

# Adaptive Spam Filtering Using Dynamic Feature Space

Yan Zhou  
School of CIS  
Univ. of South Alabama  
Mobile, AL 36688 USA  
zhou@cis.usouthal.edu

Madhuri S. Mulekar  
Dept. of Mathematics & Statistics  
Univ. of South Alabama  
Mobile, AL 36688 USA  
mmulekar@jaguar1.usouthal.edu

Praveen Nerellapalli  
School of CIS  
Univ. of South Alabama  
Mobile, AL 36688 USA  
prn301@jaguar1.usouthal.edu

## Abstract

*Unsolicited bulk e-mail, also known as spam, has been an increasing problem for the e-mail society. This paper presents a new spam filtering strategy that 1) uses a practical entropy coding technique, Huffman coding, to dynamically encode the feature space of e-mail collections over time and, 2) applies an online algorithm to adaptively enhance the learned spam concept as new e-mail data becomes available. The contributions of this work include a highly efficient spam filtering algorithm in which the input space is radically reduced to a single-dimension input vector, and an adaptive learning technique that is robust to vocabulary change, concept drifting and skewed data distribution. We compare our technique to several existing off-line learning techniques including Support Vector Machine, Naïve Bayes, k-Nearest Neighbor, C4.5 decision tree, RBFNetwork, Boosted decision tree and Stacking, and demonstrate the effectiveness of our technique by presenting the experimental results on the e-mail data that is publicly available.*

## 1. Introduction

Unsolicited bulk e-mail or spam has become an increasing problem for the Internet society. According to the International Telecommunication Union (ITU), spam constitutes as much as 85% of all e-mail in year 2004, compared to an estimate of 35% reported a year earlier<sup>1</sup>. As the amount of spam traffic grows, the cost is staggering. According to the UN Trade Conference on Trade and Development, spam could cost the world more than twenty billion dollars annually in wasted technical resources and lost productivity.

The ongoing research and development of spam filtering techniques focuses on applying machine learning algorithms to differentiate spam from legitimate e-mail.

Since most spam is sent as text messages, spam filtering has been simplified as a typical off-line text classification problem. Among the most popular approaches in text learning, bayesian analysis appeared most frequently in the published work on spam filtering [3, 18, 23]. Other attempts include instance-based learning [19], rule-based learning [16], boosted decision trees [5], and support vector machines (SVM) [14]. Although promising results were achieved with these approaches in the context of text classification, the spam problem is yet far from being solved in the real world. In reality, spam presents many challenges that an off-line batch learning problem usually does not face, such as changing vocabulary, skewed class distribution, concept drifting [24], and text distortion [9]. For instance, Hidalgo [11] shows that the percentage of spam varies significantly in different e-mail collections, from 16.6% to 88.2%. Fawcett [9] further points out that changing class distributions and concept drift are often encountered in the operational settings, and a trained classifier may not perform consistently as spam varies. Existing spam filtering techniques lack the ability to adapt over time and thus cannot meet the challenges posed by spam.

In this paper, we propose a novel spam filtering strategy that is highly efficient and able to adapt to new concepts over time. We show that practical encoding techniques such as Huffman coding [12] can be used to create a dynamic feature space representation of the input data. The validity of a message is indicated by its rank value computed based on the Huffman prefix trees, and a classifier is trained to classify messages according to their rank values. Unlike the existing off-line learning techniques, our approach takes into account the dynamic nature of e-mail data. Specifically, we model vocabulary change and concept drift as feature space drift, and update the classifier with the rank values of e-mail across different “ages” sampled with an exponentially-decayed sampling rate. The intuition is, regardless of concept and vocabulary changes, the difference between spam and legitimate mail in terms of rank values should always be greater than a threshold, and more recent

<sup>1</sup><http://www.itu.int/osg/spu/spam/>

messages should play a more important role in classifying present messages. Rank values in earlier ages are retained and sampled for fitting a new concept model, thus there is no need to re-process all the training messages in the past. In contrast, in the case of vocabulary change, concept drift or any other similar changes in the training corpus, existing off-line learning techniques will require re-processing all previous training messages to form a new structured input representation as in the vector space model and the boolean space model.

The rest of the paper is organized as follows. In section 2, we discuss existing techniques that have recently been applied in the area of spam filtering. Section 3 explains in detail the core of our approach: feature space encoding, ranking, and adaptive filtering. The pseudocode of our algorithm is also presented. Section 4 presents the experimental setup and the data used for testing, along with the experimental results comparing various techniques. Section 5 concludes our work and discusses future directions.

