# Lazy Associative Classification for Content-based Spam Detection*

Adriano Veloso
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, Brazil
adrianov@dcc.ufmg.br

Wagner Meira Jr.
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, Brazil
meira@dcc.ufmg.br

## Abstract

*Despite all tricks and mechanisms spammers use to avoid detection, one fact is certain: spammers have to deliver their message, whatever it is. This fact makes the message itself a weak point of spammers, and thus special attention has being devoted to content-based spam detection. In this paper we introduce a novel pattern discovery approach for spam detection. The proposed approach discovers patterns hidden in the message, and then it build a classification model by exploring the associations among the discovered patterns. The model is composed by rules, showing the relationships between the discovered patterns and classes (i.e., spam/legitimate message). Differently from typical eager classifiers which build a single model that is good on average for all messages, our lazy approach builds a specific model for each message being classified, possibly taking advantage of particular characteristics of the message. We evaluate our approach under the* TREC *2005* Spam Track *evaluation framework, in which a chronological sequence of messages are presented sequentially to the filter for classification, and the filter is continuously trained with incremental feedback. Our results indicate that the proposed approach can eliminate almost 99% of spam while incurring 0.4% legitimate email loss. Further, our approach is also efficient in terms of computational complexity, being able to classify more than one hundred messages per second.*

## 1 Introduction

Email is an increasingly important means of communication, both facilitating contact between individuals and enabling rises in the productivity of organizations. However, spammers are continously crawling the Web for email addresses available at Web pages, so that more and more people can be reached, thus eroding away much of the attractiveness of email communication.

A typical user of email usually spend some minutes a day removing spams from his/her inbox. This task is very annoying and represents a wastage of time. Even worse, if this time is multiplied by the total number of individuals that are simultaneously dealing with the same task of removing spam, the result is an unwanted work of extraordinary magnitude. Thus, spam has become a problem of enormous significance, and spam blocking has become essential for email to remain a viable form of communication (specially for Web-based email systems).

Several approaches to combat spammers work by setting predefined barriers in order to block spams. These approaches operate on the assumption that some facets of an email are highly indicative of spam. However, spammers are continuously evolving [9, 12] and they have shown to be excellent in adapting themselves to circumvent those predefined barriers [4]. Despite the evolving nature of spammers, the fact is that they will always have to deliver their message, whatever it is. For the human recipient, a spam message is easily recognizable. Content-based spam detection is the automation of this recognition process, and it works based on the premisse that discrimant patterns can be discovered and used for sake of classying a message.

In this paper we introduce a novel content-based spam detection approach. Our proposed approach first uncovers patterns hidden in both spam and legitimate messages, and then it associates the discovered patterns with the corresponding class (i.e., spam or legitimate message). These associations are presented in the form of rules $\mathcal{X} \rightarrow c$, where $\mathcal{X}$ is a pattern and $c$ is either $ham$ or $spam$. Instead of generating a set of rules that are good on average for all messages, our approach employes a lazy rule induction that generates a specific set of rules for each message, possibly taking advantages of particular characteristic of each message. A voting mechanism in which each rule $\mathcal{X} \rightarrow c$ is viewed as a vote from pattern $\mathcal{X}$ for class $c$, is then performed in order to elect the correct class. Our approach is

*This research was sponsored by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program, process number 200503312358.

able to evolve with spammers, in the sense that novel patterns are quickly discovered and automatically used for sake of classification. Further, our approach showed to be efficient in terms of computational complexity, being able to classify more one hundred messages per second.

Our evaluation models real filter usage as closely as possible by applying the TREC 2005 *Spam Track* evaluation framework [8], in which a chronological sequence of email messages are sent for classification. A feedback is provided after each message classified, and our filter automatically adapts itself to new messages. Our results indicate that our approach can eliminate more than 98% of spam while incurring in 0.4% innocent email loss.

We organized the paper as follows. Section 2 presents some technical background which are necessary to better understand our approach, which is introduced in Section 3. We present our experimental evaluation in Section 4. Related work is discussed in Section 5. Finally we conclude in Section 6.

## 2 Preliminaries

In this section we provide preliminary concepts of associative classification for spam detection, which are necessary to better understand our proposed technique.

