Baeza-Yates and Navarro Approximate String Matching for Spam Filtering

Monther Aldwairi and Yahya Flaifel
Network Engineering and Security Department
Jordan University of Science and Technology
Irbid 22110, Jordan
munzer@just.edu.jo, yahya.flaifel@csb.gov.jo

***Abstract*— Spam has evolved in terms of contents, methods, delivery networks and volume. Reports indicate that up to 90% of the World Wide Web email traffic is spam [1]. The contents are covering a wider range and are deviating from the conventional pharmaceuticals and adult content into more formal marketing campaigns. This illegal advertising is evolving into an underground market for bot masters who rent or sell spam agents. Progressively, spam campaigns engage new methods to ensure efficient mass delivery and dodge conventional spam detectors. They employ very complicated and vast infrastructure of Botnets and Fast Flux Networks to deliver as many emails as possible. The main concerns for spam detection process are detection and misclassification accuracies, and those remain a challenge because of the evolving techniques employed by spammers. In this paper we propose a bit-parallel string matching spam filtering system based on the improved Baeza-Yates and Navarro approximate string matching algorithm. This method has a low computational cost, is easy to implement, and has the potential to catch misspelled keywords. The proposed approach achieves 97.2% overall accuracy with a simple Naïve Bayes classifier.**

***Index Terms*— Spam filtering, approximate string matching, Baeza-Yates and Navarro, Naïve Bayes classifier.**

I. Introduction

Spam is a word commonly used to describe unsolicited email sent in bulk as opposed to regular emails, called ham. Combating spam has become a priority for network administrators and multitude of research and products to peter out spam pop up regularly. However, spammers keep changing their methods to evade detection. They use a very sophisticated infrastructure of botnets and fast flux networks to deliver their content. Bots are compromised home computers centrally managed by the bot herder. Links in spam emails point to bots and once the unsuspecting user clicks a link, the bot works as proxy relaying the request to the actual host. Bots help keep the spammers anonymous but unfortunately are unreliable and frequently go offline. Therefore, fast fluxing comes into play, which is frequently changing the domain name mappings to different IPs for different bots [2]. Before Rustock botnet take down, it accounted for 47% of all spam, while Bagle botnet on the other hand accounted for 20% [3].

Spam advertising is turning into an underground lucrative business according to Stone-Gross, Holz, Stringhini and Vigna [4]. Large email lists cost up to $20,000 and spam as a service costs $500 per million emails sent. Botnets rent for $10,000 a month and are capable of sending 100 million emails per day. Given that spammers send 52 billion spam emails per day, according to Symantec MessageLabs Intelligence annual security report [3], the spam economy is worth hundreds of millions of dollars. However, the financial losses due to lost functionality and spam detection cost are estimated in billions of dollars.

Spam detection techniques are classified into either black listing or content filtering. Black listing was one of the first attempts to deter spammers but unfortunately soon spammers adapted by nimble use of the IP addresses and botnets. Content based filtering works by analyzing the contents of emails as opposed to the source IP. To shirk content-based spam detection, spammers crafted spam emails designed to mislead detectors such as, misspelling keywords and the use of images. Unfortunately, current algorithms incur high computational costs [5]. The other main challenge is false positives leaving useful messages in the spam folder and false negatives allowing spam into user's inboxes [6].

In this paper, we use of Baeza-Yates and Navarro approximate string matching to design a better content based filtering [7]. Moreover we use bit-parallel algorithm suitable for multithreading on today's multicore processors to increase the matching speed [8]. Approximate string matching finds how many times a pattern appears in text with certain variations to that pattern. This comes in handy to detect misspelled keywords.

The rest of this paper is organized as follows. Section II discusses the background and related work. Section III introduces the proposed approach, approximate string matching and Naïve Bayes classification. Finally, Section IV presents the experimental setup, performance metrics and simulation results.