## 2. Related Work

Sahami et. al [18] trained a Naïve Bayes classifier to detect junk e-mail. They adopted the *Vector Space* model [22] in which each dimension of the feature space corresponds to a word in the entire training corpus. They reduced the feature space to the top 500 words ranked using mutual information. In addition, they included a set of hand-crafted phrasal features such as “Free Money” and a set of non-textual attributes such as the domain type “.com” that are good indications of junk e-mail. The trained classifier was used to determine the likelihood of unseen e-mail in the spam or legitimate mail category. They concluded that by considering domain-specific features, the efficiency of anti-spam filters can be greatly enhanced. However, in the real e-mail environment, spammers often hide domain-specific features by applying techniques such as text distortion, falsifying return addresses, or using spamming software to hide the true identity of their machines. It has been shown that there is no evidence that the inclusion of phrasal attributes can lead to further benefits [21].

Sakkis et. al [21] proposed a memory-based approach to spam filtering for mailing lists. They extended the basic  $k$ -Nearest Neighbor algorithm by applying different attribute and distance weighting schemes [1, 7]. They claimed that memory-based filters perform better than Naïve Bayesian filters on average, and memory-based filters are practically feasible when combined with the safety nets available in mailing lists. They also introduced cost-sensitive evaluation measures that differentiate the cost of false negative and false positive examples for a learned hypothesis.

The same group also applied a classifier ensemble model or stacked generalization [26] in the context of spam fil-

tering [20]. A classifier ensemble, or committee, consists of a set of ground-level classifiers whose predictions are used to produce an enhanced training set that incorporates the confidence of each individual member into the training examples. The enhanced training set is used to induce a higher-level classifier referred to as the “president” of the committee. The final prediction is made for an unseen example by the president based on the predictions made by the members of the committee. In their experiments, they used a Naïve Bayes classifier and a memory-based classifier as the ground-level classifiers, and the president was another memory-based classifier. They demonstrated that stacking may consistently improve the performance of spam filters. The potential disadvantage of their approach is its computational intensity since multiple classifiers are needed in both training and classification.

Zhang and Yao [28] presented a Maximum Entropy [4] based approach to junk mail filtering. They showed that comparing to Naïve Bayes, the Maximum Entropy method produces comparable or better results, and domain-specific features provided by SpamAssassin<sup>2</sup> can further enhance the performance of the Maximum Entropy approach. The disadvantages of this model include a relatively low training speed and the lack of the ability of incremental learning.

Drucker et. al [6] showed that Support Vector machines (SVM) and boosted decision trees outperform RIPPER and Rocchio in spam classification. They also suggested that a complete feature set should be used rather than a subset since searching for the optimum set of features is impractical. Carreras and Márquez [5] also showed that boosted algorithms are better than the baseline algorithms. Zhan et. al [27] proposed an LVQ-based neural network approach and demonstrated that it is superior to Bayes-based and BP-based approaches in spam filtering.

The major difference between our approach and the existing techniques is that our approach can achieve a high percentage of effectiveness at a much faster speed, and learn classification models that adapt over time as new e-mail messages are made available for training (e.g. through relevance feedback). The idea is to leverage the fact that an adaptive Huffman encoding technique such as FGK [8, 10, 13] can be used to adaptively encode the feature space of the spam and non-spam corpus as Huffman prefix trees, from which a rank value can be defined to differentiate spam from legitimate mail. Now we present our technique.

## 3. Our Adaptive Learning Model

In this section, we discuss in detail our adaptive approach to spam filtering which consists of feature space encoding, ranking, and adaptive learning.

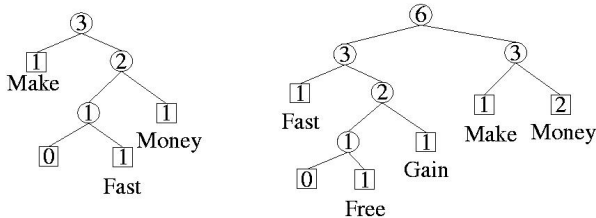
---

<sup>2</sup><http://spamassassin.org>

### 3.1. Feature Space Representation

Huffman Coding is a lossless compression algorithm that generates minimum-redundancy codes for the input symbols. The static version of the algorithm constructs a prefix tree given a list of input weights that represents the probabilities of distinct symbols in the source messages. In the static method, the prefix tree is fixed once it is built, and the symbol frequencies are determined before encoding starts. In our approach, we adopt the word-based adaptive Huffman coding algorithm FGK [8, 10, 13] so that word frequencies and the prefix tree can be updated incrementally in real time. In the adaptive method, every tree has a special node, the 0-node, whose weight is always zero. When a new word occurs in the training messages, the 0-node spawns a new 0-node and a leaf that represents the new word. Each node is assigned a weight which is the sum of the weights of its two children. The weight of a leaf node is the frequency of a distinct word. The tree is (re)computed so that the nodes are arranged bottom-up in order of non-increasing weight. Knuth has proved that the run-time of the algorithm is  $O(\ell)$  where  $\ell$  is the current length of the codeword [13]. Figure 1 shows the Huffman tree computed for message “make money fast” and the adapted tree after the new message “gain free money” is received.