**Definition 1.** [Messages] A *message* $\mathcal{M}_j$ is a non-empty set of items (i.e., words or symbols), where $j$ is a natural number called the message identifier. The training data $\mathcal{D}$ is a finite set of messages for which the category (i.e., spam or legitimate) is known. Two messages are similar if they have at least $\delta$ items in common, where $\delta$ is a user specified parameter. Let $\mathcal{I}$ denote the set of all $n$ distinct items in $\mathcal{D}$ $\{i_1, i_2, \ldots, i_n\}$.

**Definition 2.** [Patterns] A non-empty subset of $\mathcal{I}$ is called a *pattern*. For any pattern $\mathcal{X} \subseteq \mathcal{I}$, its size is the number of elements in $\mathcal{X}$. A pattern of size $k$, $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ is called a $k$-pattern. The support of $\mathcal{X}$ is the percentage of messages in $\mathcal{D}$ that contain $\mathcal{X}$ as a subset, given as:

$$\sigma(\mathcal{X}) = 100 \times \frac{|\{\mathcal{M}_j \in \mathcal{D} | \mathcal{X} \subseteq \mathcal{M}_j\}|}{|\mathcal{D}|} \text{ (with } 1 \leq j \leq |\mathcal{D}|). \quad (1)$$

A pattern is frequent if $\sigma(\mathcal{X}) \geq \sigma_{min}$, where $\sigma_{min}$ is a user specified minimum support threshold. Informally, a frequent pattern is a piece of message that occurs repeated times in different messages.

**Definition 3.** [Association Rules] An association rule is a rule with the form $\mathcal{X} \rightarrow c$, where $c$ is the message category (i.e., spam or legitimate), and $\mathcal{X}$ is a pattern. The confidence of the rule indicates how strong $\mathcal{X}$ is associated with category $c$, and is given as:

$$\theta(\mathcal{X} \rightarrow c) = 100 \times \frac{\sigma(\mathcal{X} \cup c)}{\sigma(\mathcal{X})}. \quad (2)$$

A rule $\mathcal{X} \rightarrow c$ is strong if $\mathcal{X}$ is frequent and $\theta(\mathcal{X} \rightarrow c) \geq \theta_{min}$, where $\theta_{min}$ is a user specified minimum confidence threshold. A rule $\mathcal{X} \rightarrow c$ is applicable to message $\mathcal{M}_j$ if $\mathcal{X} \subseteq \mathcal{M}_j$.

### 2.1 Generating Association Rules

Given a set of messages $\mathcal{D}$, and thresholds $\sigma_{min}$ and $\theta_{min}$, the task of generating association rules is to find all strong rules in $\mathcal{D}$. This task can be divided in two steps:

1. Enumerate frequent patterns: The set of all patterns $\mathcal{X}$ for which $\sigma(\mathcal{X}) \geq \sigma_{min}$ is generated.
2. Generate strong rules: The set of all rules $\mathcal{X} \rightarrow c$ for which $\mathcal{X}$ is frequent and $\theta(\mathcal{X} \rightarrow c) \geq \theta_{min}$ is generated.

The first step is more complex than the second one. A naive approach is to first compute all patterns in $\mathcal{D}$ and then return only those that are frequent. This approach is inappropriate because the number of all possible patterns grows exponentially. Fortunately, the following Theorem provides a powerful pruning strategy. Once pattern $\mathcal{X}$ is known to be not frequent, then no superset of $\mathcal{X}$ need to be generated since it must also be not frequent.

**Theorem 1.** If $\mathcal{X}$ is a frequent pattern, then all subsets of $\mathcal{X}$ are also frequent [2].

Thus, threshold $\sigma_{min}$ actually impacts in the number of patterns that will be generated, and consequently, in the time that will be spent to generate them. If $\sigma_{min}$ is set too low, then several patterns will be generated. On the other hand, if $\sigma_{min}$ is set too high, only few patterns will be generated.

Once all frequent patterns are generated, the second step becomes straightforward. For each frequent pattern $\mathcal{X}$, the rule $\mathcal{X} \rightarrow c$ is generated, and if $\theta(\mathcal{X} \rightarrow c) \geq \theta_{min}$ then it is a strong rule.