II. Background and Related Work

Based on deployment spam filtering techniques can be classified into origin and destination [9]. The first includes the legislative aspect of the problem as well as new protocols and standards the help limit spam. Agrawal, Kumar and Molle identified spam at the router using Bayesian classifiers of bulk emails [10]. Then they allowed the system administrator to impose limits on delivery rates. Several new standards and protocols were designed to authenticate email senders and help

limit spam such as senderID and trusted open mail standard. SenderID is an email authentication by checking the IP address of the sender against the sending domain [11]. Cook, Hartnett, Manderson and Scanlan proposed an approach based on the observation that spammers, zombie machines and auto-mailers scan the target network before starting their spam campaigns [12]. They proposed the use of system logs analysis and intrusion detection systems to detect malicious intent such as port scans. Once enough evidence is available against an IP or domain, new firewall rules block SMTP connections originating from that host. The main advantage is to stop spam before delivery to the end user by deploying the system at the ISP level.

On the other hand, destination spam filtering takes place after delivery and can be classified based on deployment into server and client based. They are further classified based on filtering technique into traditional, machine learning and hybrid methods [9]. Traditional methods rely on checking emails for spam signatures and keywords. Other traditional methods include black listing, user voting and spam traps. Bag-of-words is a popular approach where two bag-of-words are used, one filled with spam keywords and the other with ham keywords. Kolari, Java, Finin, Oates and Joshi analyzed and detected spam blogs by using bag-of-words and support vector machines (SVM) [13]. They tokenized anchor text and urls into bag-of-urls and bag-of-anchors, and used ratio based features such as anchor text ratio and number of URLs ratio.

Machine-learning techniques use artificial intelligence algorithms and classifiers as well as data mining algorithms to build classification models used to classify emails based on extracted features. Most of those techniques perform very well with an accuracy of over 90% [14]. The performance depends on the number of features, feature selection, classification algorithm, mutations, and training model and data. Accuracy is not the only measure to evaluate spam filtering but it is a tradeoff between classifying legitimate ham as spam and the other way around. Factors to take into consideration in that tradeoff are cost, speed and user preferences.

Most of the filtering methods discussed earlier focus on email header and content analysis but spammers devise new techniques to evade detection such as, misspelling, fast flux networks and converting the spam email content into images. Leiba, Ossher, Rajan, Segal and Wegman used other features such as number of attachments, IP trace back and assigned reputation [15]. Others constructed a social network from the user's inbox and used it to classify new emails [16]. Optical character recognition (OCR) is used to detect image-based spam. Because OCR is computationally extensive, Wang, Josephson, Lv, Charikar and Li used other image properties such as color, length, width, size, size of the text portion, color histograms and orientation histograms [17].

III. Proposed System

Figure 1 shows a high-level block diagram of the proposed system. First, the system uses private and public lists to filter incoming emails based on header information. The users maintain the private lists locally, which are score based rather than simple black and white. An online public spam blocking service, which the user has no control over, maintains the public lists. In both lists, we tried to list only individual IPs that sent spam to the mail server and not entire network blocks. It is easy to remove IP listings and they expire after a predetermined time span.

The upcoming subsections focus on text processing, exploiting approximate bit-parallel string matching algorithm of Baeza Yates-Navarro for spam filtering systems and Naïve Bayes classification.

A. Text Processing

This stage consists of several processing steps: cleaning, tokenization, stemming and dropping stop words [18]. In the cleaning step, a linear reader removes all characters used to format the text such as diacritics and special characters such as currency signs, dots, colons, semicolons and double spaces, to name a few. Next is tokenization, which chops the text into small tokens. The tokenization process defines the first token by reading characters until it reaches the space character. The next token starts from where the tokenization process left to the next space and so on. Next in stemming where, tokens are reduced to their roots by removing all prefixes and suffixes such as 'in', 'ing' and 'er' [6]. The final step drops stop words such as 'at', 'is', 'are' and 'the'.

B. Matching Process

After text processing, the message arrives as a sequence of tokens at the matching stage. The matching process matches the sequence of tokens against patterns stored in the database by considering each token alone using Bayesian classifier to classify the message into spam or ham. The matching uses approximate bit-parallel algorithm with a specific value for edit distance. A match between the pattern and the message token is handled with maximum difference equal to the edit distance. To increase flexibility the edit distance is variable depending on the message word length. Table I lists the edit distance that corresponds to each token length.

The system trains on ham and spam messages and therefore the database contains information about the two classes. Therefore, the system is able to classify the messages that contain common words between the two classes based on statistical information. The user helps build the keywords in the database by flagging incoming messages as ham and spam. When a message is flagged as spam, the tokens from that message are added to the database and the sender of the message is added to the private black list. If the sender email is black listed in many private lists, then it will added to the public black list. Any message not considered spam is sent to the inbox and adds to the database of normal tokens.