**Figure 1. Huffman trees for message “make money fast” followed by “gain free money.”**



In our spam filtering technique we build two Huffman prefix trees,  $\mathcal{T}_s$  from the spam corpus and  $\mathcal{T}_\ell$  from the legitimate mail corpus. Together they represent the feature space  $\mathcal{F}$  of the present e-mail collection.

### 3.2. Ranking

Given a set of messages  $\mathcal{M} = \mathcal{M}_s \cup \mathcal{M}_\ell$  for training, where  $\mathcal{M}_s$  is the spam and  $\mathcal{M}_\ell$  is the legitimate mail in  $\mathcal{M}$ , we compute Huffman tree  $\mathcal{T}_s$  from  $\mathcal{M}_s$  and  $\mathcal{T}_\ell$  from  $\mathcal{M}_\ell$ . We define the feature space as  $\mathcal{F} = \mathcal{T}_s \cup \mathcal{T}_\ell$ . In essence,  $\mathcal{T}_s$  characterizes the spam corpus, while  $\mathcal{T}_\ell$  characterizes the legitimate corpus in the training set. Next, we transform each tree into a vector as follows:

- for  $i = 1, \dots, N$ , where  $N$  is the total number of words in the tree, let

$$w_i = 1 - \frac{L_i}{H}$$

in which  $L_i$  is the length of the codeword for word  $i$ , and  $H$  is the height of the tree.

- the vector representation of the tree is defined as

$$\vec{w} = (w_1, w_2, \dots, w_N).$$

Thus the vector representation of  $\mathcal{T}_s$  is  $\vec{w}_s = (w_1, w_2, \dots, w_{N_s})$ , where  $N_s$  is the total number of words in  $\mathcal{T}_s$ . Similarly,  $\vec{w}_\ell = (w_1, w_2, \dots, w_{N_\ell})$  for  $\mathcal{T}_\ell$ , where  $N_\ell$  is the total number of words in  $\mathcal{T}_\ell$ .

Given a message  $M$ , we “decompose”  $M$  into two vector representations defined over  $\mathcal{T}_s$  and  $\mathcal{T}_\ell$ , namely  $\vec{p}_s = (p_1, \dots, p_{N_s})$  where  $p_i$  ( $i \in [1, N_s]$ ) is the probability of occurrence of word  $i$  recorded in  $\mathcal{T}_s$ , and  $\vec{p}_\ell = (p_1, \dots, p_{N_\ell})$  where  $p_i$  ( $i \in [1, N_\ell]$ ) is recorded in  $\mathcal{T}_\ell$ . Note that in both definitions,  $p_i$  is zero if word  $i$  contained in the tree does not occur in message  $M$ . On the other hand, words in  $M$  that are not contained in the tree under consideration are ignored when forming the vector representation.

**Definition 1** Given a message  $M$ , let  $r_s$  be the length of the vector  $\vec{p}_s$  projected onto the vector  $\vec{w}_s$ , and  $r_\ell$  be the length of the vector  $\vec{p}_\ell$  projected onto the vector  $\vec{w}_\ell$ . The rank  $R_M$  of message  $M$  is the ratio of  $r_s$  to  $r_\ell$ .

Given  $M$  we can compute  $r_s$  and  $r_\ell$  as follows:

$$r_s = \frac{\vec{p}_s \cdot \vec{w}_s}{\|\vec{w}_s\|}, \quad r_\ell = \frac{\vec{p}_\ell \cdot \vec{w}_\ell}{\|\vec{w}_\ell\|}.$$

Thus,

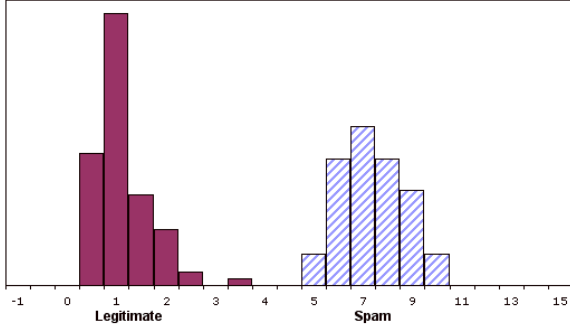
$$R_M = r_s / r_\ell = \left( \frac{\vec{p}_s \cdot \vec{w}_s}{\|\vec{w}_s\|} \right) / \left( \frac{\vec{p}_\ell \cdot \vec{w}_\ell}{\|\vec{w}_\ell\|} \right).$$

By defining the rank of a message, we drastically reduce the dimension of the input space to a single dimension. The rank values can be used to differentiate spam from legitimate mail as shown in Figure 2. The plot shows the distribution of e-mail in a training set containing 50 messages. Notice that there is a clear distinction between spam and legitimate mail in terms of their rank values. Legitimate mail tends to have relatively low rank values compared to spam.

