Multi-Classifier Classification of Spam Email on an Ubiquitous Multi-Core Architecture

Md. Rafiqul Islam, Jaipal Singh, Ashley Chonka and Wanlei Zhou
School of Engineering and Information Technology,
Deakin University, Melbourne, Australia
{rmd,jaipal,ashley,wanlei}@deakin.edu.au

Abstract

This paper presents an innovative fusion based multi-classifier email classification on a ubiquitous multi-core architecture. Many approaches use text-based single classifiers or multiple weakly trained classifiers to identify spam messages from a large email corpus. We build upon our previous work on multi-core by apply our ubiquitous multi-core framework to run our fusion based multi-classifier architecture. By running each classifier process in parallel within their dedicated core, we greatly improve the performance of our proposed multi-classifier based filtering system. Our proposed architecture also provides a safeguard of user mailbox from different malicious attacks. Our experimental results show that we achieved an average of 30% speedup at the average cost of 1.4ms. We also reduced the instance of false positive, which is one of the key challenges in spam filtering system, and increases email classification accuracy substantially compared with single classification techniques.

1 Introduction

The problem of unsolicited bulk email, more commonly known as spam, has been around since email was first used by the general public. In 2005, 80% of total email volume was considered spam [19]. The cost of managing spam is not proportional to the cost of sending these messages. While the cost of sending spam is negligible, the cost to corporations in terms of network resources, delayed emails and employee productivity is considerable and needs to be addressed. It is estimated that an average internet user spends 10 days a year dealing with spam [4]. There have been many proposals in dealing with spam, from legislation to recent advances in machine learning content classification.

While previous research in spam classification is primarily concerned with using a text based single classifier [15, 14] to detect spam messages, we have developed a novel spam filter architecture using a multi-classifier approach [13].

The use of multi-classifier systems provides a very high spam detection rate, but comes with high processing costs, if each of the classifiers is executed serially. In order for multi-classifier systems to be more efficient, classification needs to be done in parallel. Therefore, running our multi-classifier classification spam filtering system on a single core clustered-based computing or multi-core systems is ideal.

Cluster-based single core systems are usually thought as many computers that are coupled together to form a single virtual computer. By using this cluster to execute our multi-classifier algorithms, we are able to perform parallel operations, and achieving improved speed and performance. Multi-core CPUs were released in early 2000 but have become more affordable to the general public since 2005, and are combination of two or more independent microprocessor cores into a single chip [1].

In this paper we build upon our previous work [13, 8, 9], by combining our fusion based multi-classifier architecture with our ubiquitous multi-core framework, in order to provide high performance while at the same time improving the accuracy of spam detection. Based on our previous work in security and multimedia, we are seeing that our ubiquitous multi-core framework is able to be applied to most areas of computer science (as long as the system is multi-core).

Some of the benefits of MuM (multi-classifier ubiquitous multi-core) are as follows, firstly it will be cheaper to run and maintain in comparison to many high-end single core clustered computers; since parallel processing of data occurs within one CPU, keeping communication overheads much lower as the signal has to travel a shorter distance [6]. Secondly, we have discovered that complications are lessened in implementation, in comparison to a cluster of different machines.

Some of the weaknesses of MuM are: Firstly, cluster-based computing provides better redundancy compared to

MuM, but the cost of running a redundant multi-core machine would still be cheaper, compared to running multiple high-end cluster-based computers. Secondly, if MuM goes off-line, then the spam filter is completely off-line as well. However, having MuM distributed on multiple multi-core machines can be used as a backup in order to overcome this problem. The rest of this paper is organised as follows: Section 2 will provide a review of multi-core and multi-classifier spam filtering architectures. Section 3 details our proposed multi-core fusion based multi-classifier spam filter architecture. Section 4 presents the results of our proposed architecture followed by the conclusion.

2 Related Work

2.1 Multi-core processor architecture

The ability of manufacturing faster single core systems has reached a threshold due to the heat dissipation caused by placing too many transistors of a single chip. In order to overcome this problem, hardware manufacturers have developed multi-core CPU architectures [12, 2]. Multi-core or chip-level microprocessors (CMP) systems combine two or more independent microprocessor cores into a single chip [1]. Each microprocessor core has its own independent cache memory (L1) and share a common cache (L2) with other cores and peripheral devices. The next version of AMD multi-core processors will come with their own independent L1 and L2 caches.