C. Baeza-Yates and Navarro Approximate Matching

The string matching utilizes the improved bit-parallel approximate string matching algorithm by Baeza-Yates and Navarro. The non-deterministic finite automata (NFA) is defined by a $(k+1) \times (m+1)$ matrix of binary states, called s_{ij}, where active state means a match and inactive state means no match.

States of the first column, s_{i0}, are always active, signifying that the empty string is always matched. Subsequent columns are labeled in order of the characters of the pattern.

Next, scan the text once, character by character from beginning to end. For every new character of text, the entire NFA is updated in parallel according to the following rules, for each state s_{ij} , $i, j > 0$.

- Match: If $s_{(i-1)j}$ is active and the current text character matches the pattern character of column i, then s_{ij} becomes active.
- Replace: If $s_{(i-1)(j-1)}$ is active then s_{ij} becomes active.
- Insert: If $s_{i(j-1)}$ is active then s_{ij} becomes active.
- Delete: If $s_{(i-1)(j-1)}$ becomes active by any rule, then so does s_{ij}.

Similarly for $i = 0, j > 0$ but with only the first rule. If a state in the last column becomes active, then there is a corresponding match between the text and the pattern. It turns out it is not necessary to encode and update the entire original NFA. It suffices to consider the $(m - k)$ diagonals that start at some s_{0i}, $0 < i \leqslant m - k$ and extend downwards to the right [19]. The reason for this is that the lower left triangle of the NFA is always active and the upper right triangle such that if any state there becomes active then there will be at least one match. Ignoring the upper right triangle loses some information about how short a match can be made but this is generally considered to be of no concern at this point (formally we are only addressing the question of whether there is some match within edit-distance k) [7].

As an example, Fig. 2 shows what happens as the pattern "tit" is matched against the text "tat" while allowing for a maximum edit distance of one (meaning the number of rows in the NFA is two) [7].

TABLE II. THE MAXIMUM ALLOWED EDIT DISTANCE VALUE FOR EACH WORD LENGTH

Token Length	1	2	3	4	5	6	7	8	9	10	11	12
Maximum Edit Distance	0	0	1	2	2	3	3	3	3	4	4	4

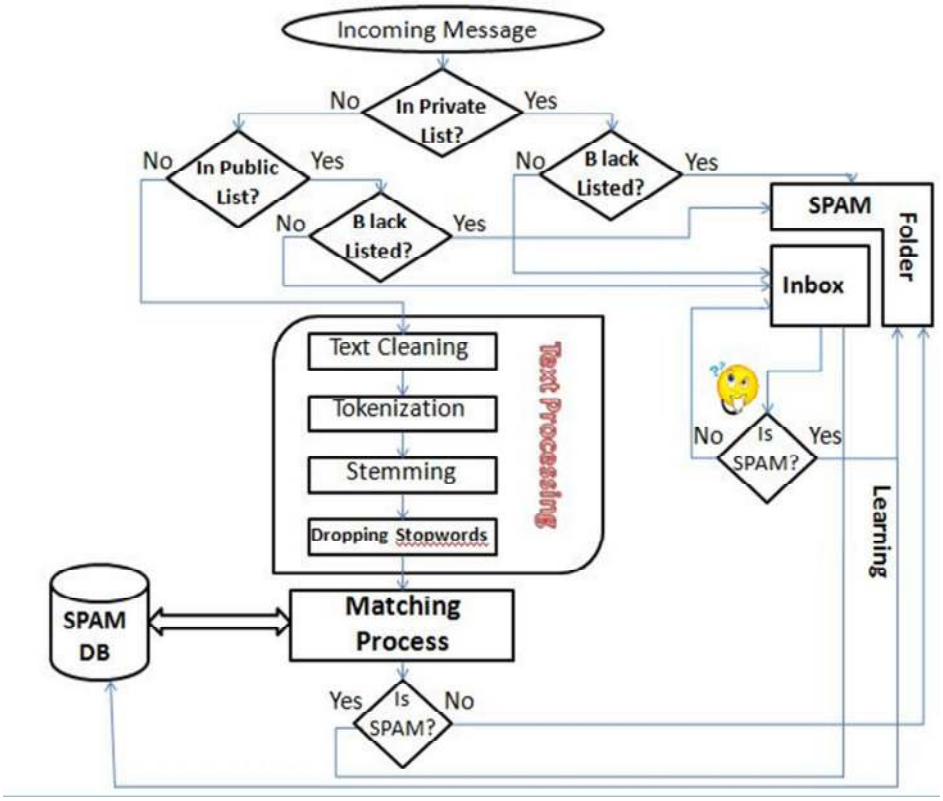

Fig. 1. Proposed spam detection system.