### 3.3. Classification Model

To accurately identify the threshold rank value that differentiates spam from legitimate mail, we apply logistic regression [2, 17, 15] to fit a binary classification model to the rank values of the e-mail in the training set. The learned model can be used to estimate the probability that a message is spam.

**Figure 2. Distribution of 50 training e-mail messages in terms of rank values.**



Consider a simple linear regression model  $Y_M = a + bR_M + \varepsilon_M$  relating the class label with the rank of the message. Here  $Y_M$  is a dichotomous outcome of the  $M^{th}$  message (spam or legitimate),  $R_M$  is the rank of the  $M^{th}$  message provided by the proposed adaptive Huffman coding scheme,  $a$  is the  $Y$ -intercept,  $b$  is the slope, and  $\varepsilon_M$  is the random error. If we assume that  $E(\varepsilon_M) = 0$ , i.e. no random error on the average, then  $E(Y_M) = a + bR_M$ . Let  $P(Y_M = spam) = \pi_M$  be the probability that the  $M^{th}$  message is spam. Since  $Y_M$  is the Bernoulli random variable with the probability distribution,

$Y_M$	Probability
Spam	$P(Y_M = spam) = \pi_M$
Legitimate	$P(Y_M = legitimate) = 1 - \pi_M$

using the definition of the expected value, we get  $E(Y_M) = \pi_M$ . Thus, the expected value of the dichotomous outcome of the  $M^{th}$  message is given by  $E(Y_M) = a + bR_M = \pi_M$ . However, in this situation the errors are non-constant and non-normal. Additionally, the response is a probability,  $\pi_M$ , hence must take values in the interval  $[0,1]$ . These constraints pose a problem in application of a linear model since assumptions of constant and normal errors underlying linear model are not satisfied. Also the constraint of response,  $P(Y_M = spam) = \pi_M \in [0, 1]$ , rules out a linear model. Theoretical considerations suggest that a sigmoidal response function of the form

$$E(Y_M) = \frac{e^{a+bR_M}}{1 + e^{a+bR_M}} \quad (1)$$

fits such a dichotomous data better. For a given sample of  $n$  training messages (spam + legitimate), the logarithm of the likelihood function of the parameters to be estimated ( $a$  and  $b$ ) is given as follows:

$$\log_e L(a, b) = \sum_{M=1}^n Y_M (a+bR_M) - \sum_{M=1}^n \log_e (1 + e^{a+bR_M}).$$

We can obtain maximum likelihood estimates of  $a$  and  $b$  by maximizing this log-likelihood function. Then from Equation 1, we can derive:

$$P(Y_M = spam) = \frac{e^{(a+bR_M)}}{1 + e^{(a+bR_M)}}.$$

To make the filter more efficient, we can simply compute the threshold rank value using Equation 2 so that if the rank value of a message is greater than the threshold, the probability that it is spam is at least  $P_{spam}$ .

$$R_{thresh} = \frac{\log_e \frac{P_{spam}}{1-P_{spam}} - a}{b} \quad (2)$$

Since false positive examples usually have much higher cost than false negative examples in spam filtering, we should pick a high  $P_{spam}$  value, for example 0.99, to compute the threshold. Thus a message is predicted as spam with a probability of 0.99 if its rank value is greater than the threshold  $R_{thresh}$ ; otherwise, the message is classified as legitimate.

Note that logistic regression is not the only way to learn the classification model. Once the rank values are produced for the training data, almost any classification algorithm will work from this point on. We demonstrate this important fact in the Experimental Results section.

### 3.4. Adaptive Learning Model

The dynamic representation of feature space alone is not sufficient to make the learning model adaptive. To learn a spam filtering model that not only adapts to the new concepts over time, but also retains, to a certain extent, the memory of the concepts learned earlier, we periodically recompute the classification model with a procedure that approximates exponential aging. As a result, training messages received at time  $t$  contribute more to classifying e-mail received at  $t + 1$  than training messages received at  $t - 1$  do.

Let  $\mathcal{X} = \{X_1, \dots, X_t\}$  be a set of training sets available at time  $t$ , in which  $X_i$  is a set of additional messages supplied for training at time  $i$ , for  $i = 1, \dots, t$ . In a conventional approach, every message in  $\cup_{i=1}^t X_i$  will be used to build a classification model for use at  $t + 1$ . However, in the case of vocabulary change or concept drift, earlier messages could pose a negative impact on the reliability of the classification model. Thus, an adaptive learning model should be able to capture time locality in the provided data. In our approach, we apply a sampling scheme that approximates the idea of exponential aging. Let  $\gamma$  be the sampling rate, at time  $t$  a subset  $X_i'$  of  $X_i$  ( $i = 1, \dots, t$ ) is selected for training, and  $|X_i'| = \gamma^{t-i} \cdot |X_i|$ . Thus the classification model for time  $t + 1$  is built on  $\cup_{i=1}^t X_i'$ .