In terms of software, a multi-core architecture provides improved multitasking performance by concurrently executing software codes on their own cores. Thus, multi-core architectures is ideal for parallel computing, where process threads can be run in parallel on different cores. Unfortunately, most of the existing software do not make use of parallel processing. Researchers are currently revisiting parallel programming for use in a multi-core environment [8, 9].

2.2 Bodyguard framework

We have developed a multi-core defence framework called bodyguard [8]. From this framework, we developed the Farmer bodyguard system. The basic hypothesis of the bodyguard framework, is to separate all security processes from other processes (email, browser, etc.), and assign them to a set of cores. The remaining cores within the system were assigned to the applications that require security. The bodyguard framework is made up of a Forward Bodyguard (FB) and Side Bodyguard (SB). For example, in our Farmer bodyguard system, the SB is responsible for providing a fast decision on whether to filter out any attack traffic. Upon detecting an attack, FB will then move in front of the application in order to protect it and initiate a filtering procedure. There are many advantages that come with the use of the bodyguard framework, such as efficient use of resources, performance increases and real-time detection and filtering.

2.3 Ubiquitous Multi-core (UM) multimedia

We have designed a ubiquitous multi-core framework and implemented it in the application of Bio-Inspired multimedia [9]. The Ubiquitous Multi-core (UM) framework is built from a divide-and-conquer approach, by dividing our applications and placing them on separate core processors. We found with our new multi-core framework, in the area of multimedia, the following benefits:

- By partitioning each application and its sub-tasks to separate cores, it will result in reducing the computational burden of the overall multi-core system.
- Memory storage requirements will be reduced, since each application is assigned its own L1 cache.
- If one of the applications is idle, then its core processor can be assigned to assist the other applications, this leads to a fully optimised usage of resources.
- Lastly, if one of the applications fails, then the rest of the doctor's applications are still able to function, while maintenance is completed.

2.4 Multi-classifier classification of spam email

Automated spam classification has traditionally been done using a single classifier technique. The classification algorithms such as SVM (Support Vector Machine) [10], NB (Naive Bayesian) [3] and Boosting [5] are used in content based spam filtering. These classification algorithms search for the most appropriate classifier in a search space that contains all the classifiers it can learn.

While single classifier techniques are fairly accurate, they require a lot of data for training. We performed comprehensive analyses of these classifiers in [13] and found that different classification algorithms return different results. The accuracy of the classification is reduced if it is used to classify generic content. In order to improve the classification accuracy, multi-classifier techniques were proposed [17, 18, 7].

The multi-classifier technique uses an ensemble of classifiers to classify email content. These classifiers are arranged in a two level hierarchy, with one classifier overseeing the results provided by two or more classifiers below it. These lower level classifiers are usually weakly trained, so as to reduce processing time. The lower level classifiers will provide a score for a message. The top level classifier will aggregate the results from the lower level classifiers and provides the final decision on whether an email message is spam or legitimate.

A combination of NB and k-Nearest Neighbour (k-NN) classifier technique in a stacking framework overseen by a memory-based classifier [17] was one of the first multi-classifier techniques used in spam detection. Several other ensemble techniques for spam detection include NB bagging [18] and semi-supervised labelled messages [7].

The current research has shown that aggregating scores from multiple classifiers improve the accuracy of classifying an email message. However, the current spam filter systems only use one or two types of classifiers, instead of a diverse range of classifiers, in their architecture. Since different classifiers provide different results, the reliance of similar classifier types will limit the classification accuracy.

In this paper, we are interested in using multi-classifier spam filters on a multi-core system to further improve the accuracy of detecting spam messages. There is currently no research in using multi-core for improving the accuracy of spam detection. In section 3, we will present an innovative multi-core framework for implementing multi-classifier classification architecture to detect spam emails.

3 Multi-classifier ubiquitous multi-core (MuM)

We have developed a generic fusion based multi-classifier classification spam filter architecture that eliminates misidentification of legitimate emails as spam (false positive) during spam detection. The spam filter architecture was originally designed and developed for a single SVM classifier as detailed in [15, 14]. We extended this architecture to be used for an ensemble based generic multi-classifier that processes the information in serial [13]. In this section, we are proposing a modified version of our spam classification architecture so that it can be executed in a multi-core environment, using different machine learning classification algorithms.