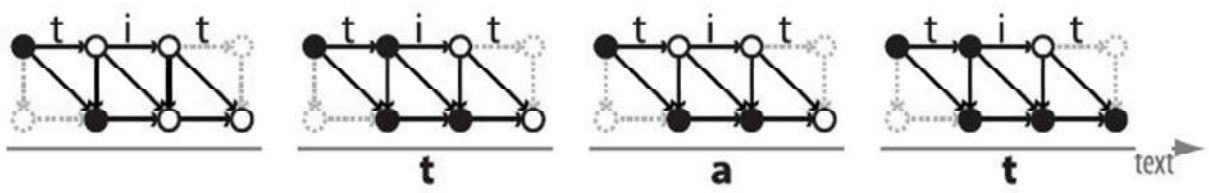

Fig. 2. Example of text matching [7].

D. Bayesian Classification

The Bayes theorem states that the probability a document, D_j , belongs to a class, C_i, can be as shown by (1).

$$P(C_i|D_j) = \frac{P(D_j|C_i)P(C_i)}{P(D_j)} \quad (1)$$

Where, *P(Ci|Dj)* is the probability the *Dj* document belongs to the *Ci* class. *P(Dj|Ci)* is the probability of document *Dj* given class *Ci*. *P(Ci)* is the probability of class *Ci*, equals to the number of words in the class divided by the number of words in all classes. Finally, *P(Dj)* is the probability of *Dj* document in the collection, and is assumed to be constant.

The document and class are modeled as a set of terms as in (2).

$$P(C_i|D_j) = \frac{\sum P(W_k|C_i)}{\text{Count(Document Terms)}} P(C_i) \quad (2)$$

Where, *Count(DocumentTerms)* is the number of terms in the document *Dj*. $P(W_k|Ci)$ is the probability of word W_k given class *Ci*, which is equal to the number of occurrences of the word W_k in the class *Ci* divided by the total number of words in Class *Ci*.

IV. Results and Evaluation

The simulation consists of three projects built using .NET 2003 platform. The spam filtering system (SFS) is ASP.NET web application coded in VB scripts, which consists of four pages: index, upload files, classification and information pages. The index page holds the menu to reach the other pages while the upload file page is for system learning. The classification page calculates the probabilities and classifies messages. Finally, the information page shows the database terms table, presents statistical information about the table, and searches it for a specific word statistics. The other two projects are Windows applications that compile into two libraries accessed by the main web application. SysCore written in VB.NET and contains functions and procedures to handle database transactions, file operations, text processing and message classification. BaezaYates is written in C++.NET and contains the approximate string matching algorithm functions.

A. Performance Metrics and Testing Data

In order to evaluate the performance of the proposed system we use accuracy, precision, recall and Receiver Operating Characteristic (ROC) curves. Table II shows the confusion matrix defined by Lutz Hamel [20] where accuracy, precision and recall are defined by (3) through (5).

To evaluate the proposed system we manually collect about 10000 email messages from work and personal email boxes over a period of six months. We call this set collected dataset (CDS2011) and manually classify the emails into 5000 ham and 5000 spam. Out of the balanced dataset, we use 8000 emails for training the system and the rest for testing. Additionally, we evaluate the system using the Spamassassin public mail corpus [21]. We test using the easy_ham, which includes 2500 ham messages easy to differentiate from spam, and spam_2, which includes 1397 spam messages.

