Demographically, 51% of observed web spambots originated from the United Kingdom followed by the United States of America at 22% (Fig 3b). However, it is quite possible for web spambots to be used by spammers on hijacked machines so these figures do not reveal the true origin of the spammers.

Web browsers commonly used by the web spambots were Internet Explorer and Opera (Fig 3c). Surprisingly, two relatively unknown browsers which were OffByOne and BrowseX were also used. Additionally, Firefox was only used by 2 web spambots and Safari was not used at all. It should be mentioned that this header information can be modified and is not a true indication of browser usage or the use of an actual browser at all. A number of specific observations were discovered from examining the behaviour of web spambots and are now discussed.

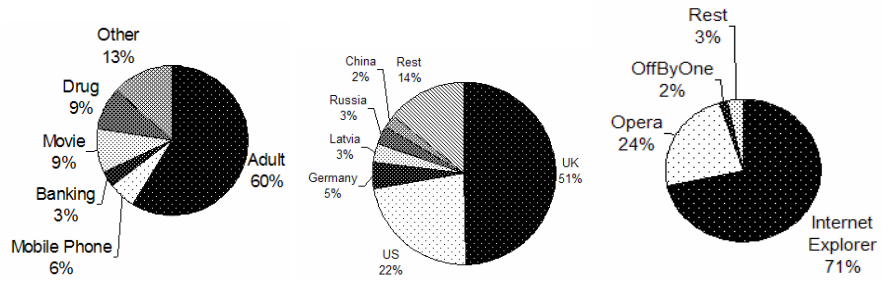


Fig. 3. Percentage of top content categories, countries and internet browsers

[1] The HoneySpam 2.0 tool has tracked spambot behaviour from the 25th of May 2009 to the 9th of August 2009. However, only 60 days of data is collected due to server downtime periods caused by maintenance and issues.

4.1 Use of Search Engines to Find Target Websites

Of the 6 monitored web hosts, the commercial server received the most attention from web spambots. This highlights the possibility that web spambots are using search engine intelligence (e.g. Google PageRank scores) to identify and focus on higher ranked websites that potentially yield more visitor traffic.

The initial 15 days of hosting and tracking did not yield much activity but a number of web spambots began registering and posting content after this period (Fig. 4a). The reason behind this could be related to search engine indexing, which also suggests that some web spambots find their targets through search engine results.

4.2 Create Numerous User Accounts

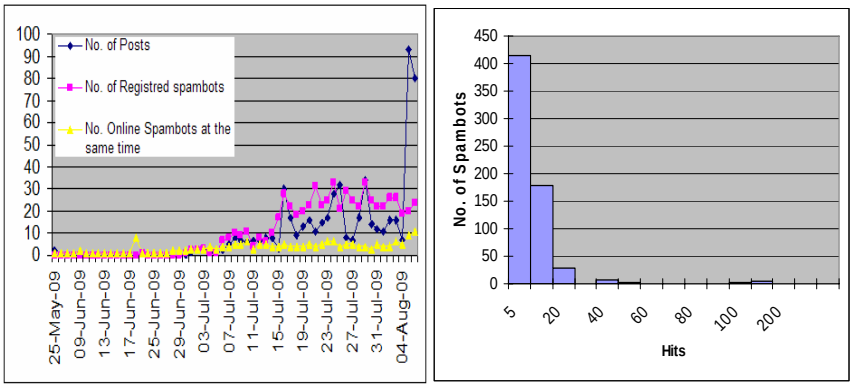
It was identified from our experiment that web spambots would create a number of new accounts to post their content rather than using their existing accounts. This is shown in Table 1, where the total number of web spambots accounts is 643 but only 314 unique IP addresses are used between these accounts. On average 7.91 new accounts were created per day within our dataset.

It is possible that web spambots adopt this behaviour for additional robustness in delaying account banning (e.g. one user account is associated with numerous spam content items is easier to detect and manage) and/or to handle the event in which their previous accounts are banned.

