On Extendable Software Architecture for Spam Email Filtering

Wanli Ma, Dat Tran, Dharmendra Sharma, *Member, IAENG*

***Abstract*—The research community and the IT industry have invested significant effort in fighting spam emails. There are many different approaches, ranging from white listing, black listing, reputation ranking, postage, legislation, and content scanning etc. Until every ISP obeys the same rules, content scanning based spam email filters still have a significant role to play in fighting spam emails. There are many content scanning based spam email filters available and also in operation. Yet we are still inundated with spam emails everyday. This is not because the filters are not powerful enough, but because the filtering systems are not flexible enough to adapt the new development of spam techniques, such as HTML tagging, image based spam, and keyword obfuscating etc. In this paper, we propose to use dynamic multiple normalizers as the preprocessors for spam filters. The normalizers convert an email to its plain text format, called normalization. With the help of the normalizers, spam filters only need to deal with plain text format, which is what the filters are good at. The flexibility of the proposed architecture does not only make the adoption to the new creations of spammers easier but also makes the integration to the other spam fighting technologies easier.**

***Index Terms*—spam, spam filters, spam detection, normalization.**

I. Introduction

Spam emails are a type of cyber nuisances we have to put up with everyday. The industry and the research community have been investing significant effort in fighting spam emails. There are many different types of proposed technologies. Black listing approach maintains a list of offenders. Any email from the listed offenders is regarded as a spam email and is discarded. White listing approach takes the opposite direction. It maintains an inclusionary list. An email from anybody who is not on the list is regarded as spam and discarded. Postage approach tries to associate some kinds of cost to sending an email in the attempt to make bulk emailing too costly and thus impossible. Legislation, of course, is another approach. By outlawing spam emails, hopefully, prosecution can prevent spam emails being sent out. The most popular approach so far is still content scanning based spam email filtering. The approach resembles how we human beings identify spam emails – learning from past experience. A detection engine is trained with identified spam emails and legitimate emails. The engine learns the features of spam emails from the training. After it is trained, it has the ability to detect spam emails.

The fundamental problem of spam emails is that the Internet is such a distributed system across the whole world, there is no central authority to enforce any anti-spam rules. Until all ISPs and the end users obey the same rules, content scanning based spam email detection and filtering still have a significant role to play in fighting spam emails.

There are many content scanning based spam email filters in operation, ranging from commercial products to open source software. To objectively assess the effectiveness of this type of spam email filters, Cormack and Lynam coordinated a comprehensive and independent evaluation on 44 spam email filters, together with 8 open source filters in 2005 [1]. Their conclusion is that "*The results presented here indicated that content-based spam filter can be quite effective*".

There are also many content scanning based filtering technologies proposed by the research community. Many papers have been published, for example, Naïve Bayes classifier [2], instance-based learning – memory-based approach [3], boosted decision tree [4], Maximum Entropy [5], Support Vector Machines [6], LVQ-based neural network [7], and practical entropy coding theory [8]. The results in the publications also give us very encouraging pictures.

Yet in our daily life, all of us have been continuously suffering from the frustration of spam emails.

On the one hand, spam filters have pretty high recognition rates in the evaluations, and most of the results can be repeated. On the other hand, the real life experience does not match the evaluations. Therefore, logically, we have to ask two questions: *where is the problem*, and *where is the solution?*

Some may claim that the problem is due to the lack of diligent training to the spam filters. We dispute the claim. Spam is a universal problem, and the training results can be easily shared on the Internet. If the training were the problem, we should not see so many spam emails. A parallel observation can be made by the operation of virus scanning software, where

Manuscript received June, 2007. This work was supported by the Divisional grants from the Division of Business, Law and Information Sciences, University of Canberra, Australia, and the University grants from University of Canberra, Australia.

