ACM Transactions on Asian Language Information Processing, Vol. 3, No. 4, December 2004, Pages 243-269.

An Evaluation of Statistical Spam Filtering Techniques

Le Zhang, Jingbo Zhu, Tianshun Yao
Natural Language Processing Laboratory
Institute of Computer Software & Theory
Northeastern University, China
ejoy@xinhuanet.com, {zhujingbo, tsyao}@mail.neu.edu.cn
http://www.nlplab.cn/

Abstract

This paper evaluates five supervised learning methods in the context of statistical spam filtering. We study the impact of different feature pruning methods and feature set sizes on each learner's performance using cost-sensitive measures. It is observed that the significance of feature selection varies greatly from classifier to classifier. In particular, we found Support Vector Machine, AdaBoost and Maximum Entropy Model are top performers in this evaluation, sharing similar characteristics: not sensitive to feature selection strategy, easily scalable to very high feature dimension and good performances across different datasets. In contrast, Naive Bayes, a commonly used classifier in spam filtering, is found to be sensitive to feature selection methods on small feature set, and fail to function well in scenarios where false positives are penalized heavily. The experiments also suggest that aggressive feature pruning should be avoided when building filters to be used in applications where legitimate mails are assigned a cost much higher than spams (such as $\lambda = 999$), so as to maintain a better-than-baseline performance. An interesting finding is the effect of mail headers on spam filtering, which is often ignored in previous studies. Experiments show that classifiers using features from message header alone can achieve comparable or better performance than filters utilizing body features only. This suggests that message headers can be reliable and powerfully discriminative feature sources for spam filtering.

1 Background

The task of spam filtering is to rule out unsolicited e-mails automatically from a user's mail stream. Spam mail, also called unsolicited bulk e-mail or junk mail, is internet mail that is sent to a group of recipients who have not requested it. These unsolicited mails have already caused many problems such as filling mailboxes, engulfing important personal mail, wasting network bandwidth, consuming users' time and energy to sort through it, not to mention all the other problems associated with spam (crashed mail-servers, pornography adverts sent to children, and so on). According to a series of surveys conducted by CAUBE.AU [1], the number of total spams received by 41 email addresses has increased by a factor of 6 in two years (from 1753 spams in 2000 to 10,847 spams in 2001). Therefore it is challenging to develop spam filters that can effectively eliminate the increasing volumes of unwanted mails automatically before they enter a user's mailbox.

In its simple form, spam filtering can be recast as Text Categorization task where the classes to be predicted are *spam* and *legitimate*. A variety of supervised machine-learning algorithms have been successfully applied to mail filtering task. A non-exhaustive list includes: Naive Bayes classifier [Sahami et al., 1998, Androutsopoulos et al., 2000a, Schneider, 2003], RIPPER rule induction algorithm [Cohen, 1996], Support Vector Machines [Drucker et al., 1999, Kolcz and Alspector, 2001], Memory Based Learning [Androutsopoulos et al., 2000b], AdaBoost [Carreras and Márquez, 2001] and Maximum Entropy Model [Zhang and Yao, 2003]. While all these approaches seem appealing, it is difficult to make a fair comparison of their performances based on published results. For one thing, the results reported in previous works were often based on different corpora, making cross-classifier evaluation difficult. For another, previous literature usually reported results obtained with optimal parameters tuned on a single corpus. It is not known whether the same setting could yield comparable results under different configurations. For example, [Androutsopoulos et al., 2000a] reported that Naive Bayes exhibits good performance on small feature set (< 300 features), but it is unclear whether Naive Bayes classifier still function well in the presence of tens of thousands of features. Several attempts have been made to evaluate the performance of machine-learning methods on general text categorization task where there are ten or more categories [Yang and Pedersen, 1997, Yang and Liu, 1999]. However,

[1]More information can be found at: http://www.caube.org.au/spamstats.html.

whether the same conclusion still holds on two-class spam filtering task remains to be an open question, especially when user defined cost-sensitive measures are used.

This paper will attempt to bridge this gap, by giving experimental comparisons of common supervised learning algorithms through a series of spam filtering experiments. More specifically, in this paper we seek answers to the following questions:

- How factors such as feature-pruning methods and feature space sizes affect the filtering accuracy of each learner with cost-sensitive measures in mind.

