SpamWall: Heuristic Filter for Web-Spam

Aasish K. Pappu, Animesh K. Trivedi
Indian Institute of Information Technology, Allahabad (U.P.), India
{akpappu_b03,aktrivedi_b03}@iiita.ac.in

Abstract— **Although email spam has been a problem since long time, we have recently witnessed tremendous growth in Web spam, i.e., web pages containing useless contents. These pages interfere with the algorithms that various search engines adopt for indexing and ranking, and should be filtered out. SpamWall analyses the content and structure of web pages based on pre-defined heuristics, and classifies the pages as spam or ham. Its architecture is based on the pipes-and-filters architecture. SpamWall can be used by the search engines to filter out pages that should not be indexed. SpamWall supports manual training using pre-classified corpus. Our implementation of the SpamWall system performed very well. Having trained SpamWall sufficiently, we asked it to crawl the web and classify the pages as it discovers. SpamWall classified about 92% of the pages correctly. In this paper, we describe the detail design of SpamWall.**

Keywords: Web Spam, Information Retrieval, Content analysis, HTML element analysis, spam filtering, Search Engines

I. INTRODUCTION

The Internet is growing rapidly. New websites are coming up every day. There is no one to monitor the content of all pages on the WWW, and therefore, many pages have questionable quality. Assigning trust [8] to information on the web has suddenly become a very important problem. In the last decade, we have seen that search engines have become a sine qua non to Internet users. People need to find information on certain topics, and search engines are working very hard to provide only relevant results. People desire that the first result in the list should be the exact resource that they were looking for. Clearly, a search engine needs to filter its web indices to find nearest matches to search queries, and then present the information in a palatable form to its users [10].

Commercial websites want the search engines to list their pages at the top of relevant searches. Pages on such sites are usually rich in advertisements and propaganda. Search engine developers use these features to increase the page rank. However, owners of some web sites are found to fool search engines by faking up the features that search engines look for. Thus, although the search engine thinks that a page is important to the user, it has no content that the user might be looking for. This is exactly what we mean by “web spam”.

Reports in 2002 have indicated that about six to eight percent of the pages indexed by a search engine were spam, while reports in 2003 and 2004 have specified this number to be 15 to 18 percent. Another study has found that about 9% of search results contain at least one spam link in the top-10 list, and 68% of all queries contain spam links in the top-200 list [3].

Users get annoyed when search engines navigate them onto pages with no useful content. The search engine could have been fooled because the structure and content of the malicious website was appealing to the crawler’s algorithm. To assist the search engine in lowering the ranks of spam pages, we can put a SpamWall filter between the crawler and indexing. SpamWall would get rid of most of the spam pages, thus improving the performance of that search engine, and by satisfying the users.

In this paper, we first discuss previous work in this area that motivated us. We would highlight shortcomings, if any, that the work suffered from, and illustrate how SpamWall overcomes them. We then briefly describe various techniques that web-spammers generally use to fool search engines, and provide information about how SpamWall tackles the same. In later sections, we describe the framework of SpamWall, provide its results, and finally conclude by mentioning the future scope of the system.

II. THE PROBLEM

Identifying pages that are most relevant to the user based on the search string is a critical problem. Search engines crawl and index a huge number of web pages. Google had reported to have indexed more than 8 billion textual documents by the end of 2005. The number of pages is increasing very rapidly, thereby posing problems to the page ranking algorithms. Since page ranks are an important metric, a problem in evaluating them puts forth significant challenges to robustness, flexibility, and usefulness of the results of search engines.

Crawling and indexing is becoming increasingly difficult, and hence, it is important to ensure that only useful pages are actually indexed. Similar to techniques used in clustering search results, we need to analyze the content and structure of web pages to see if they should be indexed.

III. RELATED WORK

Henzinger et al. identified [13] web spam as one of the most important challenges to web search engines. Manasse et al. presented [7] techniques for detecting hyperlinks pointing to spam web pages based on anomalies in statistical data of the web crawls. A Broder et al. investigated [14] the structure of the web graph. Mehran Sahami et al. examined the challenges in web information retrieval [10] and discussed techniques of placing additional invisible keywords on a web page and suggested methods to tackle such spam by analyzing content using Natural Language Processing and Machine Learning Tools.