In our approach, we fit a classification model to the rank values of the training messages. Even though the feature

space representation may have been changed at time  $t$ , we do not update the rank values of the messages received before  $t$ . In other words, if a message  $x \in X_i$  ( $i \leq t$ ) is selected for training at time  $t$ , we use its rank value computed at time  $i$ . This way we do not have to keep and re-process earlier messages. Table 1 gives the detailed pseudo-code of our adaptive spam filtering algorithm.

**Table 1. The pseudo-code of our algorithm.**

<p>Algorithm: Adaptive-Spam-Filtering  Initialize <math>t \leftarrow 1</math>  Repeat    Let <math>X_i</math> be a set of new training data supplied at time <math>i \in [1, t]</math>    Let <math>\mathcal{M}_s</math> be the set of spam in <math>X_t</math>    Let <math>\mathcal{M}_\ell</math> be the set of legitimate mail in <math>X_t</math>    Let <math>\mathcal{U}_t</math> be the test set at time <math>t</math>    Compute Huffman tree <math>\mathcal{T}_s</math> with <math>\mathcal{M}_s</math>    Compute Huffman tree <math>\mathcal{T}_\ell</math> with <math>\mathcal{M}_\ell</math>    <b>For</b> each <math>M \in \mathcal{M}_s \cup \mathcal{M}_\ell</math>      <math>\star d_i</math> is the <math>i</math>th word in <math>M \star</math>      <math display="block">r_s(M) = \frac{\sum_{d_i \in \mathcal{T}_s} p_s(d_i) \times (1 - \frac{L(\mathcal{T}_s, d_i)}{H(\mathcal{T}_s)})}{\sum_{d_i \in \mathcal{T}_s} (1 - \frac{L(\mathcal{T}_s, d_i)}{H(\mathcal{T}_s)})}</math>     <math display="block">r_\ell(M) = \frac{\sum_{d_i \in \mathcal{T}_\ell} p_\ell(d_i) \times (1 - \frac{L(\mathcal{T}_\ell, d_i)}{H(\mathcal{T}_\ell)})}{\sum_{d_i \in \mathcal{T}_\ell} (1 - \frac{L(\mathcal{T}_\ell, d_i)}{H(\mathcal{T}_\ell)})}</math>     <math>R_M = r_s(M)/r_\ell(M)</math>      <math>\star X'_i</math> is sampled from <math>X_i</math> at time <math>t \star</math>      <math>\mathcal{X} = \emptyset</math>      <b>For</b> <math>i = 1</math> to <math>t</math>        Randomly select <math>X'_i \in X_i</math> S.T. <math> X'_i  = \gamma^{t-i} \cdot  X_i </math>        <math>\mathcal{X} = \mathcal{X} \cup X'_i</math>      <math>a, b \leftarrow \text{LogisticRegression}(\bigcup_{M \in \mathcal{X}} R_M)</math>      <math>\star</math> Choose high <math>P_{spam}</math> to avoid false positives <math>\star</math>      <math>R_{thresh} = (\log_e(P_{spam}/(1 - P_{spam})) - a)/b</math>      <b>For</b> each message <math>M</math> in <math>\mathcal{U}_t</math>        Compute <math>R_M</math>        <b>If</b> <math>R_M &gt; R_{thresh}</math>        <b>Return</b> Spam        <b>Else</b>        <b>Return</b> Legitimate      <math>t \leftarrow t + 1</math></p>
--

## 4. Experimental Results

We compared our adaptive learning model to several existing off-line learning techniques, including Support Vector Machine, Naïve Bayes,  $k$ -Nearest Neighbor, C4.5 deci-

sion tree, RBFNetwork, Boosted decision tree and Stacking, on the Ling-Spam corpus used by Sakkis et. al [19]. The corpus consists of 481 spam messages and 2412 legitimate messages sent to the Linguist list. All messages have been preprocessed so that HTML tags, attachments and duplicated messages are removed. Additionally, stemming and stop-list were applied.

We created the vector space model for the e-mail data  $\mathcal{D}$  for the off-line learning algorithms. The  $j^{th}$  message in the data set is represented as a sparse vector  $(w_{1j}, \dots, w_{dj})$  in a  $d$ -dimensional space for all  $j \in [1, |\mathcal{D}|]$ , whereby  $d$  is the number of distinct terms remaining after the preprocessing. Terms are weighted using the TFIDF weighting scheme. Let  $f_{ij}$  be the frequency of term  $i$  in message  $j$ , term frequency  $tf_{ij} = f_{ij}/\max\{f_{ij}\}$  is the normalized  $f_{ij}$ , where  $\max\{f_{ij}\}$  is the maximum term frequency across the entire corpus. Let  $df_i$  be the number of messages containing term  $i$ , and  $idf_i = \log_2(|\mathcal{D}|/df_i)$ . Then,

$$w_{ij} = tf_{ij} \times idf_i = tf_{ij} \times \log_2(|\mathcal{D}|/df_i)$$

We used the implementation of the selected off-line algorithms provided in Weka [25]. For  $k$ -NN we used  $k = 5$ , and for stacking we used Naïve Bayes and C4.5 decision tree as the ground-level classifiers and a Naïve Bayes classifier as the president.