## 3 Lazy Associative Spam Detection

In this section we present our lazy classification approach for spam detection, which consists of generating frequent patterns (i.e., pieces of message), and mapping them to the corresponding classes. This mapping is done through the discovery of association rules $\mathcal{X} \rightarrow c$, where the antecedent $\mathcal{X}$ is a combination of different pieces of message, and the consequent $c$ is a class.

Let $\mathcal{D}$ be a set of messages
Let $\mathcal{F}$ be the set of all frequent patterns
1. $\mathcal{F} \leftarrow \emptyset$
2. $\mathcal{F}_1 \leftarrow$ all frequent 1-patterns
3. $k \leftarrow 1$
4. while $\mathcal{F}_k \neq \emptyset$ do
5. $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}_k$
6. $\quad k \leftarrow k + 1$
7. $\quad \mathcal{F}_k \leftarrow$ all frequent $k$-patterns
8. return $\mathcal{F}$

**Figure 1. Frequent Pattern Enumeration.**

### 3.1 Pattern Enumeration and Rule Induction

The starting point of our classification approach is the frequent pattern enumeration. Frequent patterns are generated in a level-wise manner, and the support of each pattern is calculated using Equation 1. First, all 1-patterns (i.e., patterns of size one) are generated. Then, the frequent 1-patterns are used to generate 2-patterns. This process continues generating longer patterns (always using frequent $(k-1)$-patterns to generate $k$-patterns) until all frequent patterns are found. This process is shown in Figure 1. Step 7 is the core of the frequent pattern enumeration, where $k$-patterns are generated by combining two frequent $(k-1)$-patterns. Efficient methods for subset combination and support computation were already proposed in the literature [2, 18].

After all frequent patterns are found, strong rules are induced. The confidence of each rule is calculated using Equation 2. Figure 2 shows the rule induction process.

### 3.2 Ranking Rules and Voting Process

After induction, the strong rules are sorted/ranked in ascending order of $\theta$. Ties are broken by also considering their $\sigma$ values, again in ascending order. Any remaining ties are broken arbitrarily. The resulting ranking, given by $\mathcal{R}$, is then used to assign a numerical weight to each rule; the weight being the rank/position of the rule in $\mathcal{R}$, given by $\mathcal{R}[\mathcal{X} \rightarrow c]$. Thus each rule $\mathcal{X} \rightarrow c \in \mathcal{R}$ is interpreted as a (weighted) vote by pattern $\mathcal{X}$ for class $c$. Higher ranked rules thus count for more in the voting process. Formally,

Let $\mathcal{F}$ be the set of all frequent patterns
Let $\mathcal{R}$ be the set of all strong rules
1. $\mathcal{R} \leftarrow \emptyset$
2. for each frequent pattern $\mathcal{X} \in \mathcal{F}$ do
3. $\quad$ if $\theta(\mathcal{X} \rightarrow ham) \geq \theta_{min}$ then
4. $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{X} \rightarrow ham\}$
5. $\quad$ if $\theta(\mathcal{X} \rightarrow spam) \geq \theta_{min}$ then
6. $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{X} \rightarrow spam\}$
7. return $\mathcal{R}$

**Figure 2. Strong Rule Induction.**

Let $\mathcal{R}$ be a sorted set of strong rules
1. score(ham)$\leftarrow 0$
2. score(spam)$\leftarrow 0$
3. for each rule $\mathcal{X} \rightarrow ham$ do
4. $\quad$ score(ham)$\leftarrow$ score(ham)+weight($\mathcal{X}$, ham, $\mathcal{R}$)
5. for each rule $\mathcal{X} \rightarrow spam$ do
6. $\quad$ score(spam)$\leftarrow$ score(spam)+weight($\mathcal{X}$, spam, $\mathcal{R}$)
7. if $\frac{score(spam)}{score(ham)} \geq \lambda$ then return $spam$
8. return $ham$

**Figure 3. Voting Process.**

the weighted vote given by pattern $\mathcal{X}$ for class $c$ is given by:

$$weight(\mathcal{X}, c, \mathcal{R}) = \begin{cases} \mathcal{R}[\mathcal{X} \rightarrow c], if \mathcal{X} \rightarrow c \in \mathcal{R} \\ 0, \text{otherwise} \end{cases} \quad (3)$$

Finally, the score of a class is the sum of the weighted votes assigned to it, represented by the function:

$$score(c) = \sum_{\mathcal{X} \rightarrow c \in \mathcal{R}} weight(\mathcal{X}, c, \mathcal{R}). \quad (4)$$

The poll/voting process is shown in Figure 3. Because the relative costs associated with ham and spam misclassification may vary from one situation to another, a cost factor $\lambda$ is introduced. If the ratio between ham and spam scores is no less than $\lambda$, then the message is classified as spam. Otherwise, it is classified as a ham. The cost factor $\lambda$ may be adjusted to decrease ham misclassification at the expense of spam misclassification, or vice-versa.