TABLE II. Confusion Matrix Format

		Observed: Spam	Observed: Ham
Predicted	**Spam**	TP	FP
	Ham	FN	TN

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$\text{precision} = \frac{TP}{TP+FP} \quad (4)$$

$$\text{recall} = \frac{TP}{TP+FN} \quad (5)$$

B. Experimental Results

Tables III and IV show the confusion matrices for Naïve Bayes classifier over the CDS2011 and Spamassassin test emails for edit distance of *k=1*. Table V summarizes the proposed system performance numbers for both datasets. The proposed system achieves an accuracy of 97.2%, 98% precision and 96.4% recall for CDS2011 and an accuracy of 97.1%, 98.2% precision and 97.2% recall for the Spamassassin dataset. Accuracy represents the overall prediction accuracy for both spam and ham emails. Precision represents the ratio of correctly identified spam emails over all predicted spam emails while recall represents the ratio of correctly predicted spam emails over the total number of existing spam emails.

Fig. 3 shows the ROC curves for Naïve Bayes for both datasets. The larger the area under the curve the better the system because the area indicates a high true positives and low false positives rates. A point on the top-left corner of the ROC curve represents the ideal classifier and therefore the closer the curve to that square angle the better. In order to improve system performance, a better classifier than the simple Naïve Bayes is to be used. Currently we are simulating Random Forest Classifier (RF), which reported better results by earlier studies [22].

Finally, we vary the edit distance between *k=0* representing exact matching and *k=1* representing approximate matching. It can be seen from Fig. 4 that the true positives are a lot less in the case of exact matching because a lot of spam emails misspelled keywords by switching character with similar symbols. Larger edit distances catch those keywords thus increasing the detection accuracy on the expense of slower matching and more memory usage. By comparing the two performance curves, the classification process using exact string matching never reaches 100% spam detection rate and has greater false positive rate. Approximate matching with high true positives and low false positives rates, proves to be better than exact matching.

V. Conclusions

As spam emails increase rapidly, it is important to find an efficient spam filtering system that combines a small false positives rate with a high spam detection rate. In this work, we introduced a spam filtering system using inexact string matching algorithm to achieve better performance. The

introduced system shows a high spam detection rate with low false positives rate as opposed to systems using exact matching. The proposed system achieves a high 97.1% accuracy, 98.2% precision and 97.2% recall for standard testing databases. To improve performance further, random forest classifier is currently being simulated as a replacement for the simple Naïve Bayes classifier.

TABLE III. NAÏVE BAYES CONFUSION MATRIX FOR CDS2011

		Observed	
		Spam	**Ham**
Predicted	**Spam**	1928	39
	Ham	72	1961

TABLE IV. NAÏVE BAYES CONFUSION MATRIX FOR SPAMASSASSIN

		Observed	
		Spam	**Ham**
Predicted	**Spam**	1961	39
	Ham	72	1928

TABLE V. NAÏVE BAYES PERFORMANCE

	Accuracy	Precision	Recall
CDS2011	0.972	0.980	0.964
Spamassassin	0.971	0.982	0.972

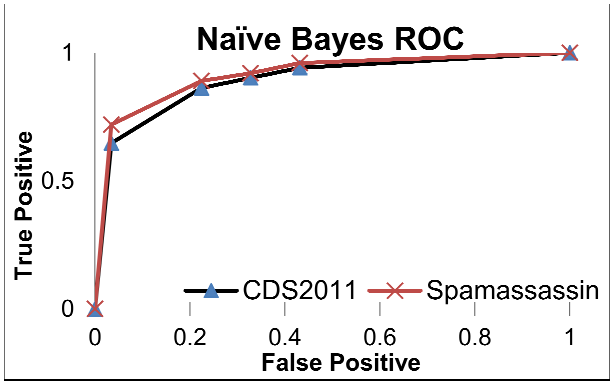

Fig. 3. ROC Curves for Naïve Bayes for CDS2011 and spamassassin datasets.

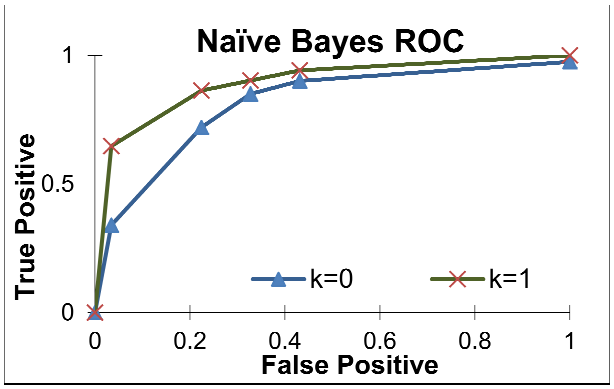

Fig. 4. ROC curves for Naïve Bayes for CDS2011 and variable edit distance (*k*)

REFERENCES

[1] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. Voelker, V. Paxson and S. Savage. "Spamcraft: an inside look at Spam campaign orchestration", *in Proc. 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009, Boston, Massachusetts.