4.3 Low Website Webpage Hits and Revisit Rates

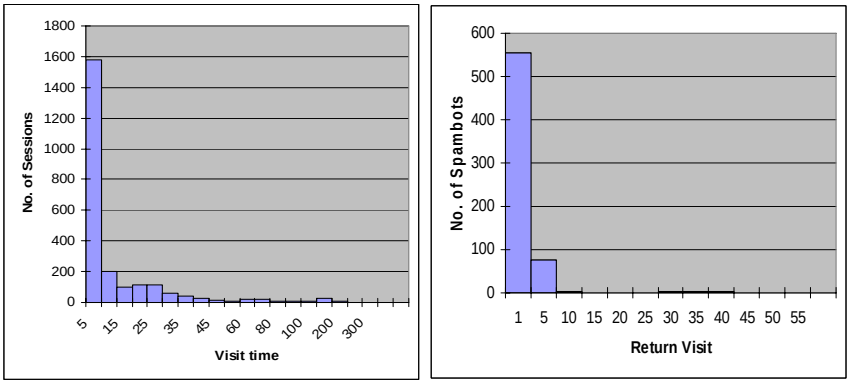
The observed web spambots adopt similar surfing patterns to one another and commonly visit a number of specific web pages. As illustrated in Fig. 4b most web spambots perform less than 20 page hits in total during their account lifetime.

Fig. 5b supports the claim that web spambots do not revisit many times as 554 web spambots only revisit the websites once and 75 bots revisit less than 5 times. It is likely that these bots revert to creating new user accounts to continue their spamming activities as discussed in section 4.2.



(a) (b)

Fig. 4. Number of posts, registered, and online web spambots, frequency of Spambots visits on each Webpages



(a) (b)

Fig. 5. Frequency of Web Spambots' spending time (sec) on each session, frequency of revisiting by Web Spambots (1 means they just visit once)

4.4 Distribute Spam Content in a Short Period of Time

The amount of time most web spambots spend on each visit (session) is less than 5 seconds. This indicates that web spambots are distributing spam content in a short amount of time possibly to move on to other websites to distribute more spam content (Fig. 5a). Additionally their online activitiy time was an average of 2.71 second.

4.5 No Web Form Interaction

While not shown in the figures, the form-tracking component of the HoneySpam 2.0 tool was unable to track form usage behaviour of web spambots. These bots adopt a conventional method of directly posting through their own forms / interface to submit content. We suspect they have not been designed to be intelligent enough to mimic the behaviour of a human in submitting a web form because of a lack of a requirement to do so. Therefore, we believe that simple form usage heuristics can lead to web spambot detection and filtering.

4.6 Generated Usernames

It is apparent that web spambots are generating usernames in order to create numerous accounts. Some examples of these usernames include *Oxidylozaxygixoc*, *Ufigowigelyniwyl* and *Ynutaznazabalekil*. Pattern analysis of these randomly generated usernames could serve as a means of identifying and blocking web spambots, which we will investigate in the future.

5 Related Work

The study of web robots has been thoroughly investigated by the research community since bots consume network bandwidth and make web usage mining difficult. Conventional methods to detect web robots are based on identifying IP addresses and

user agents [14]. However, unknown and camouflaged web robots can not be detected by these techniques.

In the web usage mining, Tan and Kumar [6] proposed a model to discover web robots. They use navigational pattern on click-stream data approach to detect web robots. They show that the features such as the request method, the length of the session, the depth and width of webpage coverage of web robots are different from humans. They did not explicitly focus on web spambots as the aim of their work was to detect web robot such as search engine crawlers, offline browsers, link checkers and email collectors. In our proposed work, we concentrate on profiling web spambots which are different to other types of web robots not covered by Tan and Kumar.

By looking at the existence of HTTP requests for CSS and JavaScript files along with mouse movement activities on the Web pages, Park et al. (2006) proposed a technique to detect malicious web robots [7]. The focus of their work was on camouflaged web robots that are normally used for security attacks, harvesting email addresses etc. Similar to previous work, their focus was not on identifying web spambots.