Let $\mathcal{D}$ be the set of all messages already classified
Let $\mathcal{M}$ be the set of all messages to be classified
Let $\mathcal{R}$ be the sorted set of all strong rules in $\mathcal{D}$
1. $\mathcal{D} \leftarrow \emptyset$
2. for each message $\mathcal{M}_i \in \mathcal{M}$ do
3. $\quad \mathcal{R} \leftarrow$ all strong rules in $\mathcal{D}$
4. $\quad$ perform poll using rules in $\mathcal{R}$
5. $\quad$ predict the winner class ($ham$ or $spam$)
6. $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}_i$

**Figure 4. Eager Email Classification.**

### 3.3 Eager and Lazy Spam Detection

In this section we introduce two approaches for spam detection. Both approaches use the strategies discussed in Sections 3.1 and 3.2. The difference of the two spam detection approaches that will be presented next is the set of rules that is used for sake of classification. In the eager approach, a set of strong rules is generated based on all known messages. Figure 4 shows the basic steps of the eager approach. The strong rules that are generated in step 3 are induced by all $i-1$ known messages (i.e., $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{i-1}\}$), and as a consequence, a large fraction of the generated rules may not be relevant for classifying message $\mathcal{M}_i$, representing a wastage of computational resources. Further, if the items within the message $\mathcal{M}_i$ have occurred rarely in the known messages (i.e., $< \sigma_{min}$), there will be no rules for classifying message $\mathcal{M}_i$. In that way, if $\sigma_{min}$ is set too high, then novel patterns will possibly take too much time to become frequent, and this may degrade accuracy. On the other hand, if $\sigma_{min}$ is set too low, then several useless rules will be generated, and this may degrade performance.

To overcome the aforementioned problems of the eager classification approach, we propose a lazy classification alternative, in which only the useful known messages are mined for generating only rules that are applicable to the message being classified (i.e., $\mathcal{M}_i$). The useful messages are conditioned to the message being classified, $\mathcal{M}_i$, and are represented by all the messages that are similar to $\mathcal{M}_i$.

Figure 5 shows the lazy classification approach. The difference from the eager approach is exactly in steps 3 and 4, where only the conditional set of messages that are similar to $\mathcal{M}_i$ are used for generating the rules. The support of pattern $\mathcal{X}$ must be redefined in order to accommodate these modifications:

$$\sigma(\mathcal{X}) = 100 \times \frac{\mid \{\mathcal{M}_j \in \mathcal{D}_i | \mathcal{X} \subseteq \mathcal{M}_j\} \mid}{\mid \mathcal{D}_i \mid} \text{ (with } 1 \leq j < i\text{).} \quad (5)$$

That is, only the messages in $\mathcal{D}_i$ (i.e., the messages in $\mathcal{D}$ that are similar to $\mathcal{M}_i$) are used to calculate the support of a pattern. These simple modifications are usually sufficient to overcome the problems associated with the eager classification approach. First, since all generated rules are induced by the message being classified all rules are applicable for sake of classification, and there is no wastage of resources. Second, since messages with rare items tends to have only few similar messages, novel patterns quickly become frequent and are incorporated to the set of rules.

Let $\mathcal{D}$ be the set of all messages already classified
Let $\mathcal{M}$ be the set of all messages to be classified
Let $\mathcal{D}_i$ be the set of messages that are similar to $\mathcal{M}_i$
Let $\mathcal{R}$ be the sorted set of all strong rules in $\mathcal{D}_i$
1. $\mathcal{D} \leftarrow \emptyset$
2. for each message $\mathcal{M}_i \in \mathcal{M}$ do
3. $\quad \mathcal{D}_i \leftarrow$ all messages in $\mathcal{D}$ that are similar to $\mathcal{M}_i$
4. $\quad \mathcal{R} \leftarrow$ all strong rules in $\mathcal{D}_i$ induced by $\mathcal{M}_i$
5. $\quad$ perform poll using rules in $\mathcal{R}$
6. $\quad$ predict the winner class ($ham$ or $spam$)
7. $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}_i$

**Figure 5. Lazy Email Classification.**

In the next section we empirically evaluated the proposed approaches.