Wanli Ma is with School of Information Sciences and Engineering, University of Canberra, ACT 2601, Australia (phone: +61.2.62012838; fax: +61.2.62015231; e-mail: Wanli.Ma@canberra.edu.au)

Dat Tran and Dharmendra Sharma are with School of Information Sciences and Engineering, University of Canberra, ACT 2601, Australia (e-mail: {Dat.Tran,Dharmendra.Sharma}@canberra.edu.au).

The paper is an extension of the previous paper of the same title published in the Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, 21-23 March, 2007.

virus signature data can be updated reasonably effectively. With virus scanning software properly installed and also properly configured for updating, one can almost be assured of being free of virus attacks.

The real reason for spam problem is actually due to the swift adoption of new techniques by the spammers and the inflexibility of spam filters to adapt the changes.

Almost all of content scanning based spam filters and research proposals are for text based emails, and the evaluations and the research results are also on text based emails. Although the spam email corpus used for some evaluations does contain images, HTML tags, and some attachments, the text part of the emails always has some degree of the indication of its spam nature. In the real world, spammers try everything they can to conceal the text which reveals the spam nature of an email. There are several popular ways of hoodwinking the spam filters. The text part of a spam email may not have a trace of its spam nature at all. Graham-Cumming maintains a comprehensive list of the techniques the spammers use to circumvent spam email filters [9]. Some examples are:

- Using deliberately misspelled words (obfuscating): for example, spell Viagra, a very popular spamming topic, as "v1agra", "V!@gra" or "VlhAGRA". The obfuscations are still humanly readable, but they pose serious challenges to a computer program to catch over 6×10^{20} ways of obfuscations just for the word "Viagra" alone [10].
- Concealing text in images as email attachments: Aradhye et al [15] estimated that 25% of spam emails contain images. C.-T. Wu et al's count is 38% [11]. One of the authors of this paper counted spam emails received from his working email address. Among the 256 spam emails received within 15 days, 91 or 36% of emails were image based. Text based email filters are helpless in dealing with image based spamming. Given the fact that image based spam can successfully circumvent spam filters, the situation can only get worse in the future.
- Obscuring keywords by HTML tags: instead of spelling "Viagra" as it is, individual character is wrapped by HTML tags, such as Viagra.

The combination of these techniques makes it even harder for a spam filter to correctly judge the nature of an incoming email.

In this paper, we propose more flexible software architecture for spam email filtering. We introduce multiple dynamic normalizers as the preprocessors for content scanning based spam email filters. The normalizers are defined by their acceptable input formats and output formats. The system automatically chains a series of normalizers and pipes through an email before it reaches a spam filter. The normalizers on the chain restore, step by step, the underlying message of the email into its plain text format. The spam filter at the end only has to deal with the plain text format, which, from the evaluation and research results, is actually quite effective.

There are a few proposals of combining different spam filtering techniques together. E. Damiani, S, Vimercati, S. Paraboschi, and P. Samarati suggested a distributed P2P-based collaborative spam filtering system [12]. K. Albrecht, N. Burri, and R. Wattenhofer proposed an extendable spam filter system [13]. The system concentrates on a framework for fast spam filtering software development. The "extendable" ability means the implementation of multiple filters, in parallel mode. The system which is close to our proposal is "a unified model of spam filtration" proposed by W. Yerazunis, S. Chhabra, C. Siefkes, F. Assis, and D. Gunopulos [14]. In the model, 6 sequential operations are employed to process an incoming email. They are:

"*1. Initial arbitrary transformation (a.k.a. MIME normalization)*
2. Tokenization
3. Feature extraction
4. Feature weighting
5. Feature weight combination
6. Thresholding, yielding a go/nogo result"

During the first operation, the incoming email is transferred into a text stream. However, there is a significant difference between this proposal and our proposal. The unified model is a fixed and closed system. The operations are predefined by the system. Our proposal is a dynamic and open system. Different operations can be easily introduced into and removed from the system at any time.

Of course, all the approaches discussed previously, such as black listing, white listing, and postage etc., have their merits, and also problems. A holistic approach with the integration of the merits of different approaches is needed to effectively identify spam emails. In this paper, due to the page limit, we will mainly concentrate on content scanning based technologies. However, we should emphasize that our proposal does not exclude the use of the other technologies. Our proposal is a flexible spam filtering software architecture, which is open and dynamic. The flexibility makes the integration of other technologies easier.