In the area of spam detection, Webb et al. (2008) proposed a social honeypot for detection of spammers in social networking websites [15]. They hosted 51 social honeypots in one of the famous social networking websites. The result of their work shows that temporal and geographic patterns of social spam have unique characteristics.

Andreolini et al. (2005) proposed a method to slow down email harvesting process and poison spamming email databases [16]. Additionally they proposed the creation of fake open proxies to increase the probability of email spam detection.

6 Conclusion and Future Work

In this paper we detailed the design and implementation of HoneySpam 2.0 for detecting, monitoring and analysing web spambots. HoneySpam 2.0 is based upon the idea of honeypots to monitor and characterize web spambots which are a type of web robot that are programmed to surf through the web and distribute the spam content. We integrated Honeyspam 2.0 in popular open source web applications for period of 60 days to monitor web spambot behaviour.

Through our experiments, we discovered that web spambots use search engines to find target websites, create numerous user accounts, distribute spam content in a short amount of time, do not revisit the website, do not interact with forms on the website, and register with randomly generated usernames. By profiling web spambots, we provide the foundation for identifying such bots and filter and removing their spam content.

While this paper focussed on analysing web spambots as a way to combat spam in the future we will focus on adopting clustering algorithms to detect web spambots on the fly. Additionally, artificial neural networks and analytical techniques can be used to gain a better understanding of the data collected from web spambots.

References

[1] Gyongyi, Z., Garcia-Molina, H.: Web spam taxonomy. In: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web, Chiba, Japan (2005)

[2] Hayati, P., Potdar, V.: Toward Spam 2.0: An Evaluation of Web 2.0 Anti-Spam Methods. In: 7th IEEE International Conference on Industrial Informatics Cardiff, Wales (2009)

[3] Zeitgeist, L.S.: Comment Spam. In: Akismet, ed. (2009), http://akismet.com/stats/
[4] Cobb, S.: The Economics of Spam. EPrivacyGroup (2003), http://www.eprivacygroup.com
[5] Workathome, Work from home online ad placing work pay per posting (2009), http://www.workathomeforum.in/online-adplacing-homejob.htm, http://www.workathomeforum.in/online-adplacing-homejob.htm
[6] Tan, P.-N., Kumar, V.: Discovery of Web Robot Sessions Based on their Navigational Patterns. Data Mining and Knowledge Discovery 6, 9–35 (2002)
[7] Park, K., Pai, V.S., Lee, K.-W., Calo, S.: Securing Web Service by Automatic Robot Detection. In: USENIX 2006 Annual Technical Conference Refereed Paper (2006)
[8] Chellapilla, K., Simard, P.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In: NIPS (2004)
[9] Abram, H., Michael, W.G., Richard, C.H.: Reverse Engineering CAPTCHAs. In: Proceedings of the 2008 15th Working Conference on Reverse Engineering. IEEE Computer Society, Los Alamitos (2008)
[10] Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: Proceedings. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, p I-134-I-141 (2003)
[11] Baird, H.S., Bentley, J.L.: Implicit CAPTCHAs. In: Proceedings SPIE/IS&T Conference on Document Recognition and Retrieval XII (DR&R2005), San Jose, CA (2005)
[12] Ogbuji, U.: Real Web 2.0: Battling Web spam (2008), http://www.ibm.com/developerworks/web/library/wa-realweb10/
[13] Mertz, D.: Charming Python: Beat spam using hashcash (2004), http://www.ibm.com/developerworks/linux/library/l-hashcash.html
[14] Cooley, R., Mobasher, B., Srivastava, J.: Web mining: information and pattern discovery on the World Wide Web. In: Proceedings of Ninth IEEE International Conference on Tools with Artificial Intelligence 1997, pp. 558–567 (1997)
[15] Webb, S., Caverlee, J., Pu, C.: Social Honeypots: Making Friends with a Spammer Near You. In: Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008), Mountain View, CA (2008)
[16] Andreolini, M., Bulgarelli, A., Colajanni, M., Mazzoni, F.: HoneySpam: honeypots fighting spam at the source. In: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop (2005), Cambridge, MA, p. 11 (2005)