## 4 Experimental Evaluation

In this section we describe the experimental results for the evaluation of our proposed classification approaches in terms of both classification effectiveness and computational efficiency.

We quantify the effectiveness of the approaches through the conventional *ham*, *spam*, and *overall misclassification percentages*. Ham misclassification percentage ($hm\%$) is the fraction of all innocent messages classified as spam. Similarly, spam misclassification percentage ($sm\%$) is the fraction of all spam messages classified as ham. Overall misclassification percentage ($om\%$) is the fraction of all messages incorrectly classified. We also employed the *logistic average misclassification percentage* [8] ($lam\%$), defined as $lam\% = logit^{-1}(\frac{logit(hm\%)+logit(sm\%)}{2})$, where

$logit(x) = log(\frac{x}{100\%-x})$ and $logit^{-1}(x) = 100\% \times \frac{e^x}{1+e^x}$. We held fixed $\delta$=3, and varied $\sigma_{min}$, $\theta_{min}$, and $\lambda$ parameters. We compared the effectiveness of our approach against the current state-of-the-art spam filters.

The computational efficiency is evaluated through the average number of messages that are classified per second. The experiments were performed on a Linux-based PC with INTEL PENTIUM III 1.0 GHz processor and 1.0 GBytes RAM.

### 4.1 Eager vs. Lazy Classification

We start our analysis by investigating the differences between the proposed eager and lazy classification approaches. In these experiments we held fixed $\theta_{min}$=90% and $\lambda$=1, and varied $\sigma_{min}$ from 5.0% to 0.01%. We used the TREC05P-1/FULL public corpus[1]. The corpus consists of chronologically sequence of 92,189 messages, from which 39,399 are ham and 52,790 are spam. Table 1 shows the results used to compare both approaches.

As we can see, for higher values of $\sigma_{min}$ the eager approach generates only few rules on average, hurting classification effectiveness. Although only few rules are generated, the total number of patterns that are examined is actually huge. However, the fraction of patterns that ganerate applicable rules is small, representing a wastage of computational resources, and, as a result, the eager approach is not able to classify more than 73 messages per second. The lazy approach, on the other hand, is able to generate more rules by focusing on the useful portion of the training data. Since only the useful portion of the training data is used, the patterns that are generated are more likely to generate applicable rules. In fact, we observed that the actual number of patterns that are generated by the lazy approach is not so far from the number of rules generated, and consequently, the lazy approach is able to classify much more messages per second than the eager one. Further, the lazy approach also shows to be better in terms of classification effectiveness. This is due to the fact that the lazy approach generates more rules, possibly covering more cases than the eager approach.

As we lower the value of $\sigma_{min}$ we observed that both eager and lazy approaches generate more rules. Also, the superiority of the lazy approach, specially in terms of computational complexity, becomes more evident for lower values of $\sigma_{min}$. The lazy approach continues generating more applicable rules than the eager approach, and thus it continues to be more effective.

[1] http://plg.waterloo.ca/~ gvcormac/treccorpus

### 4.2 Overall Results

We continue the analisys of the lazy approach by performing experiments varying $\theta_{min}$ and $\lambda$. In addition to TREC05P-1/FULL, we also used more four corpora in these experiments, as shown in table 2.

| Corpus | #Ham | #Spam | #Messages |
|---|---|---|---|
| TREC05P-1/FULL | 39,399 | 52,790 | 92,189 |
| TREC05P-1/HAM25 | 9,751 | 52,790 | 62,541 |
| TREC05P-1/HAM50 | 19,586 | 52,790 | 72,376 |
| TREC05P-1/SPAM25 | 39,399 | 13,179 | 52,578 |
| TREC05P-1/SPAM50 | 39,399 | 26,283 | 65,682 |

**Table 2. Corpus Statistics.**

Figure 6 shows how misclassification percentages evolve as a function of messages processed. We set $\sigma_{min}$=0.05%, $\theta_{min}$=90% and $\lambda$=1.50. It is important to notice that there is no abrupt variation of spam misclassification percentages. This means that even with the changing nature of spams, our approach is able to quickly learn their patterns. Some corpora show a big variation in ham misclassification percentage. We believe this is because there are a relatively large number of spam messages in the corresponding portion of each corpus, and the majority of the frequent patterns refers to spam messages. Consequently, much more patterns will vote for spam while only few will vote for ham.