The rest of the paper is organized as follows. Section II explains the techniques used to circumvent spam filters. Section III discusses in details the normalizers, and Section IV describes the extendable software architecture for spam email filtering. Section V discusses the issues of assembling normalizers. Section VI presents our experiment results. In Section VII, we conclude the paper with future work and the discussion on the new trend of the images used in spam emails.

II. Circumvent Text Based Spam Email Filters

At the very beginning, emails were in plain text format only [15]. To be able to convey rich presentation styles, they were extended with multimedia abilities [16]. Image based spam emails take the advantage of using the MIME multipart/alternative directive, which is designed to accommodate multiple displays of an email, such as plaintext format and HTML format. The directive suggests that the

enclosed parts are the same in semantics, but with different presentation styles. Only one of them will be chosen to display, and a mailer "*must place the body parts in increasing order of preference, that is with the preferred format last*" [16].

```
From: spammer <faked_email address>
To: recepent_email_address
Content-type: multipart/alternative;

--Boundary_(ID_fkG49yFmM6kAJ0sBSY0dzg)
##### Part 1: plain text format #####
Langdon looked again at the fax an ancient myth confirmed in black and white.

--Boundary_(ID_fkG49yFmM6kAJ0sBSY0dzg)
##### Part 2: HTML format  #####
<textarea style="visibility: hidden;">Stan Planton for being my</textarea>

--Boundary_(ID_fkG49yFmM6kAJ0sBSY0dzg)
##### Part 3: picture format. It has nothing to do with Part 1 or 2 #####
Content-type: image/jpeg; name=image001.jpg
Content-disposition: attachment; filename=image001.jpg

/9j/4AAQSkZJRgABAgAAZABkAAD/7AARRHVja3kAAQAEAAAAHgAA/
+4ADkFkb2JIAGTAAAAAAf/bXFxcXHx4XGhoaGhceHiMlJyUjHi8vMzML
…
--Boundary_(ID_fkG49yFmM6kAJ0sBSY0dzg)--
```

Table I. An imaged based spam email sample

Table I is an example of a spam email. The email has three alternative parts: part one contains plain text paragraphs cut from a book, part two has HTML formatted paragraphs cut from a book as well, and part three is a JPEG formatted picture as in Fig. 1(a). A mailer believes that these three parts are semantically identical and only displays one part, Fig. 1(a) in this case. But in this email, the first two parts have nothing to do with the third part. They are purposely included in the email to deceive text based spam filters. Another similar example can be found in Fig. 1(b).

(a)

(b)

Fig. 1. Images in spam emails

HTML tags can be used to efficiently obscure the keywords of a spam email. The example given in Section I (V<u></u>iagra) can be easily dealt with – removing all HTML tags reveals the underlying keyword. However, it is not so easy to untangle well crafted HTML tag obscurations.

Using an invisible HTML table is one of the examples. The keyword of a spam email, say Viagra, is dissolved into the cells of the table, one character each cell. The visual appearance is still the same, but the HTML text does not have the keyword as a single word any more, Table II(a).

It is not a trivial task to merge the contents of different table cells together, let alone using different alignments of the keyword, e.g., vertical spelling in Table II(b). Using non-uniform table cell structure can further complicate the situation.