  The unpruned feature space consists of tens of thousands of features, and not all classifiers can operate well in the presence of many noisy features. Feature pruning is frequently employed to reduce the size of feature space, making the computation of certain learners tractable. How these factors affect the classifiers' performance under cost-sensitive measures will be an important issue to be considered when delivering statistical spam filtering techniques to real-world applications.

- Whether the same filtering techniques successfully applied to English can still yield comparable results in other language settings or not.

  The "bag of features" filtering model is known to work well on filtering English spams. To verify the language independence of this model, we conducted several experiments on a newly compiled Chinese spam corpus.

- To what extent the information in different parts of a mail can improve filtering accuracy.

  While most researches on anti-spam filtering focus on building "pure" content based filter that uses features from message body only, relatively few attempts have been made to investigate the discriminative power of message headers. As will be shown in this paper, message headers can be reliable sources of discriminative features for spam filtering.

The rest of this paper is structured as follows, Section 2 provides a high-level overview of the general machine-learning framework for spam filtering. Section 3 outlines the major characteristics of each learning method investigated in this evaluation. Detail information on four benchmarking corpora is presented in Section 4, including a newly compiled Chinese corpus. This is followed in Section 5 with the description of the experimental setup, how the dataset was chosen, what feature selection strategies were used, and the analysis of experimental results. We review related researches in Section 6 and formulate our conclusion in Section 7.

2 Spam Filtering as Text Categorization

Spam filtering is a real-world application of Automated Text Categorization (ATC) task [Sebastiani, 2002], a field that has undergone intensive research in recent years. The early approaches to TC were to manually construct document classifiers with rules compiled by domain experts, which is appropriate when few machine-readable texts were available and the computational power was expensive. Recent trends in the TC community have shifted to building classifiers automatically by applying some machine-learning algorithms to a set of pre-classified documents (training data). This is also called the *statistical* approach, in the sense that differences among documents are usually expressed statistically as the likelihood of certain events, rather than some heuristic rules written by human. This trend is reflected in the goal of statistical spam filtering, which aims at building effective spam filters automatically from email corpus. We begin with a formal definition of automated spam filtering task, followed by a discussion on feature pruning methods and performance measures used in this work.

2.1 Definition

Follow the description used in [Sebastiani, 2002], we now give a definition of Automated Spam Filtering.

Given message set $\mathcal{D} = \{d_1, d_2, \ldots, d_j, \ldots, d_{|\mathcal{D}|}\}$ and category set $\mathcal{C}=\{c_1 = \texttt{spam}, c_2 = \texttt{legitimate}\}$, where d_j is the jth mail in $\mathcal{D}$ and $\mathcal{C}$ is the possible label set. The task of Automated Spam Filtering is to build a boolean categorization function $\Phi(d_j, c_i) : \mathcal{D} \times \mathcal{C} \to \{\texttt{True}, \texttt{False}\}$. When $\Phi(d_j, c_i)$ is `True`, it indicates message d_j belongs to category c_i; when $\Phi(d_j, c_i)$ is `False`, it means d_j does not belong to c_i.

In the setting of spam filtering there exist only two category labels: `spam` and `legitimate`. Each message $d_j \in \mathcal{D}$ can only be assigned to one of them, but not both. Therefore we can use a simplified categorization function $\Phi_{\texttt{spam}}(d_j) : \mathcal{D} \to \{\texttt{True}, \texttt{False}\}$ instead. A message is classified as spam when $\Phi_{\texttt{spam}}(d_j)$ is `True`, and legitimate otherwise.

Using the above notation, applying supervised machine-learning algorithms to spam filtering consists of two stages: the training stage and the classification stage:

Table 1: Summary of different term selection methods

Function	Denoted by	Mathematical form
Document Frequency	$DF(t_k)$	$\mid \{ \vec{d_j} \mid \vec{d_j} \in \mathcal{D}$ and t_k occurs in $\vec{d_j} \} \mid$
Information Gain	$IG(t_k, c_i)$	$IG(t_k) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} Pr(t,c) \cdot \log \frac{Pr(t,c)}{Pr(t) \cdot Pr(c)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\mid\mathcal{D}\mid \cdot [Pr(t_k,c_i) \cdot Pr(\bar{t}_k,\bar{c}_i) - Pr(t_k,\bar{c}_i) \cdot Pr(\bar{t}_k,c_i)]^2}{Pr(t_k) \cdot Pr(\bar{t}_k) \cdot Pr(c_i) \cdot Pr(\bar{c}_i)}$