Table 3 shows misclassification percentages as a function of $\lambda$. For this experiment we set $\sigma_{min}$=0.05% and $\theta_{min}$=90%. As expected, for $\lambda$=1.00, ham misclassification approaches spam misclassification. When $\lambda$ is increased to 10.00, few messages were classified as spam, and consequently, ham misclassification decreases at the expense of spam misclassification. The best results were obtained by applying $\lambda$=1.50.

| $\lambda$ | hm% | sm% | lam% |
|---|---|---|---|
| 1.00 | 1.01 | 1.96 | 1.41 |
| 1.50 | 0.40 | 1.61 | **0.80** |
| 2.00 | 0.35 | 2.18 | 0.88 |
| 10.0 | 0.25 | 7.95 | 1.45 |

**Table 3. Misclassification for Different Cost Factors.**

Figure 7 shows misclassification percentages as a function of $\theta_{min}$. We set $\sigma_{min}$=0.05% and $\lambda$=1.50. As we can see $\theta_{min}$ represents a trade-off between the number of strong rules and the quality of the rules. Lowering $\theta_{min}$ the number of strong rules increases, but their quality decreases. If $\theta_{min}$ is set too low, several rules will participate in the poll, but a large number of those rules will possibly

| $\sigma_{min}$ | Avg. Number of Applicable Rules | | Avg. Messages/second | | hm% | | sm% | | lam% | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Eager | Lazy | Eager | Lazy | Eager | Lazy | Eager | Lazy | Eager | Lazy |
| 5.0% | 3 | 17 | 73 | 128 | 12.6 | 3.2 | 14.7 | 5.1 | 13.6 | 4.3 |
| 2.0% | 9 | 37 | 58 | 109 | 9.0 | 2.2 | 10.1 | 3.2 | 9.5 | 2.6 |
| 1.0% | 23 | 62 | 43 | 91 | 6.4 | 0.9 | 8.1 | 1.9 | 7.2 | 1.3 |
| 0.5% | 113 | 241 | 37 | 79 | 2.7 | 0.4 | 3.7 | 1.2 | 3.2 | **0.7** |
| 0.1% | 147 | 262 | 22 | 72 | 1.6 | 0.4 | 2.7 | 1.2 | 2.1 | 0.7 |
| 0.01% | 209 | 315 | 8 | 50 | 1.0 | 0.4 | 2.1 | 1.5 | **1.45** | 0.8 |