	HTML code	display
(a)	<table style="text-align: left" border="0" cellpadding="0" cellspacing="0"> <tbody><tr> <td style="vertical-align: bottom; font-family: arial;"> V </td> <td style="vertical-align: bottom; font-family: arial;">i </td> <td style="vertical-align: bottom; font-family: arial;">a </td> <td style="vertical-align: bottom; font-family: arial;">g </td> <td style="vertical-align: bottom; font-family: arial;">r </td> <td style="vertical-align: bottom; font-family: arial;">a </td> </tr></tbody></table>	Viagra
(b)	<table style="text-align: left" border="0" cellpadding="0" cellspacing="0"> <tbody> <tr style="font-family: arial;"> <td style="text-align: center; vertical-align: top;">V </td></tr> <tr style="font-family: arial;"> <td style="vertical-align: top; text-align: center;">i </td></tr> <tr style="font-family: arial;"> <td style="vertical-align: top; text-align: center;">a </td></tr> <tr style="font-family: arial;"> <td style="vertical-align: top; text-align: center;">g </td></tr> <tr style="font-family: arial;"> <td style="vertical-align: top; text-align: center;">r </td></tr> <tr style="font-family: arial;"> <td style="vertical-align: top; text-align: center;">a </td></tr> </tbody></table>	V i a g r a

Table II. HTML code and visual display

Deliberated misspelling (obfuscating) is also hard to detect. It is not hard to detect the misspells of Viagra as V1agra, \/iagra (V as \ and /) and Vi@gra etc. However, it is not so easy for a program to determine that VlhAGRA is actually Viagra. Given so many ways of obfuscating a keyword, e.g., 6×10^{20} ways of obfuscating Viagra as listed in [10], it is not an easy task for a spam filter to recognize all possible obfuscations, yet makes no mistakes on other words of the email.

The spamming techniques discussed in the previous paragraphs are just some examples. It is almost impossible for a single spam filter to tackle these many different types of spamming techniques, let alone new techniques are constantly being invented. A more flexible approach is needed to easily adapt the ever changing spamming techniques.

III. The Need For Normalizers

In a general sense, spam email filtering can be illustrated in Fig. 2. An incoming email is fed into a filter, or a detection engine. The engine decides, based on its mathematical model, whether the email is spam or not. If it is spam, the email is quarantined; otherwise, the email (called ham) is sent to the

intended recipient as it is. Depending on the system, there might be feedback loops between the end users and the detection engine. The feedback loops help the engine to learn from past mistakes and improve its detection accuracy.

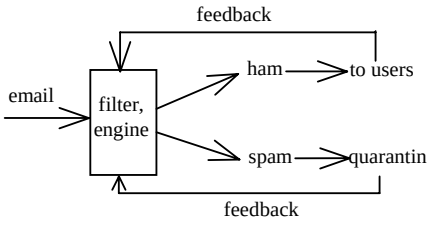

Fig. 2: Spam Email Filtering

Content based spam email filters are effective in dealing with text based spam emails; however, they are helpless in dealing with obscured spam emails, such as imaged based spam emails, using HTML tags to conceal spam email keywords, and spam email keyword obfuscating etc. However, in essence, the purpose of spam emails is to deliver messages. Ultimately, text, as least, humanly readable text, has to be displayed on the screen. As text based spam email filters can be effective in dealing with text, and the ultimate goal of spam emails is to deliver text to the screen for humans to read, effective spam filtering will rely on the building of tools which can convert the obscured formats of spam text into the plain text format. From the detection engine point of view, it is desirable to have all emails in their plain text format. We called the process of converting obscured formats to the plain text format *normalization*, and the tools used to perform the conversion *normalizers*.

Spam emails come with all kinds of tricks and visual appearances. Spammers keep inventing new tricks to hoodwink detection engines. It is difficult for a single engine (or filter) to convert all possible masked spam messages into their plain text format. Therefore, we advocate a multiple and dynamic normalizers approach.

A normalizer is specialized in dealing with a single trick of masking spam messages. For example, we developed a Trigram normalizer and a Markov normalizer to recover obfuscated text into its original format, e.g., from V1agra to Viagra. To deal with imaged based spam, we also developed an OCR normalizer to extract the text from the images.