3.1 Design of multi-classifier classification filter

Figure 1 provides a description of our proposed email classification using multi-classifier classification (MCC) technique.

Our architecture should be used in a two-stage approach, at the mail server and at the recipient's mailbox. The email server will automate the email classification process while the user will be given the option to manually identify messages that do not fall collectively within the category of legitimate or spam. Emails messages that are cannot be identified as either legitimate (TP) or spam (TN) are termed grey

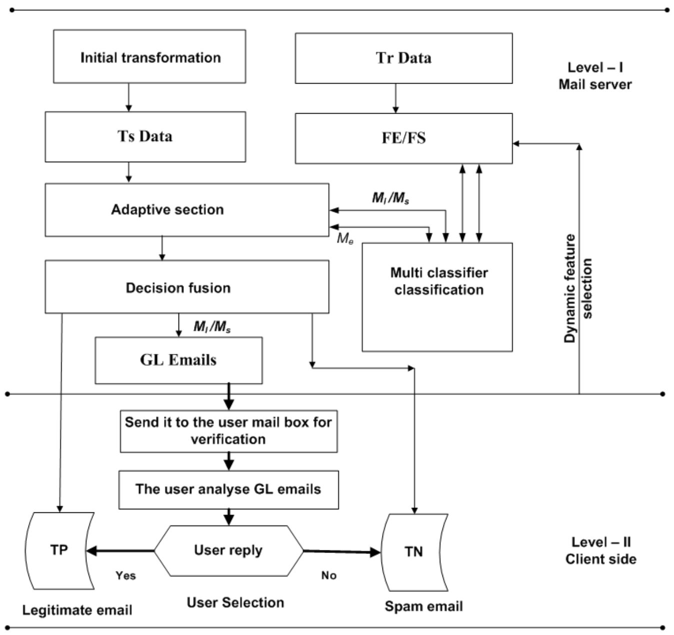

Figure 1. Multi-classifier classification (MCC) spam filter architecture.

list (GL) messages. The characteristics of email messages that have been successfully identified both legitimate and spam is used to retrain the multi-classifiers so as to reduce the requirement for the user to manually identify spam messages.

Before our architecture can be used to classify email messages, all of the classifiers need to be trained to recognise the attributes to be classified, whether they may be Boolean, frequency or N-gram attributes. The classifiers can be used to check for spam as well as non-spam attributes in an email message. This training is done using attributes extracted from training (T_T) data. The training process is an offline process that is done when the classifiers are idle.

In the first stage, the email server receives all incoming emails for the organisation. The server will index the email corpora. All of the incoming indexed emails (T_S) will be sent to the adaptive section. The main function of the adaptive section is to allocate the email messages to the classifiers and collect all of the results from the classifiers. The results of the email classification will be given a value of 1 for true positive (legitimate) or 0 for true negative (spam).

These results will be forwarded to the decision fusion to calculate the final result for an email message. If the decision fusion component receives the same results for a particular email messages, it can be classified as either legitimate (TP) or spam (TN). If the total result is not 0 or 1, that email is a GL email. This process can be represented in equation 1.

$$f(WM_e) = \frac{\sum_{i=1}^{N} C_i M_e}{N} \tag{1}$$

The results of N classifiers (C_i) for message (M_e) will return $f(WM_e) = 1$ for true positive (legitimate), $f(WM_e) = 0$ for true negative (spam) and $0 < f(WM_e) < 1$ for grey list.

All of these messages, legitimate, spam and grey list will be forwarded to the user's mailbox. The detailed functions and algorithms for the message transformation, adaptive section, feature extraction/feature selection (FE/FS) and decision fusion mechanisms will not be shown in this paper as they can be found in [15].

Our approach of using three categories for email messages, legitimate (white list), spam (black list) and unidentified (grey list) will provide greater accuracy in classifying spam messages and legitimate messages from the vast bulk of emails received by the email server. Once the messages have been categorised, they will be sent to the user (stage 2).

The second stage occurs once the email messages are received by the user. These messages will be presented in their respective mailboxes. The true positive (legitimate) emails will be sent to the user's inbox (TP) while the true negative (spam) messages will be in the spam mailbox (TN). The unidentified messages (GL) will be analysed by the user to categorise them as legitimate or spam. Such an approach is beneficial since categorising email messages is subjective. Some user might consider a message to be spam while other users might consider the same message as legitimate.