Zoltn et al. [3] has discussed about common and latest spam techniques, also mentioned few detection techniques like Statistical language for term spamming, analysis of link-count, analysis of pageRank and collusion detection for link spam farms and trust rank for other types of spam.

Minoru et al. [5] proposed a spam detection technique using the text clustering based on vector space model for email spams. Rajesh et al. proposed a new approach for textual classification based on a method for scoring of documents using the suffix tree for email spams. We are extending similar approach that has been mentioned in [9] for content analysis of web pages.

IV. Spam Techniques

In this section, we briefly mention different techniques that web-spammers use, in order to fool search engines into giving high ranks to their pages.

A. *Term Spamming*

Textual content of a page is edited in order to manipulate the link-structure around that document. For example, someone might give a large number of hyperlinks to a certain commercial website on his page.

B. *Weaving*

Spammers are found to duplicate parts of news articles and online encyclopedia entries to insert large number of randomly picked keywords on their page.

C. *Phrase Stitching*

Spammers may combine sentences and phrases from different sources, usually by using RSS feeds, to create a web page containing potential keywords in a search string.

D. *Content Hiding*

Spammers may place hidden content on a page by setting the same color for text foreground and background. Although this content will not be trivially accessible to human visitors, search engines might consider it to be an important resource.

E. *Cloaking*

Spammer tries to detect the user agent accessing their website, and based on whether a crawler is accessing it or a human visitor is trying to obtain the page, it provides different content.

F. *Redirection*

Crawlers usually do not follow HTTP Redirects via JavaScript. So, pages that might be considered to be useful by a search engines would immediately navigate onto useless pages when some user tries to access it using JavaScript enabled browser.

G. *Intra-word Characters*

The use of intra-word characters, which may be non-alphanumeric. For example the word, "Viagra" can be written as "Vi.agr.a", while the word "medications" written as "med-ica.tions".

H. *Non Alphanumeric Character Replacement*

The use of non-alphanumeric characters instead of letters. For example the word, "Viagra" can be written as "$V|@gr@$".

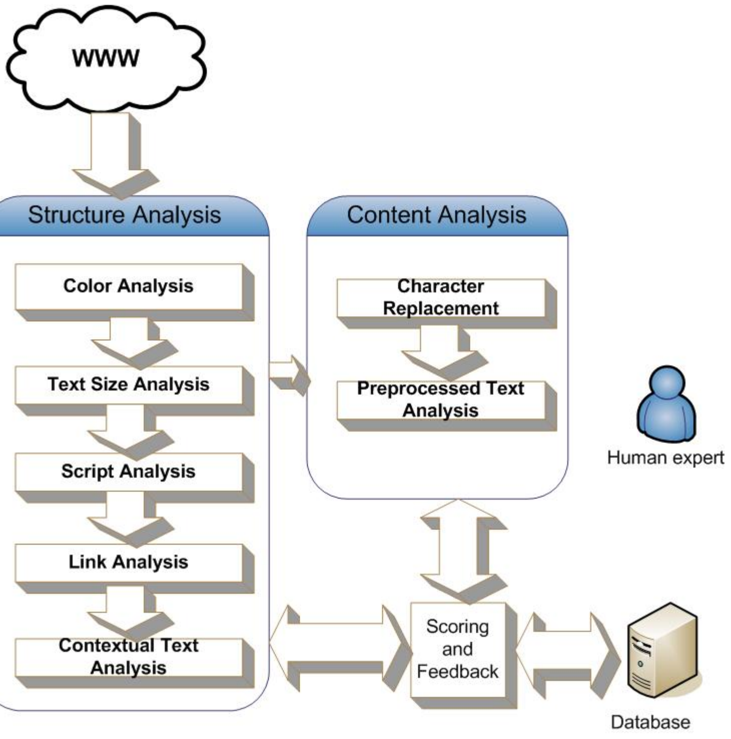

Fig. 1. Architecture of SpamWall

V. Architecture

As stated earlier SpamWall has Pipe and Filter architecture. SpamWall first analyzes the content of a web page, and then analyzes the structure. The figure 1 depicts the architecture of SpamWall. The structural analysis phase is given higher preference, as most of the web-spam apply spam techniques related to HTML tags. The structural analysis phase deals with color, font and size of the text. The content analysis phase constructs a suffix tree of the page and based on that system does further analysis.