IV. The Extendable Software Architecture

Using multiple normalizers as preprocessors for a spam detection engine improves its ability to deal with many different ways of masking spam messages. It also provides the flexibility to adapt to new masking tricks. As illustrated in Fig. 3, an incoming email is first duplicated, by a *duplicator* (dup), and then channeled through a number of *normalizers* (N1 to N4). Each of the normalizers tries to recover the underlying text of the email. At the end, a *merger* (mrg) merges the recovered text from all normalizers into a single piece of text and then feeds it into the spam detection engine. The engine then decides the nature of the email, spam or ham, as usual.

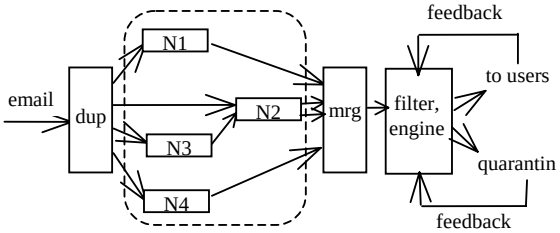

Fig. 3: The Extendable Software Architecture

Every normalizer accepts a particular input format and converts it into another format for output. For example, a HTML normalizer accepts the HTML format, strips off HTML tags, and outputs in the plain text format. An OCR normalizer accepts image formats, jpeg and gif etc., extracts text from images by using OCR technology, and then outputs the extracted text in the plain text format. A Trigram or Markov normalizer accepts noisy text – words with misspellings – and restores the correct spellings.

Not every normalizer can produce the plain text format needed by the detection engine. For example, an OCR normalizer is supposed to extract text from email images and outputs the extracted text in plain text format. However, the text produced by the OCR normalizer very likely contains OCR noise, i.e., misspellings due to misrecognitions. The output text cannot yet be accepted by a content based spam filter. Instead, it should go through another normalizer, e.g., a Trigram or Markov normalizer, to restore the correct spellings from the noisy OCR text.

A normalizer is defined by its acceptable input format and output format, written as (InFormat, OutFormat). An HTML normalizer can be defined as (HTML, Text), an OCR normalizer as (Image, NoisyText), and a Trigram normalizer as (NoisyText, Text). The input format and the output format can be a simple format, e.g., Text, or a combined format, e.g., Image, which can be further defined as jpeg and gif etc. formats.

The duplicator first generates a unique identifying number for an incoming email and then duplicates the email into multiple streams by its all possible formats, e.g., Text, HTML, and Image etc. It then notifies the merger this unique number and the number of the streams. Each stream carries the unique identifying number of the email and is channeled into a normalizer, based on the acceptable input formats of normalizers. The normalizer or a chain of normalizers reproduce the email in the plain text format for the detection engine. Upon receiving a unique number from the duplicator, the merger knows an email is coming in several streams. It starts to count all normalization streams of this email. After collecting all streams, the merger concatenates the pieces of text from these streams into a single piece of text and then passes it to the detection engine.

The order an incoming email going through the normalizers could be rather complicated. However, after clearly defining

the input format and the output format of each normalizer, the system should be able to dynamically find its way through the normalizers by chaining input formats and output formats of the normalizers. There are no predefined pathways among the normalizers. A normalizer is only defined by its input format and output format, and the normalizers are chained dynamically based on the formats. The ultimate goal is to produce the plain text format (Text) for the detection engine. In Fig. 3, the pathways within the round cornered rectangle of dotted line are dynamically assembled based on the content of the incoming email.

All normalizers are equal. A normalizer can be easily introduced into or removed from the system. Within the system, a normalizer is only chained with another normalizer when it is necessary. A normalizer may not be aware of the existence of other normalizers. The only restriction to the system is that all streams out from the duplicator should find their ways to the merger.

Finally, a normalizer may not just trivially convert the input format into the output format. It does the conversion with the emphasis of highlighting spam features. For example, an HTML normalizer, in addition to stripping off HTML tags, also tries to restore words arranged in vertical order, Table II(b).

V. Assembling Normalizers

While the normalizers are dynamically assembled, there are several issues which have to be addressed. For example, how could the merger know it receives all results from all possible normalizers, how many streams of data flow exist, what happens if there is no match to the output format of a normalizer, how to deal with the potential looping among some normalizers, and how a normalizer is introduced into and maintained in the system?