**Table 1. Comparison between Eager and Lazy Approaches.**

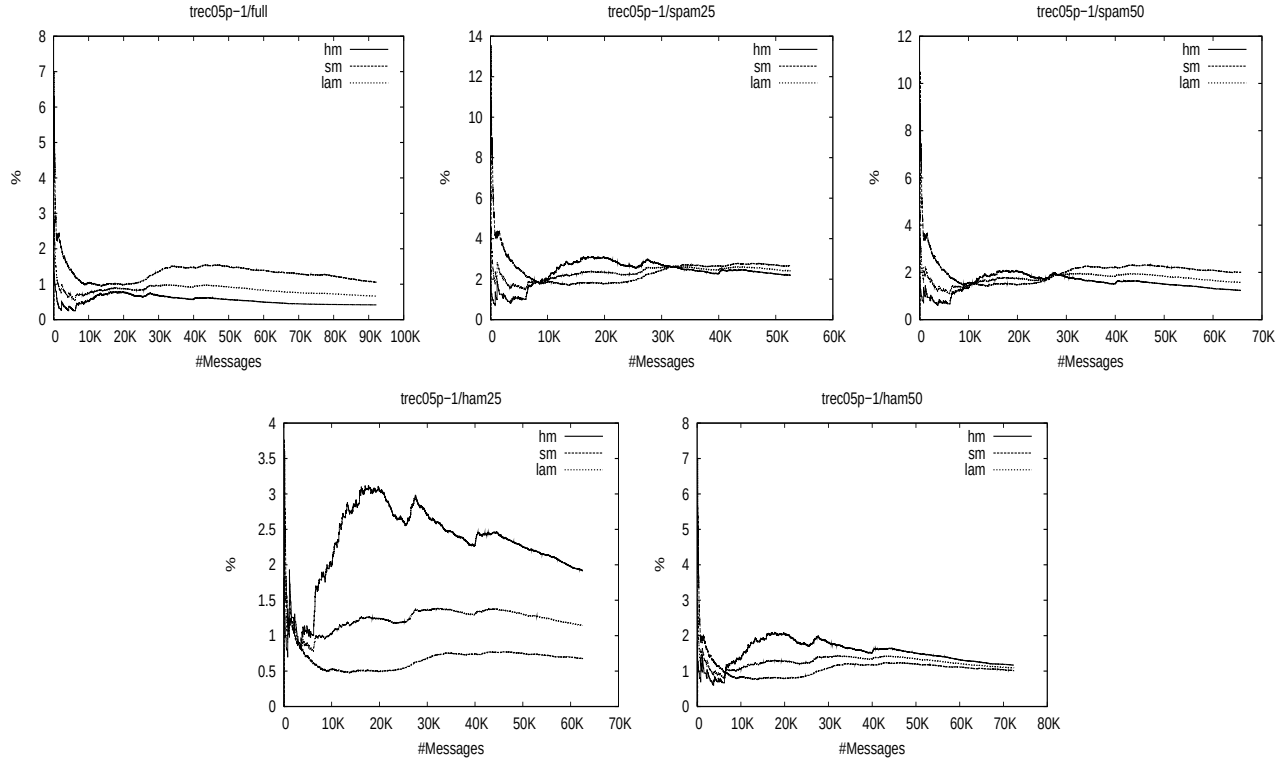


**Figure 6. Learning Curve.**

vote for the wrong class. On the other hand, if $\theta_{min}$ is set too high, there will be only few high-confident rules in the poll. We observed that $\theta_{min}$=90% is the best configuration for most of the corpora (except for TRECP05-I/SPAM25 corpus).

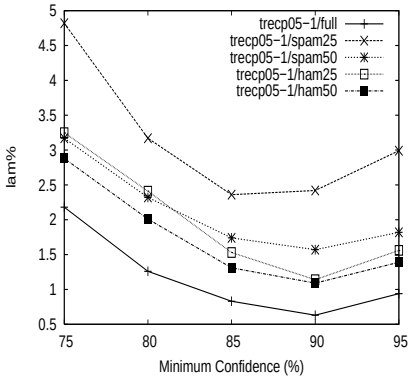


**Figure 7. Misclassification for Different Minimum Confidence Values.**

### 4.3 Comparison against other Filters

We compared our lazy approach, LAZYFILTER, against the well-known spam filters BOGOFILTER [14], SPAMASSASSIN [1], SPAMBAYES [13], and SPAMPROBE [5]. We held fixed $\sigma_{min}$=0.5%, $\theta_{min}$=95%, $\delta$=3, and $\lambda$=1.50.The publicly available SPAMASSASSIN corpus was used in this experiment. It consists of 6,034 messages – 4,019 ham and 1,885 spam – gathered from various sources at various times. Although it is not a chronological sequence of messages delivered to a single recipient, the messages contain original headers with minor elision for the sake of privacy. We used the SPAMASSASSIN corpus and performed ten-fold cross validation to compare the results obtained from our lazy classification approach with the results obtained from other filters. The results are showed in Table 4.

| Filter | hm% | sm% | om% |
|---|---|---|---|
| BOGOFILTER | **0.12** | 9.7 | 3.5 |
| SPAMASSASSIN | 0.14 | 3.9 | 1.3 |
| SPAMBAYES | 0.17 | 4.4 | 1.5 |
| SPAMPROBE | 0.14 | 3.4 | 1.2 |
| LAZYFILTER | 0.15 | **2.8** | **1.1** |

**Table 4. Comparison against other Filters.**

BOGOFILTER is the best performer in terms of ham misclassification, classifying only 0.12% of the innocent messages as spam messages. However, BOGOFILTER is the worst perform in terms of spam misclassification, classifying almost 10% of the spam messages as ham. LAZYFILTER is competitive to other filters, achieving 0.15% of ham misclassification, but the actual advantage of LAZYFILTER is the lowest spam misclassification. This also reflects in the overall misclassification, which is near 1.1%.