Those GL messages that are legitimate will be sent to the user's inbox while the spam messages will be in the spam mailbox. In order to automate the process of detecting spam messages, our architecture has a dynamic feature selection component. This component (FE/FS) will extract the relevant features from the legitimate and spam email messages and send this data to the mail server in order to train the classifiers. This training data ($\mathrm{T_r}$) will be updated every time the user identifies a grey list email. This ensures that the messages identified as legitimate or spam is according to the personal preference of the user.

In our approach, the use of grey lists provides the user with fine-grain controls to classify messages as legitimate or spam. We also update the training data on the mail server using dynamic feature selection so that most of the e-mail messages will be identified correctly before reaching the user. These two stages ensure that our multi-classifier classification architecture is scalable and can eliminate the false positive problem.

3.2 Design of Multi-classifier classification filter with Ubiquitous Multi-core framework (MuM)

In order to improve the classification of email messages, we build upon previous work with multi-core systems by developing a fusion based ubiquitous multi-classifier architecture (FUMA). FUMA uses fully trained data sets to generate results and supports any machine learning classifier technique such as SVM, NB, k-NN and Boosting.

The execution of all the classifiers is done in parallel so as to reduce the time taken to classify a message. In a single core CPU, the execution of multiple classifiers at the same time will fully utilise the available CPU power. We believe by our application of the ubiquitous multi-core framework, we greatly improve the performance and resource usage of our multi-classifier architecture.

While most multi-core research look at improving the communication between cores and application, through our development of an ubiquitous multi-core framework, we will reduce the CPU computation of n full featured classifiers in order to correctly identify legitimate email messages from spam messages.

Our proposed multi-classifier classification filter with ubiquitous multi-core (MuM) framework used by the classifiers is shown in Figure 2. Each of the classifiers (Classifier-n) in the spam filter system will run on their own independent core. The same email input will be sent by the adaptive section to each of the classifiers. The classifiers will run in parallel, thus improving the speed in analysing the emails. Each classifier process ($\mathrm{C_n}$) will execute their sub processes ($\mathrm{P_m}$) in parallel. Once the classifiers have completed their analysis, they will send the results back to the adaptive section as described in section 3.1.

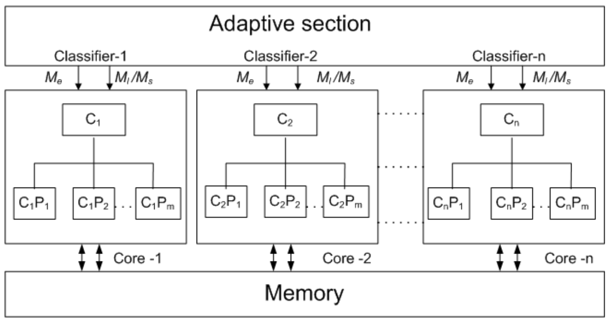

Figure 2. Multi-classifier classification filter with ubiquitous multi-core (MuM).

3.3 Benefits of MuM

Through our use of ubiquitous multi-core framework we are summarizing the following key benefits of MuM:

Firstly, the partitioning of each classifier and its tasks to a separate core will reduce the computation burden of the overall mail server system.

Secondly, the reduction of memory storage requirements for email messages (T_S). Since the same email messages (T_S) are sent to all classifiers, the system buffer will store the email message once.

Thirdly, in terms of processing time, all of the classifiers will process the email messages in parallel. Unlike ensemble based multi-core architectures, our approach allows a classifier to process an email message independently from other classifiers. This allows for faster processing of messages compared to other architectures.

Fourthly, the multi-classifiers are trained using (T_r) data when the system is idle. Since each core is independent, the training can be done at different times as some cores will complete the classification tasks faster than other cores. This will mean that the mail server resource usage will be optimised.

Lastly, MuM is robust, as the adaptive selection can still provide accurate email classification if one of the core fails. The adaptive selection component can choose to either reduce the number of classifiers or redistribute the classifier (and the sub-tasks) to another core. Although this is a non-optimum solution, this is a robust solution in the event of one of the core fails during operation.