When the mail server receives an email, it first assigns a unique identification number to the email and stores the email in a temporary repository. It then makes a copy of the email, with the unique identification number, and passes this copy to the duplicator (dup) of our filtering system. Upon receiving the result from the filtering system, yes or no of spam, the mail server either forwards the stored email to the end user(s) or simply discards this email.

When the duplicator (dup) receives a copy of email, it does some preliminary processes: separating the header, body, and attachment parts of the email and identifying the types of those parts. Every separated part inherits the unique identification number from the original email. This unique identification number will be used by the merger at the end to assemble the text segments of the same email together. Unpacking attachments requires extra effort. An attachment could be in any format: image (jpeg or jif etc), video, audio, pdf, or Word etc., and even worse being compressed. At this stage, only one level uncompressing is implemented. The duplicator identifies the type of each part of the email and then invokes the relevant normalizer, for example, an OCR normalizer for Image type. After passing all parts to the normalizers, the duplicator notifies the merger how many normalizers it invokes for this email, in other words, how many data streams flowing through the normalizers. Upon receiving the results of all data streams, the merger concatenates the segments of text together and passes the combined text to the spam email detection engine. Although the combined text loses the display rendering information of the email, and it may also have duplications among the text segments, it is good enough for spam detection purpose. If any of the segment triggers the spam email detection engine, the email is spam; otherwise, it is not. The result is thus passed back to the mail server, which will take the action accordingly.

When a normalizer receives its input data, it processes the data and produces the outcomes. The type of output data is predefined by the normalizer. Most normalizers have one type of input and another type of output. However, a normalizer may produce multiple types of outcomes, for example, a normalizer responsible for uncompressing data. Regardless the number of output types, the normalizer further invokes subsequent normalizers the same way as the duplicator does. If a normalizer invokes more than one subsequent normalizer, where the data flow branches, it notifies the merger to increase its counter of data streams of this email. The merger thus will correctly collect the result from this extra branch. At this stage, we do not consider the situation where a normalizer takes two incoming streams, i.e., requiring 2 different types of incoming data. However, there exists the possibility that more than one normalizer matches the output data format. If it happens, the current normalizer multiples the output data stream and invokes all possible normalizers. It also notifies the merger of the increment of the data streams.

Normalizers are introduced into the system over the time base on the need. A normalizer is introduced into the system by a triplet: input type, output type, and the identity name of the normalizer. If an OCR normalizer is needed, it is introduced into the system. However, there might be more than one normalizer of the same input and output types in the system, for example, OCR normalizers of different OCR engines. Multiple normalizers of the same type increase the accuracy of spam detection, because the results of all normalizers are concatenate together. Duplications do not have negative effect. However, multiple normalizers of the same type impose performance penalty to the system. The system maintains a list of normalizers available. The list is consulted for the purpose of invoking normalizers.

As a normalizer is dynamically invoked without a predefined path, it is possible that a normalizer cannot find a suitable subsequent normalizer. This is called a dead end situation. If it does happen, this branch of data will just be silently dropped. The current normalizer notifies the merger to decrease the number of data streams by one.

The most difficult and also dangerous problem of dynamically assembling normalizers on the way is the looping problem. The simpliest and tightest loop involves 2 normalizers. Normalizer A is of the type (X, Y), and normalizer B has the type (Y, X). As soon as any of the data streams

reaches any of these 2 normalizers, it cannot escape anymore. The result will never reach the merger, and therefore, we come to a looping problem or deadlock. As termination problem is undecidable, it is impossible to detect the looping problem. There are mature deadlock prevention algorithms. However, they are costly and may not be suitable for our application. At this stage, we do not enforce any looping check, but we envisage in the future we will introduce the levels of normalizers. The data streams can only go from lower levels to higher levels. More research work is needed in this area.

VI. Experiment Results

So far, we have been concentrating on an imaged based normalizer and 2 obfuscating normalizers [17-21].