## 5 Related Work

Content-based filters have been the focus of considerable interest, with work on nearest neighbor classifiers [17], decision trees [6] and Bayesian classifiers [3, 16]. Several state-of-the-art machine learning classification approaches, such as support vector machines and random forests were also applied to the problem of email spam detection [10,15]. The Bayesian approach has been used by many spam filters, including SPAMASSASSIN [1], SPAMPROBE [5], and SPAMBAYES [13].

Data Mining techniques were already applied to the spam detection problem [11]. In [9] spam detection was modeled as a game between spammers and filters, and an adversarial classification approach is then proposed. The spam detection problem was also explored in the KDDCUP 2004 [7].

Our proposed approach is the first one to adopt associative classification for spam detection. Not only words and symbols are used to classify a message, but also combinations of them. Further, our approach employes a lazy rule induction, which naturally deals with the problem of rare patterns, and thus, novel and emerging patterns are quickly incorporated to the classification model. Our lazy rule induction approach differs from the kNN approach in several ways. For instance, our approach performs dimensionality reduction while the kNN approach reduces the number of points by selecting the k nearest ones.

## 6 Conclusions

In this paper we propose and evaluate a novel spam detection method, which introduces innovations in the way patterns are discovered and used for sake of classification. First, we propose two associative classifiers, one that is eager in the sense that it induce rules based on all known messages, and another that is lazy in the sense that only the useful messages are used to induce the rules. Then we evaluate both approaches and empirically concluded that the lazy approach is superior, both in terms of classification effectiveness as well as in computational complexity. Our lazy approach was compared against state-of-the-art well-known spam filters. We concluded that our approach, LAZYFILTER, is competitive in terms of classification effectiveness.

As future work, we intend to further evaluate the effectiveness of our approach using other corpora. Further, we

will explore other application scenarios, such as bioinformatics and message categorization.

## References


[1] Spamassassin. *http://spamassassin.apache.org*, 2005.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the Int. Conf. on Management of Data, SIGMOD*, pages 207–216, Washington, USA, May 1993. ACM.

[3] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proc. of the Work. on Machine Learning in the New Information Age*, 2000.

[4] I. Androutsopoulos, E. Magirou, and D. Vassilakis. A game theoretic model of spam e-mailing. In *Proc. of the Conf. on Email and Anti-Spam, CEAS*, Stanford, USA, 2005.

[5] B. Burton. Spamprobe - a fast bayesian spam filter. *http://spamproble.sourceforge.net*, 2002.

[6] X. Carreras and L. Marquez. Boosting trees for anti-spam email filtering. In *Proc. of the Intl. Conf. on Recent Advances on Natural Language Processing, RANLP*, September 2001.

[7] R. Caruana, T. Joachims, and L. Backstrom. KDD-Cup 2004: results and analysis. *SIGKDD Explor. Newsl.*, 6(2):95–108, 2004.

[8] G. Cormack and T. Lynam. Overview of the TREC 2005 spam evaluation track. In *Proc. of the Text Retrieval Conference, TREC*, Gaithersburg, USA, 2005. NIST.

[9] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining, KDD*, pages 99–108, 2004.

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

[11] T. Fawcet. 'in vivo' spam filtering: A challenge problem for data mining. *KDD Explorations*, 2(5), December 2003.

[12] D. Lowd and C. Meek. Adversarial learning. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining, KDD*, pages 99–108, Chicago, USA, 2005. ACM Press.

[13] T. Meyer and B. Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *Proc. of the Conf. on Email and Anti-Spam, CEAS*, July 2004.

[14] E. Raymond, D. Relson, M. Andree, and G. Louis. Bogofilter. *http://bogofilter.sourceforge.net*, 2004.

[15] G. Rios and H. Zha. Exploring support vector machines and random forests for spam detection. In *Proc. of the Conf. on Email and Anti-Spam, CEAS*, July 2004.

[16] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk email. In *Proc. of the Work. on Learning for Text Categorization*, Madison, USA, 1998.

[17] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6:49–73, 2003.

[18] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, pages 283–290, August 1997.