4 Results

We evaluate the performance and accuracy of our MuM by simulating our fusion based multi-classifier classification architecture on a 4 core multi-core system. We have dedicated 3 cores for implementing 3 different classifiers and use the fourth core to implement the other components, such as initial transformation, adaptive section and decision fusion, of our multi-classifier architecture (figure 1). This method ensures that the performance of the classifiers is not affected by the adaptive section and decision fusion components. The performance of MuM is described in detail below.

4.1 Multi-core performance benchmark

Once we have the execution times t_s, computational time t_{comp}, and communication time t_{com}, we can establish what the speedup factor (formula 2) and computation/communication ratio (formula 3) from a single core to multi-core system.

$$\text{speedup factor } = \frac{t_s}{t_{cp}} = \frac{t_s}{t_{comp} + t_{com}} \tag{2}$$

where t_s will stand for execution time on a single core processor (t_{cp}), this includes computation time and communication time. The Computation/Communications ratio is derived from [11].

$$\text{C/C ratio} = \frac{t_{comp}}{t_{com}} \tag{3}$$

Apart from speedup and the Computation/Communications ratios, we also evaluate the multi-classifier algorithm, through the use of Time Complexity or "big-oh", also referred to as "order of magnitude" [16].

$$f(x) = O(g(x)) \tag{4}$$
$$[0 \leq f(x) \leq cg(x)] \text{ for all } x \geq 0$$

where $f(x)$ and $g(x)$ are functions of x. A positive constant, c, has to exist for all otherwise it is zero. To evaluate Time complexity, we use the total sum of computation and communication (formula 2).

$$\begin{aligned} \text{Time Complexity} &= T_p \\ &= t_{comp} + t_{com} \\ &= (\frac{n}{cp+1}) + \\ &\quad (2t_{startup} + (\frac{n}{cp+1})t_{msgdata}) \end{aligned} \tag{5}$$

where n is the number of threads on each core processor. The last benchmark we will use is the cost and cost-optimal.

Cost	=	execution time * total number of processor used
Cost Optimal	=	time complexity * number of processor
	=	(n log n)

4.2 Multi-core system evaluation

To measure and evaluate the performance of MuM, we wrote 4 simple programs to simulate the multi-classifier applications. We assigned them to 4 cores within our multi-core system by using affinity methods. The multi-classifier functions are simulated, by the 4 programs just to demonstrate our framework, though 3 actual multi-classifier programs are planned in the future.

Based on our evaluations, displayed in table 1 and figure 3, we see that a speed average of 30% was archieved at the average cost of 1.4ms. This is achieved by separating out each application and allowing them to run on their own separate cores. The time complexity results also show that the efficiency of our algorithm is at 3.0. This means that for 4 computational steps (estimate) we achieved 3 data items.

So, the more computations that are done the more data items we complete. For example, 5 computational steps will give us 3.5 data items. One of the results, the Computation/Communication Ratio, shows that it was less than Time Complexity. This means, it will not improve speedup or efficiency beyond the figures we already have. Lastly, we see that the cost of running our program was below the cost-optimal, and at the same time achieving an average of 95% CPU (see figure 3). This means that our model/program was quite cost efficient to run and resource usage (CPU) was almost fully optimised. Since the Time Complexity is higher then Computation/Communication Ratio, it would not be worthwhile trying to send our costs up to reach the optimal threshold, since we would gain no performance benefit.

Table 1. Results of multi-core speedup and the costs.

	Core 1	Core 2	Core 3	Core 4
Exe Time	1.4ms	1.5ms	1.4ms	1.4ms
Comp Time	0.4 ms	0.10ms	0.4ms	0.09ms
Comm Time	1ms	0.5ms	1ms	0.5ms
Speed Ratio	50%	20%	50%	20%
C/C	0.4	2	0.4	1.8
Time Complex	3	3	3	3
Cost	1.4	1.5	1.4	1.4
Cost-Optimal	2	2	2	2

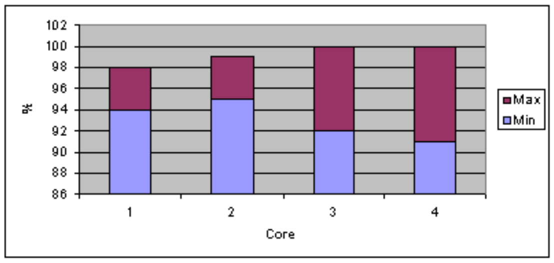

Figure 3. Min(90%)-Max(100%) CPU usage that was archieved during our simulation.