A. *Structural Analysis*

Structural analysis of a document uses its HTML DOM tree. We extract this tree using a HTML parser.

1) Color Analysis: Whenever a webpage is found with a text that has color almost similar to that of its background, then it can be assumed to obscure text. This type of spamming is very popular. We calculate the difference between the RGB components of the foreground and background colors of a text. If any of these three differences is below a threshold, we increase the probability that the document is spam. Based on a survey that we conducted in our labs to find out which shades people can distinguish, we set this threshold to 150. If difference between any RGB component of foreground to background crosses threshold (150), then text is visible and system does not categorizes it as obscure text.

2) Text Size Analysis: Keywords in a document are usually emphasized by increasing their font sizes relative to the document text. Search engines give more weightage to keywords found inside the title and heading tags. Spammers write important keywords inside heading tags, but follow it with useless content. To arrest such attempts, we use a natural language processing module to evaluate the relevance of the heading to the text that follows it. This analysis consumes more time, so we carried out in parallel with other analyses.

3) Server-side and Client-side Script Analysis: Crawlers find it difficult to emulate a browser as a user agent, when a complicated JavaScript has been used in the document. SpamWall uses a JavaScript evaluator that could generate an equivalent webpage that would appear if the page was accessed in a browser. Although there is a large number of client side scripting languages, SpamWall currently supports only JavaScript.

A server side script that redirects a browser from one page to other can be easily tackled. For every webpage visited by crawler, SpamWall tries to emulate the redirection by using "curl". If a page has a redirecting script then that would fetch the actual page and avoid the cloaking problem discussed earlier.

There is always a high chance of putting the crawler into an infinite loop of redirections. To avoid such an accident SpamWall maintains a policy for redirection. The policy states that for every redirection we have maintained a log to differentiate between normal link crawling and redirection. For each consecutive redirection of the page the redirection counter is incremented, if the counter crosses the threshold value, that particular crawling thread is blocked, assuming to be the web-page to be a potential spam. The redirection counter threshold is taken as 4 (based on observation).

4) Link Analysis: Linkfarms being the famous spam technique that involving the HTML links and many solutions have been already proposed for the same. We would give more emphasis on the doorway pages. These pages usually contain less content and more links, which would easily annoy the web-page visitor. Although, when a web site uses frames kind of structure, may leave a page with links only (without any content). This issue can be handled easily, by verifying whether that particular page contains any *frame* tag. If it contains any such tag, then the page is permissible to be a ham otherwise it has high chances to be a spam.

5) Contextual text analysis: Sometimes a particular URL may contain a relevant conten that has been indexed by the Search Engine, but due to some reasons, the webpage under that URL might have changed to an irrelevant page. It is also possible that the content of the page depends on the User Agent information in the HTTP request. This technique of poisoning search engine indices can be handled by faking the User Agent during crawling phase, without breaking the Robot Exclusion Principle.

B. Content Analysis

If a document passes all the structural spam-tests, its content is analyzed to measure relevance. Three steps of content analysis are required before a document could be classified as spam or ham.

1) Character Replacement: Textual spam is usually written in non-alphabet characters that obfuscate the spam features, for which our filter is trained for. This issue can be tackled by performing the character replacement procedure based on the pattern matching. The initial procedure of this phase is identifying the non-alphabet character that occurs in the middle of a proper word i.e. a word formed with only alpha-numeric characters. The modification or replacement of that non-alpha character takes place, based on the knowledge base that has been populated with rules during training. A simple example would be concluding whether the @ character has been used in the context of an email address or used as replacement for the character *a*. The rule corresponding to such a replacement is as follows:-

```
if word.Pattern = ^[\\w-\\.]{1,}\\@([\\da-zA
-Z-]{1,}\\.){1,}[\\da-zA-Z-]{2,4}\$

then word.Context = email
else word.Context = alpha
```

The above mentioned regular expression is for identifying an email address. Similarly, every letter has a set of corresponding non-alpha representation that can be written in a text. We have considered two levels of replacement of the text

Instant: An instant replacement would substitute ascii characters like ̀A, Â, A̧, Ă with A.