We tested a Trigram normalizer and a Markov normalizer by using a test set containing 50 keywords, their 500+ misspellings, and 10,000 normal words. Our experiments showed that the normalizers could restore keywords from their misspellings in the test set with the equal error rate (false rejection error = false acceptance error) of 0.1%. In other word, the recovering rate (from misspellings to correct spellings) reaches 99.9%.

An OCR normalizer is developed by using ripmime and gocr. Both are open source software. We grouped the text embedded in the images into two categories: image text, Fig. 1(a), and *text generated image text* (tegit), Fig. 1(b). In the experiment, the OCR normalizer provided good outcomes in terms of being accepted by the Trigram normalizer or the Markov normalizer. Out of the 33 spam images we recently collected, 23 (70%) are tegits, and 10 (30%) are of image text. Among the 23 files which are produced by the OCR normalizer from tegits, we achieved 100% recovering rate. And, for the 10 files from image text, we achieved 70% recovering rate. Therefore, the overall weighted recovering rate is 91%.

The preliminary experiment is very encouraging, and a large scale evaluation of the effectiveness of the normalizers is on the way. We anticipate better results in the near future.

VII. Conclusion and Future Work

The extendable software architecture for spam email filtering has been presented. The most important feature of the architecture is the flexibility in easily adapting techniques to fight new spamming inventions. Normalizers can be easily introduced into or removed from the system. A normalizer is defined by its input format and output format, and a processing pathway is dynamically decided by chaining the normalizers based on their input and output format specifications. The spam detection engine, with the help of the normalizers, only deals with plain text format, which is pretty effective.

The architecture brings in a few advantages. First, the same detection engine is used to process the text part and other obscured parts of a spam email. Training and updating the detection engine are very expensive operations and in most cases, human intervention is needed. From an operational point of view, keeping one engine saves on cost, and from a system point of view, one engine preserves the integrity of the data and logic. Second, Bayesian filters of text based detection engines are the working horses on the field [22] and also quite effective in detecting text based spam emails [1]. Our proposal takes full benefit of current text based detection engines. Finally, the system is ready to implement on real mail servers. The multiple normalizer pathways only add extra branches to the data flow of an existing spam detection setup. These extra branches of data flow can be easily and seamlessly integrated into the existing spam filters.

The architecture we proposed does not exclude other technologies. To the contrary, it tries to increase the flexibility and thus makes the adaptation to the other technology easier. A normalizer could very well be the one to read the email header information and retrieve other information about this email, say, the reputation rating of the ISP of the sender etc.

We have developed several normalizers. Our next step is to integrate the normalizers into a single system and then conduct a large scale testing on the effectiveness of the normalizers, based on the spam emails from SpamArchive corpus and our own private collections.

Finally, we are aware of that the spammers are trying to sabotage OCR programs by obfuscating images, such as introducing noises to the images and skewing and rotating the text on the images etc. We believe that we have found a solution to recognize these images used for spam purposes. We are conducting more experiment to test the solution. And on the other hand, as the final note, spammers do not spam for fun. They do it for financial return. The obfuscated images, although still humanly readable, won't bring the expected return to the spammers, as human readers are less likely to respond to this type of images. Spammers try to circumvent the spam email filters, but still prefer that the emails look normal so that they won't raise human alarm. In essence, the purpose of spam emails is to deliver messages, and in a nice format. This is the Achilles' heel of spam emails, and it is also the key for us to fight spam emails.

References

[1] Cormack, G. and T. Lynam. *TREC 2005 Spam Track Overview*. in *The Fourteenth TExt Retrieval Conference (TREC 2005)*. 2005. Gaithersburg, MD, USA.

[2] Sahami, M., S. Dumais, et al. *A Bayesian Approach to Filtering Junk E-mail*. in *AAAI-98 Workshop on Learning for Text Categorization*. 1998.

[3] Sakkis, G., I. Androutsopoulos, et al., *A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists.* INFORMATION RETRIEVAL, 2003. **6**(1): p. 49-73.