[2] X. Hu, M. Knysz, and K.G. Shin, "Measurement and analysis of global IP-usage patterns of fast-flux botnets", *in Proc. INFOCOM*, 2011, pp.2633–2641.

[3] MessageLabs intelligence: 2010 Annual security report http://www.messagelabs.com/mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf

[4] B. Stone-Gross, T. Holz, G. Stringhini and G. Vigna. "The underground economy of Spam: a botmasters perspective of coordinating large-scale Spam campaigns", *in Proc. 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, Boston, Massachusetts.

[5] D. Sculley, G. Wachman and C.E. Brodley. "Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers", in Proc. TREC, 2006.

[6] R. Sigal, J. Crawford, J. Kephart and B. Leiba. "SpamGuru: an enterprise anti-Spam filtering system". IBM Thomas J. Watson Research Center, 2005.

[7] R. Baeza-Yates and G. Navarro, "Faster approximate string matching," Algorithmica, vol. 23, pp.127–158, 1999.

[8] H. Hyyro and G. Navarro. "Faster bit-parallel approximate string matching", *in Proc. 13th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 2002, pp. 203–224, Alberto Apostolico and Masayuki Takeda (Eds.). Springer-Verlag, London, UK.

[9] S. Nazirova. "Survey on Spam filtering techniques," Communications and Network, vol. 3, pp. 153–160, 2011.

[10] B. Agrawal, N. Kumar and M. Molle. "Controlling Spam emails at the routers," *in Proc. IEEE International Conference on Communications (ICC)*, vol. 3, pp. 1588–1592, 2005.

[11] Microsoft sender ID framework. http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx.

[12] D. Cook, J. Hartnett, K. Manderson, and J. Scanlan. "Catching Spam before it arrives: domain specific dynamic blacklists," *in Proc. Australasian Workshops on Grid Computing and e-research (ACSW Frontiers), pp.*193–202, Rajkumar Buyya, Tianchi Ma, Rei Safavi-Naini, Chris Steketee, and Willy Susilo (Eds.), vol. 54. Australian Computer Society, Inc., Darlinghurst, Australia, 2006.

[13] P. Kolari, A. Java, T. Finin, T. Oates and A. Joshi. "Detecting Spam blogs: a machine learning approach," *in Proc. 21st National Conference on Artificial Intelligence (AAAI)*, pp. 1351–1356, Anthony Cohn (Ed.), Vol. 2. AAAI Press, 2006.

[14] E. Blanzieri and A. Bryl. "A survey of learning-based techniques of email spam filtering," Artif. Intell. Rev. 29, 1, pp. 63–92, 2008.

[15] B. Leiba, J. Ossher, V.T. Rajan, R. Segal and M. Wegman. "SMTP path analysis," *in Proc. 2nd Conference on Email and Anti-Spam (CEAS),* Stanford University, 2005.

[16] P. Boykin, and V. Roychowdhury. "Leveraging social networks to fight spam," Computer, vol 38(4), pp. 61–68, 2005.

[17] Z. Wang, W. Josephson, Q. Lv, M. Charikar, and K. Li. "Filtering image spam with near-duplicate detection," *in Proc. 4th Conference on Email and Anti-Spam (CEAS)*, Mountain View, California, 2007.

[18] P.Z. Peebles, Probability, Random Variables, and Rand Signals Principles. 3rd ed., McGrae-Hill Inc., 1993.

[19] M. Onsjo, Y. Aono, and O. Watanabe. "Online approximate string matching with CUDA," Technical Report, Tokyo Institute of Technology, 2009.

[20] L. Hamel, Model Assessment with ROC Curves, the Encyclopedia of Data Warehousing and Mining, 2nd ed., 2009.

[21] Spamassassin public mail corpus. http://spamassassin.apache.org/publiccorpus/readme.html.

[22] I. Khater, Hierarchal Email Spam Filtering (HESF), MS Thesis, Jordan University of Science and Technology, 2011.