Verification: Some characters like @ needs a verfication(the example discussed above) whether to be replaced by character *a* or not. There are few more characters that follows same principle, characters like *\$* can be replaced *S*.

Group Some need to be considered as a group of characters that look like a alpha character. For example $|-|$ can be replaced by *H*,

```
|\/|
```

replaced by *M*, $|<$ is substituted by *K* etc.

This kind of representation of alphabets in the form of non-alpha characters has been thoroughly observed by human experts in many spam pages usually sites carrying pornographic or software cracks and serial keys web sites. Although, the population of the knowledge base is currently done manually, an adaptive learning procedure can be applied to train the system with pattern matching of particular character or group of characters with alphabets.

2) Preprocessed Text Analysis: This is the most important step in content analysis. This phase basically consists of suffix tree construction of the document as discussed in [9]. Firstly, the suffix trees have been constructed over the corpus of web-pages. The corpus has been developed by authors themselves in an unbiased manner, due to the non-availability of corpus for web pages unlike email corpora.

The documents are classified into two classes namely, ham and spam. Class tree is developed for each class (ham and spam). We have two suffix trees over which we have to match new webpage which we want to mark spam or ham. For every page first we convert it into a long sequence of string after the initial preprocessing. In our algorithm, stop words are not removed but given minimum weightage. Notation used in subsequent sections are

T is suffix tree
s is string under consideration
m is match (m is prefix of s or may be complete s)
m consists of m_0, m_1, m_2, m_3, m_4, m_5, m_6
(m_i is simply ith alphabet matched)

3) Scoring a Match: A string is said to have match if there exists a path in constructed suffix from root node to any prefix of that string. During the construction of suffix tree we had already distributed probablities to diffrent nodes. So, now for match m we ahve to assign score to that match by giving weightage to each alphabet matched as:-

$m = match(s, T)$ means we have match of some prefix string of s in tree T
$m = m_0m_1m_2m_3m_4$ where each m_i represent a alphabet
$score(m) = v(m|T)\sum_{i=0}^{n}\phi[p(m_i)]$

The function ϕ is used as a significance function. We had used following while constructing tree for ham and spam classes.

$$\phi[p] = \begin{cases} p & \text{linear} \\ p^2 & \text{square} \\ \sqrt{p} & \text{root} \end{cases}$$

These three functions are most popular significance functions in machine learning. In ham matching we had used sub-linear significance function, on the other hand, in spam matching we had used super-linear significance function. We have chosen such a distribution because we want to give more weightage to spam matching rather than ham matching.

$sig(0.5) = 0.25$ for ham while using sublinear function

sig(0.5) = 0.71 for spam while using superlinear function

Although, we have given more weightage to the spam matching, but problem with this approach tends to gives rise to more false positives.

After finding out a match and adding up all significant probability distribution over the tree, now we will consider significance of match m or specifically significance of match of string length L. It might be the case that tree at that depth contains many strings of length L, and hence more the number of strings at that level with same string length less the number of significance of the match. So we had used next level of significance distribution as match significance of string.

$$v(m|T) = \begin{cases} 1 & \text{match unnormalized} \\ \frac{f(m|T)}{\sum_{i\in(m*|T)} f(i)} & \text{match normalized} \\ \frac{f(m|T)}{\sum_{i\in(m'|T)} f(i)} & \text{match length normalized} \end{cases}$$