4.3 Performance of multi-classifier classification

We implemented 3 text based classifier algorithms to measure their accuracy on a single core system compared with our multi-core system. We executed scalable vector machine (SVM) classifier on core-1, AdaBoost classifier on core-2 and Naive Bayesian classifier on core-3. Each of these classifiers was implemented in their own process

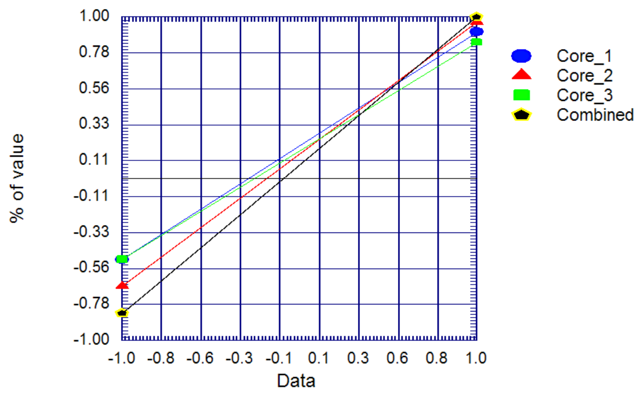

Figure 4. The comparison of average precision-recall curve.

thread to simulate a multi-core system. We ran each type of classifier through 6 email data sets from public data set PUA [3]. We have converted the data sets in six different parts based on our experimental setup to test their accuracy in correctly detecting spam messages.

Table 2 shows the precision (false positive) and recall (false negative) in classifying email messages. From all the results, our proposed multi-classifier system did not return any false positive values. We strongly state the case that the use of a fusion based multi-classifier will eliminate the false positive problem. Our simulations also clearly show that a multi-classifier returns very few false negative results compared with just using single classifier algorithms.

Figure 4 shows the comparison of average precision-recall curve of data sets for individual cores output along with combined three cores outputs. It has been shown from the fig. 4 that the precision is always better for combined result compared to individual result and it is 100 percent for all data sets which is promising. It is clear that the multi classifier classification approach reduces the instance of false positive to the zero level.

The average receiver operating characteristic (ROC) report of our multi-classifier approach is shown in table 3. From the results of our experiments with 6 datasets, the accuracy of our fusion based multi-classifier system is 10% to 13% higher than any single classifier algorithm.

Figure 5 shows the ROC curve for sensitivity and specificity analysis of the classifier classification algorithm accuracy in detecting spam from legitimate email messages. The figures show the average ROC curve for our datasets.

5 Conclusion

In this paper, we introduced a novel multi-core based framework that we used in our fusion based multi-classifier

Table 2. Comparison of precision-recall of individual cores with combined cores.

Data	Condition variable	Core-1	Core-2	Core-3	Combined-cores
Data 1	-1	-0.5555556	-0.7777778	-0.5555556	-0.7777778
	1	1	0.96875	1	1
Data 2	-1	-0.4285714	-0.7142857	-0.7142857	-0.9285714
	1	0.7777778	1	0.6666667	1
Data 3	-1	-0.4545455	-0.6363636	-0.2727273	-0.6363636
	1	0.8571429	1	0.7142857	1
Data 4	-1	-0.5555556	-0.5555556	-0.5555556	-0.9444444
	1	1	0.875	1	1
Data 5	-1	-0.5384616	-0.2307692	-0.3846154	-0.6923077
	1	0.8333333	1	0.8333333	1
Data 6	-1	-0.6	-0.2	-0.4	-0.8
	1	0.8666667	1	0.7333333	1

Table 3. Average ROC of classification algorithms.

Criterion	Estimate of AUC	AUC's Standard Error	Lower 95% Confidence Limit	Upper 95% Confidence Limit
Core-1 (SVM)	0.852818333	0.0378933	0.75851	0.911685
Core-2 (Boost)	0.87327	0.033725	0.78470667	0.923985
Core-3 (NB)	0.826265	0.039185	0.73109	0.8886267
Combined-cores	0.949143333	0.0228333	0.86701167	0.9777817