### 4.1. Comparing Various Techniques

The Ling-Spam corpus has been divided into ten parts. We randomly selected a set  $\mathcal{U}_i$  of 100 messages from each part  $i \in [1, 10]$  to form the test set  $\mathcal{U} = \bigcup_{i=1}^{10} \mathcal{U}_i$ , with approximately 16.7% spam and 83.3% legitimate mail as in the original corpus. From the rest of the messages in each part  $i \in [1, 10]$ , we randomly selected a set  $\mathcal{L}_i$  of 50 messages. We set the training set at time  $t \in [1, 10]$  be  $\bigcup_{j=1}^t \mathcal{L}_j$ , thus the size of the training set varies from 50 to 500, incremented by 50 at each time  $t$ . For each training size, we conducted ten experiments on different random training samples and tested the learned model on the 1000 messages in the test set  $\mathcal{U}$ . The results were averaged over the ten independent runs.

Table 2 shows the success rate averaged over ten independent runs on the test set as the size of the training set increases from 50 to 500. We selected a sampling rate  $\gamma = 0.5$  for our adaptive algorithm in this experiment. We also tried different  $\gamma$  values. Due to space restriction, we do not report the rest of the results here. The general observation is that a relatively low sampling rate improves performance when many new messages are supplied over time, while a high sampling rate achieves better success rate when a small number of additional messages are added for training.

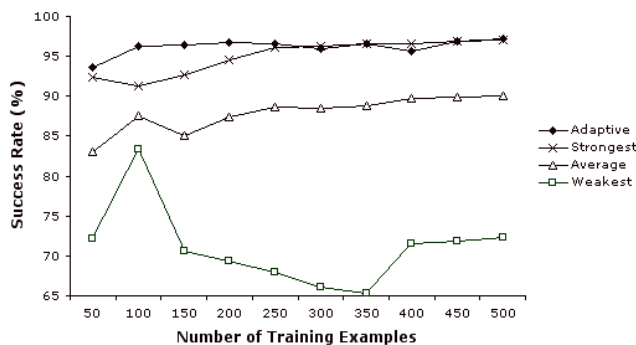
In Figure 3 we show the average success rate of our adaptive algorithm, the best performance, the average per-

**Table 2. The success rate (%) of various algorithms on the test set  $U$ . The highest success rate in each row is bolded. The first column is the size of the training set.  $\gamma$  is the sampling rate.**

Size	NB	SVM	5-NN	C4.5	RBF	Boosted Tree	Stacking	Adaptive ( $\gamma = 0.5$ )
50	92.4	86.35	83.1	78.74	72.14	82.77	85.88	<b>93.55</b>
100	91.33	90.2	83.41	85.61	83.74	90.45	88.24	<b>96.34</b>
150	90.71	92.7	70.55	82.58	83.55	91.28	83.79	<b>96.5</b>
200	91.81	94.62	69.33	88.03	85.33	93.44	89.06	<b>96.81</b>
250	91.84	96.14	67.9	90.85	86.68	95.46	91.49	<b>96.56</b>
300	92.13	<b>96.2</b>	66.11	91.58	85.85	95.88	91.46	96.02
350	92.12	96.56	65.33	92.38	87.11	96.45	91.58	<b>96.58</b>
400	92.12	<b>96.61</b>	71.46	92.61	87.4	96.42	91.77	95.59
450	92.06	96.86	71.8	92.94	87.74	96.37	91.72	<b>96.95</b>
500	91.8	97.06	72.25	93.1	88.23	96.67	91.23	<b>97.17</b>

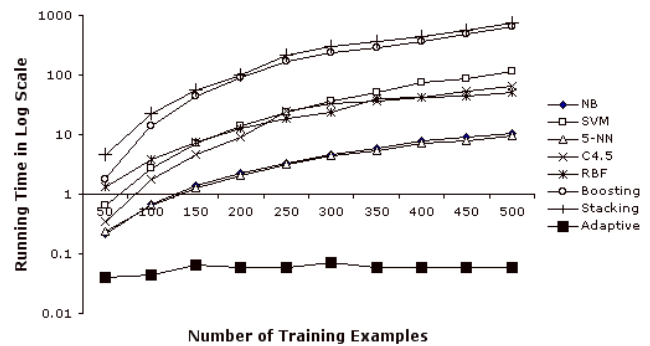
formance and the worst performance of the selected off-line algorithms at each training size. Obviously, our algorithm consistently achieves a high percentage of effectiveness throughout the selected training sizes. Note that the best performance of the off-line algorithms shown in the plot is not produced by the same algorithm. None of the selected off-line algorithms can perform as consistently as our algorithm does and still achieve a high success rate. The best of the selected off-line algorithms is the support vector machine algorithm, which performs comparably to our algorithm after the size of the training set has increased to 250. However, our algorithm runs approximately 667 times faster than the support vector machine algorithm on average. Figure 4 shows the running time (in seconds) of all algorithms on logarithmic scale.