4) Scoring a Document: Scoring of document is two phase process. First generate different strings that have to match with tree and then second score the mach which we had already described in previous section. Documents should be scored by match of all its suffixes as explained in [9] hence finally score of document is as below:-

$$SCORE(s, T) = \frac{\sum score(s(i), T)}{\tau}$$

Finally tree level normalization we had done as to hide differences between size and structure of suffix tree for different classes.

$$v(m|T) = \begin{cases} 1 & \text{unnormalized} \\ size(T) & \text{match normalized} \\ density(T) & \text{density normalized} \\ logTotalFreq(T) & \text{log total frequency normalized} \\ avFreq(T) & \text{average frequency normalized} \\ avFirstLevelFreq(T) & \text{average first level frequency} \end{cases}$$

After that when score is found we define certain threshold to mark it as ham or spam The document is matched with both of the class trees. The score that has been brought to this phase from structure analysis phase, by the document is added to these new scores. The element phase scoring is explained in the next section.

C. Scoring and Feedback System

The scoring system assigns and maintains the score of a particular document at each level of filtering. The qualification or disqualification of a document at particular level is assigned a specific score for being potential spam or ham. Each level of filtering has been assigned a weightage for assigning the score of a document. Each filter has an associated weight that contributes to the level score of a document.

$$score_{next} = score_{current} + score_{level} \times weight_{level}$$

The *score* of a document is carried forward to the next level, after the document passes through the content analysis. The final score of the document is calculated and the

spam:ham Ratio	spam Recall	spam Precision
1:1	87.00%	92.48%
4:6	89.71%	91.75%
1:5	85.22%	86.07%
1:1	92.15%	92.12%
4:6	92.25%	84.76%
1:5	93.9%	91.89%
1:1	89.09%	92.00%
4:6	94.22%	89.50%
1:5	85.14%	90.10%

Fig. 2. Recall and Precision results against the reciprocal of hsr(ham to spam ratio)

possibility of spam and ham is calculated as below:-

$$hamScore = score_{final} + score_{ham}$$
$$spamScore = score_{final} + score_{spam}$$

The calculation of $score_{ham}$ and $score_{spam}$ has been discussed in the previous section.

$$hsr = hamScore/spamScore$$

if hsr is greater than set threshold value, then the document is considered to be potential ham, weak spam and vice-versa.

VI. Results

As an experimental setup, Web-Sphinx library has been used as a crawler that was developed by Carnegie Mellon University (CMU). Web-Sphinx is supplied with a set of seeded URLs to fetch the documents to the local drive. The testing process is carried out over those documents. Figure 2 shows the spam precision and recall against the hsr. Documents have been parsed with HTML parser to separate the content and DOM tree structure. Previously, SpamWall has been trained with the corpus that has been created by the authors themselves. Few example documents of the corpus are given below

Example 1: document with non-alphabetic characters

```
<html>
<body bgcolor=``lightgray''>
</table>
<font size=2>
<spam content>
h! i am \/|@gr@ p0r!\!
asdf@spam.com
123abc456spammer98098 XXX909ZZZ
&alpha;&alpha;\$&iota;
```

Example 2: Invisible text example

```
<html>
<body bgcolor=``#421D1D''>
<font color=``#6E4A4A''>
This is a standard invisible text
</font>
</body>
</html>
```

Example3: document using server-side script redirection

```
<?php

header('Location:spampage.php');

?>
```

Example 4: document using js redirection

```
<script language=``javascript''><!--
location.replace(``target.html'')
--></script>
```

The sample documents given as input to the SpamWall are converted to the following document after preprocessing (inspite of losing the hamscore to spamscore).

Example 1 after Character-Replacement Phase

```
<html>
<body bgcolor=``lightgray''>
</table>
<font size=2>
<spam content>
hi i am viagra porn
asdf@spam.com
abc spammer.
aast;
```

In the above modified document the text has been re-written by the Character-replacement phase according to the Knowledge Base that has been provided. A keen observation says that in the above modified text, the character @ has not been changed when the context is email whereas it has been modified when the context was alphabet. Number string as that carries no weightage for the spam detection has been eliminated. The elimination has been done based on the heuristics that has been assumed. On regular observation of web pages, it has been assumed that a number of length more than 3(threshold assumed) would not carry special weightage in that particular alpha numeric token.