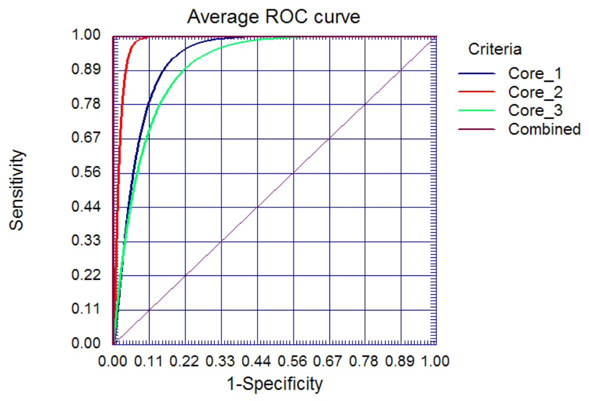

Figure 5. The average sensitivity and specificity curve.

classification spam filter architecture. Our proposed multi-core framework ensures that the multi-classifiers perform more efficiently, thus reducing the burden on system resources while detecting spam from email messages. We have shown through simulations that our proposed multi-classifier architecture performs better than any other text based single classifier system. Our multi-classifier architecture eliminates all false positive results when detecting spam.

In the near future, we are planning to implement our multi-classifier system on an actual multi-core based enterprise grid to gain better results, particularly in terms of the number of classifiers required to provide high accuracy without incurring high system resource usage.

References

[1] Advanced Micro Devices. Multi-core processors - the next evolution in computing. Available from: http://multicore.amd.com/Resources/33211A_Multi-Core_WP_en.pdf [Accessed on: 18 Apr. 2008], 2005.

[2] Advanced Micro Devices. Amd multi-core products. Available from: http://multicore.amd.com/us-en/AMD-Multi-Core/Products.aspx [Accessed on: 20 Apr. 2008], 2008.

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive

bayesian anti-spam filtering. In *Proc. of the 11th European Conference on Machine Learning*, pages 9–17, 2000.

[4] J. M. Carpinter. Evaluating ensemble classifiers for spam filtering. Honours thesis, University of Canterbury, 2005.

[5] X. Carreras and L. Marquez. Boosting trees for anti-spam email filtering. In *Proc. of the European Conference in Recent Advances in NLP*, pages 58–64, 2001.

[6] L. Chai, Q. Gao, and D. K. Panda. Understanding the impact of multi-core architecture in cluster computing: A case study with Intel dual-core system. In *Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid*, pages 471–478, May 2007.

[7] V. Cheng and C. Li. Personalized spam filtering with semi-supervised classifier ensemble. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 195–201, 18-22 Dec. 2006.

[8] A. Chonka, W. Zhou, K. Knapp, and Y. Xiang. Protecting information systems from ddos attack using multicore methodology. In *Proc. of the IEEE 8th International Conference on Computer and Information Technology*, 2008.

[9] A. Chonka, W. Zhou, and Y. Xiang. Bio-inspired multimedia using ubiquitous multicore (um) methodology. In *Proc. of the International Workshop on Multimedia Signal Processing*, 2008 [Currently under review].

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

[11] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison Wesley, 1995.

[12] Intel. Intel multi-core technology. Available from: http://www.intel.com/multi-core/index.htm [Accessed on: 20 Apr. 2008], 2008.

[13] M. R. Islam and W. Zhou. Email categorization using multi-stage classification technique. In *Proc. of the 8th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 51–58, 3-6 Dec. 2007.

[14] M. R. Islam and W. Zhou. An innovative analyser for email classification based on grey list analysis. In *Proc. of the IFIP International Conference on Network and Parallel Computing*, pages 176–182, 18-21 Sept. 2007.

[15] M. R. Islam, W. Zhou, and M. U. Choudhury. Dynamic feature selection for spam filtering using support vector machine. In *Proc. of the 6th IEEE/ACIS International Conference on Computer and Information Science*, pages 757–762, 11-13 July 2007.

[16] D. E. Knuth. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2):18–24, Apr.-June 1976.

[17] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proc. of the 6th Conference on Empirical Methods in Natural Language Processing*, pages 44–50, 2001.

[18] Z. Yang, X. Nie, W. Xu, and J. Guo. An approach to spam detection by naive bayes ensemble based on decision induction. In *Proc. of the 6th International Conference on Intelligent Systems Design and Applications*, pages 861–866, Oct. 2006.

[19] T. Zeller Jr. Law barring junk e-mail allows a flood instead. Available from: http://www.nytimes.com/2005/02/01/technology/01spam.html [Accessed on: 19 April 2008], 1 Feb. 2005.