**Figure 3. The average success rates.**



As we mentioned in earlier sections, once the rank values of the messages are computed, any learning algorithm can be used to learn a classification model. Our experimental results show that, using our dynamic feature space representation, ranking and exponential aging schemes, the performance of most of the off-line algorithms can be largely

**Figure 4. The run-time comparison.**



improved. Table 3 shows the increase in average success rate of each of the selected off-line learning algorithms using our dynamic feature model instead of the vector space model. Notice that all the selected algorithms except the support vector machine (SVM) algorithm achieved much better performance in most cases in our adaptive model. We believe that the reason SVM performs worse in our adaptive model is that it tries to produce a model that generalizes well, while our adaptive model is biased towards recent messages to capture the time locality.

We also provide the results with a combined measure for precision and recall, called the F-Measure [25].

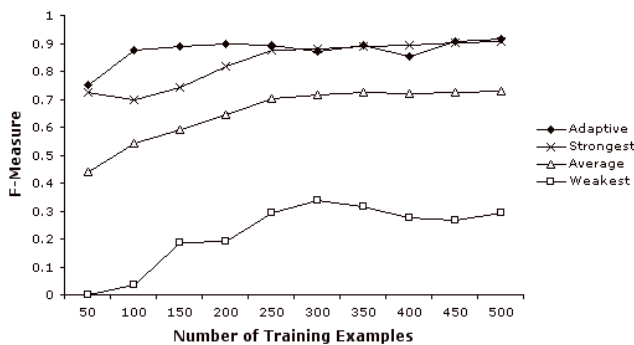
$$F\text{-Measure} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Figure 5 shows the F-Measure results of our adaptive algorithm, the best, average, and worst performance of the selected off-line algorithms. Once again, we found our adaptive algorithm performs more consistently and is more effective in general.

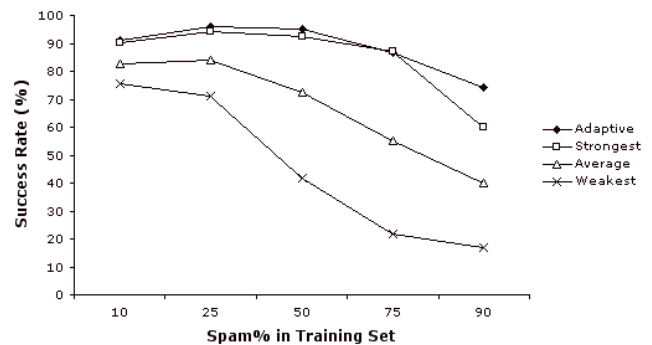
**Table 3. The increase in success rate (%) of the selected off-line algorithms when our adaptive model is used. The first column is the size of the training set. The sampling rate  $\gamma$  is 0.5.**

Size	NB	SVM	5-NN	C4.5	RBF	Boosted Tree	Stacking
50	3.00	0.95	8.54	13.46	22.45	9.43	8.66
100	5.43	-1.77	12.49	10.45	13.43	5.67	8.23
150	5.38	-1.77	25.82	13.04	12.57	4.22	12.67
200	5.59	-3.36	27.48	9.18	11.78	3.83	8.38
250	-0.01	-7.84	28.87	5.36	9.41	0.33	5.2
300	0.91	-7.48	30.45	4.81	9.8	0.15	5.57
350	4.22	-4.12	31.51	4.16	9.67	-0.21	5.42
400	4.88	-4.54	24.42	3.14	7.93	-1.02	4.59
450	3.07	-4.25	24.89	4.03	8.05	0.59	4.89
500	4.92	-3.17	24.77	3.82	8.51	-0.04	5.68

**Figure 5. The average F-Measure.**



**Figure 6. Results with skewed distributions.**



## 4.2. Investigation of Skewed Distributions

To investigate the impact of skewed spam distributions on a classification model, we randomly selected 50 messages for training and tested the learned models on the same test set  $\mathcal{U}$  containing 1000 randomly selected messages. The percentage of spam in the training set varied from 10% to 90%. Table 4 shows the success rates on the test set averaged over ten runs. The success rate varied significantly as the percentage of spam in the training set increased from 10% to 90%. The highest success rate in each row is bolded.