In the Example 2, the individual differences of R, G, B components of font color and background fails to cross the threshold limit 150(discussed in architecture section).

In Example 3, the redirection is detected by the SpamWall.

VII. Our Contribution

We have proposed tool named SpamWall for detecting and eliminating web spam. Here we have adopted a novel approach for detecting spam pages based on structure and content analysis. Although most of the spam techniques we have dealt are quite common and popular, we maintain that the solutions provided would produce promising results. In the content analysis, no regular preprocessing or stemming of the document is required, as the SpamWall does not neglect the stop words, but only gives lesser weightage. The invisible text technique is handled quite carefully with the help of proposed mathematical formula. Text size has also been considered. A crude JavaScript evaluation is one of the potential solutions given.

VIII. Conclusion and Future Scope

We assert that SpamWall would be a good contribution in the realm of content classification. Adding to the major works that have been done already, SpamWall tries to excel the shortcomings of the previous works. More improvizations need to be done to SpamWall for better classification of web pages. The spam techniques and the solutions discussed in this paper are quite limited. There are many other unexplored spam techniques that would be standing tall in the future. We would like to make the system more robust, adaptive and learning. The current procedure of suffix tree construction of a document is memory intensive. It took nearly 6 hours for processing 1000 average sized documents. Optimization of suffix tree construction has been proposed by Sandeep et al [12] and can be incorporated into SpamWall.

REFERENCES

[1] Pokorny, J.: “Web Searching and Information Retrieval”. Computing in Science Engineering, 2004, Volume 6, pp. 43-48.

[2] Shari Lawrence Pfleeger, Gabrielle Bloom, “Canning spam: Proposed Solutions to Unwanted Email,“ IEEE Security and Privacy, vol. 03, no. 2, pp. 40-47, Mar/Apr, 2005.

[3] Zoltn Gyngyi, Hector Garcia-Molina, “spam: It’s Not Just for Inboxes Anymore”, Computer, vol. 38, no. 10, pp. 28-34, Oct., 2005.

[4] Guido Schryen, “A Formal Approach towards Assessing the Effectiveness of Anti-spam Procedures”, Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 06 January 04 - 07, 2006.

[5] Minoru Sasaki, Hiroyuki Shinnou, “spam Detection Using Text Clustering”, cw, pp. 316-319, 2005 International Conference on Cyberworlds (CW’05), 2005.

[6] K. Albrecht, R. Wattenhofer, “The TROOTH Recommendation System”, pp. 110- 110,Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006).

[7] D. Fetterly, M. Manasse, and M. Najork. “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 (WebDB), Paris, France, 2004.

[8] Mirena S. Chausheva, “Calculating web page trustworthiness by exploring communities on the web”. In Journal of Computing Sciences in Colleges archive Volume 19 ,Issue 5 (May 2004) Pages: 314 – 315.

[9] Rajesh M. Pampapathi, Boris Mirkin, Mark Levene, “A Suffix Tree Approach to Text Categorisation Applied to spam Filtering”, arXiv:cs.AI/0503030 v1 14 Mar 2005

[10] Mehran Sahami, Vibhu Mittal, Shumeet Baluja, and Henry Rowley. “The happy searcher: Challenges in web information retrieval”. In Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI), 2004.

[11] S. Chakrabarti, M. Joshi, V. Tawde, “Enhanced Topic Distillation Using Text, Markup, Tags and Hyperlinks”, SIGIR Conf., 2001.

[12] Sandeep Tata, Richard A. Hankins, Jignesh M. Patel, “Practical Suffix Tree Construction” In Proceedings of the 30th International Conference on Very Large Databases, Publisher: Springer Berlin / Heidelberg, Pages: 281–299, September 2005.

[13] M. Henzinger, R. Motwani, C. Silverstein. “Challenges in Web Search Engines”. SIGIR Forum 36(2), 2002.

[14] A. Broder, M. Najork and J. Wiener. “Efficient URL Caching for World Wide Web Crawling”. In 12th International World Wide Web Conference, May 2003.