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.talevski@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 compared 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 content.
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.

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 sources1. 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

---

1IP Addresses of Search Engine Spiders: http://www.iplists.com/
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 website. Figure 2 is an example of in the robots.txt file. It forbids all web robots from accessing whole website.

User-agent: *
Disallow: /

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 closest 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, \( w_k \), 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

\[
d_j = \| x - w_j \| = \sqrt{ \sum_{k=0}^{nw} (x_k - w_{kj})^2 } \tag{1}
\]

The minimum \( d_j \) is selected as BMU. After finding BMU, SOM updates \( w_k \) for BMU and its neighbours nodes according to E.q 2 so they get closer to input vector.

\[
w_k(t + 1) = w_k(t) + \alpha(t) h_{ci}(t) [ x - w_i(t) ] \tag{2}
\]

where \( t \) is the time-step, \( h_{ci}(t) \) is a Gaussian neighbourhood kernel and decreases with time \( t_o(t) = \alpha(0) (1 - t) \) is linear learning rate, \( T \) is training length (neighbourhood size less than number of nodes in one dimension of map).

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 as 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.

### 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 (\(aS\))**

Given a set of webpages \(W = \{w_1, w_2, \ldots, w_{|W|}\}\), \(A\) is defined as a set of **Actions**, such that

\[
A = \{a_i \mid a_i \subseteq W\} = \{\{w_1, \ldots, w_k\}\} \quad 1 \leq I, k \leq |W| \tag{4}
\]

Respectively \(s_i\) is defined as

\[
s_i = \{a_j\} \quad 1 \leq i \leq |T|; \quad 1 \leq j \leq |A| \tag{5}
\]

\(s_i\) 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, \(a_j\), as a feature. Hence, we represent an action set as a bit vector

\[
\overrightarrow{aS} = \{v_1^i, \ldots, v_{{|A|}}^i\} \tag{6}
\]

where

\[
v_j^i = \begin{cases} 
1 & a_j \in s_i \\
0 & \text{otherwise}
\end{cases} \tag{7}
\]

**Definition 2: Action Frequency (\(aF = \langle h_1^i, \ldots, h_{|A|}^i\rangle\))**
Table 1 Sample Dwell Time Input Vector for Class 1 Transactions.

| a1 | a2 | ... | a|A| |
|----|----|-----|---|---|
| d1 | 5  | 4  | ...| 0 |
| d2 | 0  | 3.5| ...| 0 |
| ...| ...| ...| ...| ...
| d|T| | 0 | 2 | ... | 1 |

Table 2 All Possible Combinations of Features

<table>
<thead>
<tr>
<th>aT</th>
<th>aF</th>
<th>aS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session (S)</td>
<td>(S, aT)</td>
<td>(S, aF)</td>
</tr>
<tr>
<td>User (U)</td>
<td>(U, aT)</td>
<td>(U, aF)</td>
</tr>
<tr>
<td>IP (I)</td>
<td>(I, aT)</td>
<td>(I, aF)</td>
</tr>
</tbody>
</table>

Where S, U and I are set of all session, users and IP addresses.

Action frequency 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**

\[
\overrightarrow{aT} = \left\{ d_1, ..., d_{|A|} \right\}
\]

We define action time as a vector where

\[
d^j_i = \begin{cases} 
\sum_{k \in a_j} \frac{d_k}{h^j_k} & a_j \in s_i \\
0 & \text{otherwise}
\end{cases}
\] (8)

\( d^j_i \) is a dwell time for action \( a_j \) in \( s_i \), which is equal to total amount of time spent 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. Table 1 shows a sample action time input vector.

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.

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, 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.

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
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 5 Major Characteristic of Spambots in the User Level.

<table>
<thead>
<tr>
<th></th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dwell time</td>
<td>N/A</td>
<td>2.3</td>
<td>2.5</td>
</tr>
<tr>
<td>Frequency</td>
<td>1</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Actions</td>
<td>C, CC</td>
<td>AEFFABCB</td>
<td>AEFABCBBBD</td>
</tr>
</tbody>
</table>

Table 6 Major Characteristic of Spambots in the IP Level.

<table>
<thead>
<tr>
<th></th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>8</td>
<td>11</td>
<td>100+</td>
</tr>
<tr>
<td>Actions</td>
<td>AEFFABCB</td>
<td>D...</td>
<td>AEFABC...</td>
</tr>
</tbody>
</table>

(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, 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:

- 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.
- Spambots 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