[4] Carreras, X. and L. Marquez. *Boosting Trees for Anti-Spam Email Filtering*. in *4th International Conference on Recent Advances in Natural Language Processing (RANLP-2001)*. 2001.

[5] ZHANG, L. and T.-s. YAO. *Filtering Junk Mail with A Maximum Entropy Model*. in *20th International Conference on Computer Processing of Oriental Languages (ICCPOL03)*. 2003.

[6] Drucker, H., D. Wu, and V.N. Vapnik, *Support vector machines for spam categorization.* IEEE Transactions on Neural Networks, 1999. **10**(5): p. 1048-1054.

[7] Chuan, Z., L. Xianliang, et al., *A LVQ-based neural network anti-spam email approach.* ACM SIGOPS Operating Systems Review, 2005. **39**(1): p. 34 - 39.

[8] Zhou, Y., M.S. Mulekar, and P. Nerellapalli. *Adaptive Spam Filtering Using Dynamic Feature Space*. in *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*. 2005.

[9] Graham-Cumming, J. *The Spammers' Compendium*. 2006 15 May 2006 [cited 2006 May]; Available from: http://www.jgc.org/tsc/.

[10] Cockeyed. *There are 600,426,974,379,824,381,952 ways to spell Viagra*. 2006 [cited 2006 October 2006]; Available from: http://cockeyed.com/lessons/viagra/viagra.html.

[11] Wu, C.-T., K.-T. Cheng, et al. *Using visual features for anti-spam filtering*. in *IEEE International Conference on Image Processing, 2005 (ICIP 2005)*. 2005.

[12] Damiani, E., S.D.C.d. Vimercati, et al. *P2P-based collaborative spam detection and filtering*. in *4th IEEE International Conference on Peer-to-Peer Computing (P2P'04)*. 2004. Zurich, Switzerland.

[13] Albrecht, K., N. Burri, and R. Wattenhofer. *Spamato - An Extendable Spam Filter System*. in *2nd Conference on Email and Anti-Spam (CEAS'05)*. 2005. Stanford University, Palo Alto, California, USA.

[14] Yerazunis, W.S., S. Chhabra, et al., *A Unified Model of Spam Filtration*. 2005, Mitsubishi Electric Research Laboratories, Inc: 201 Broadway, Cambridge, Massachusetts 02139, USA.

[15] Postel, J.B. *Simple Mail Transfer Protocol*. 1982 [cited 2006 May]; Available from: http://www.ietf.org/rfc/rfc0821.txt.

[16] Freed, N. and N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. 1996 [cited 2006 May]; Available from: http://www.ietf.org/rfc/rfc2046.txt.

[17] Ma, W., D. Tran, et al. *Detecting Spam Email by Extracting Keywords from Image Attachments*. in *Asia-Pacific Workshop On Visual Information Processing (VIP2006)*. 2006. Beijing, China.

[18] Tran, D., W. Ma, and D. Sharma. *Fuzzy Normalization for Spam Email Detection*. in *Proceedings of SCIS & ISIS*. 2006.

[19] Tran, D., W. Ma, and D. Sharma. *A Noise Tolerant Spam Email Detection Engine*. in *the 5th Workshop on the Internet, Telecommunications and Signal Processing (WITSP'06)*. 2006. Hobart, Australia.

[20] Ma, W., D. Tran, and D. Sharma. *Detecting Image Based Spam Email by Using OCR and Trigram Method*. in *International Workshop on Security Engineering and Information Technology on High Performance Network (SIT2006)*. 2006. Cheju Island, Korea.

[21] Tran, D., W. Ma, et al. *A Proposed Statistical Model for Spam Email Detection*. in *Proceedings of the First International Conference on Theories and Applications of Computer Science (ICTAC 2006)*. 2006.

[22] Pelletier, L., J. Almhana, and V. Choulakian. *Adaptive filtering of spam*. in *Second Annual Conference on Communication Networks and Services Research (CNSR'04)*. 2004.