Figure 6 shows the average success rates of our adaptive algorithm, the best, average, and worst performance of the selected off-line algorithms as the spam distribution in the training set increases from 10% to 90%. Clearly our adaptive algorithm is more robust to skewed spam distributions and performs more consistently as the percentage of spam in the training set varies.

## 5. Concluding Remarks

We have presented a highly efficient and effective adaptive spam filtering algorithm. We demonstrated that adap-

tive Huffman coding algorithm can be used to create a dynamic feature space representation for the input data. Our algorithm runs at a much faster speed mainly due to two factors. First, our ranking scheme radically reduces the input domain to a single-dimension vector, thus greatly speeds up the learning process. In addition, training messages are randomly sampled with a procedure that approximates exponential aging. Therefore the size of the training set is significantly reduced. Due to its fast running speed, our algorithm can be easily applied in real time. Since we do not need to update the representation of earlier messages, there is no need to store and re-process those messages as new messages are available. We also demonstrated that our algorithm is more robust to challenges often faced in the operational settings, such as vocabulary change, concept drift, and skewed distributions.

In addition to the problems discussed in this paper, e-mail spam also has an adversarial nature that needs to be taken into consideration. In the future, we plan to develop an adaptive strategy that handles adversarial attacks. We also plan to apply our adaptive learning algorithm to other fields such as text classification and information retrieval.

**Table 4. The success rate (%) of various algorithms on the test set  $U$  with skewed spam distributions in the training set of size 50. The first column is the percentage of spam in the training set.**

Spam%	NB	SVM	5-NN	C4.5	RBF	Boosted Tree	Stacking	Adaptive Filter
10	90.27	85.03	83.1	77.36	75.56	83.54	84.05	<b>90.96</b>
25	94.37	90.77	82.22	77.24	71.32	84.02	88.35	<b>95.96</b>
50	92.31	79.62	44.07	77.95	41.67	80.96	90.98	<b>94.9</b>
75	80.27	30.68	21.7	70.05	26.44	69.63	<b>87.26</b>	86.74
90	60.06	18.67	16.9	55.87	19.23	50.16	58.03	<b>74.27</b>

## References

- [1] W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.
- [2] J. Anderson and S. Richardson. Logistic discrimination and bias correction in maximum likelihood estimation. *Technometrics*, 21:71–78, 1979.
- [3] I. Androutopoulos, J. Koutsias, K. Chandrinou, and C. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of SIGIR-2000*, pages 160–167. ACM, 2000.
- [4] A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.
- [5] X. Carreras and L. Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001.
- [6] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.
- [7] A. Dudani. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(4):325–327, 1976.
- [8] N. Faller. An adaptive system for data compression. In *Record of the 7th Asilomar Conference on Circuits, Systems and Computers*, pages 593–597, 1973.
- [9] T. Fawcett. “in vivo” spam filtering: A challenge problem for data mining. *KDD Explorations*, 2:140–148, 2003.
- [10] R. Gallager. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, 24(6):668–674, November 1978.
- [11] J. Gómez Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *Proceedings of SAC-02, 17th ACM Symposium on Applied Computing*, pages 615–620. Madrid, ES, 2002.
- [12] D. Huffman. A method for the construction of minimum-redundancy codes. *Proc. Inst. Radio Engineers*, 40(9):1098–1101, September 1952.
- [13] D. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6(2):163–180, 1985.
- [14] A. Kolcz and J. Alsepector. SVM-based filtering of e-mail spam with content-specific misclassification costs. In *Proceedings of the TextDM’01 Workshop on Text Mining*, 2001.
- [15] J. Neter, M. Kutner, C. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models 4th Ed.* Irwin, USA, 1996.
- [16] J. Provost. Naive-bayes vs. rule-learning in classification of email. Technical report, University of Texas at Austin, 1999.
- [17] F. Ramsey and D. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis.* Duxbury, CA, USA, 2002.
- [18] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin*, 1998.
- [19] G. Sakkis, I. Androutopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering. Technical report, National Center for Scientific Research “Demokritos”, 2001.
- [20] G. Sakkis, I. Androutopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, pages 44–50, 2001.
- [21] G. Sakkis, I. Androutopoulos, 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.
- [22] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.
- [23] K. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.
- [24] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden context. *Machine Learning*, 23(1):69–101, 1996.
- [25] I. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations.* Morgan Kaufmann, San Francisco, CA, USA, 2000.
- [26] D. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.
- [27] C. Zhan, X. Lu, M. Hou, and X. Zhou. A lqv-based neural network anti-spam e-mail approach. *ACM SIGOPS Operating Systems Review*, 39(1):34–39, 2005.
- [28] L. Zhang and T. Yao. Filtering junk mail with a maximum entropy model. In *Proceedings of the 20th International Conference on Computer Processing of Oriental Languages*, pages 446–453, 2003.