- During Training
  A set of labeled messages must be provided as training data, which are first transformed into a representation that can be understood by the learning algorithms. The most commonly used representation for Spam Filtering is Vector Space Model (VSM), in which each document $d_j \in \mathcal{D}$ is transformed into a real vector $\vec{d_j} \in \mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ is the vocabulary (feature set), and the coordinates of $\vec{d_j}$ represent the weight of each feature in $\mathcal{V}$. Then we can run a learning algorithm over the training data to create a classifier $\Phi_{\texttt{spam}}(\vec{d_j}) \rightarrow \{\texttt{True}, \texttt{False}\}$.

- During Classification
  The classifier $\Phi_{\texttt{spam}}(\vec{d_j})$ is applied to the vector representation of a new document d to produce a prediction whether d is spam or not.

2.2 Feature Dimension Reduction

The original feature space transformed with the Vector Space Model may contain tens of thousands of different features, and not all classifiers can handle such a high dimension gracefully. Dimension reduction (also called feature pruning or feature selection) is usually employed to reduce the size of the feature space to an acceptable level, typically several orders of magnitude smaller than the original one. The benefit of dimension reduction also includes a small improvement in prediction accuracy in some cases [Yang and Pedersen, 1997].

[Yang and Pedersen, 1997] compared the effectiveness of several feature pruning methods in the context of general Text Categorization task. Their experiment showed that Document Frequency (DF), Information Gain (IG) and χ^2 statistics (CHI) have a clear advantage over two other commonly used feature selection methods: Mutual Information (MI) and Term Strength (TS) in TC task. We therefore decide to restrict our attention to DF, IG, and CHI in this work.

Let t denote the feature (term) of interest, and c is the category label (either `spam` or `legitimate`). Then $Pr(t), Pr(c)$, and $Pr(t \wedge c)$ are the probabilities of t, c, and both t and c occur in training set $\mathcal{D}$. The mathematical forms of the three feature selection methods are summarized in Table 1.

It is worth mentioning that Information Gain is just the weighted average of the Mutual Information between (t, c) and $(\bar{t}, c)$, and is also called the *average mutual information* [Fano, 1961]. IG has been used in several anti-spam experiments under the name of "Mutual Information" [Sahami et al., 1998, Androutsopoulos et al., 2000c, Schneider, 2003]. We shall use the name IG in the rest of this paper to avoid possible confliction with Mutual Information, which is defined to be:

$$MI(t,c) = \log \frac{Pr(t \wedge c)}{Pr(t) \cdot Pr(c)} \tag{1}$$

2.3 Performance Measures

We now introduce the performance measures used in this paper. Let S and L stand for spam and legitimate message respectively. $n_{L \rightarrow L}, n_{S \rightarrow S}$ denote the numbers of legitimate and spam messages correctly classified by the system. $n_{L \rightarrow S}$ represents the number of legitimate messages misclassified as spam (false positive), and $n_{S \rightarrow L}$ is the number of spam messages wrongly treated as legitimate (false negative). Then spam precision(p), spam recall(r) and F_1-measure are defined as follows:

$$Precision(p) = \frac{n_{S \rightarrow S}}{n_{S \rightarrow S} + n_{L \rightarrow S}} \tag{2}$$

$$Recall(r) = \frac{n_{S \rightarrow S}}{n_{S \rightarrow S} + n_{S \rightarrow L}} \tag{3}$$

$$F_1(p,r) = \frac{2pr}{p+r} \tag{4}$$

When filtering spam, it is worth noting that misclassifying a legitimate mail as spam is much more severe than letting a spam message pass the filter. Letting a spam go through the filter generally does no harm while misblocking an important personal mail as spam can be a real disaster. The usual precision/recall/F_1 measures tell little about a filter's performance when false positive and false negative are weighted differently. To introduce some cost-sensitive evaluation measures that assign false positive a higher cost than false negative, a Weighted Accuracy (WAcc) measure specially tailored for this scenario can be used. WAcc was introduced by [Androutsopoulos et al., 2000b] and has been used in several spam filtering benchmarks [Androutsopoulos et al., 2000b, Carreras and Márquez, 2001]. WAcc is defined as:

$$WAcc_{\lambda} = \frac{\lambda \cdot n_{L \to L} + n_{S \to S}}{\lambda \cdot N_L + N_S} \tag{5}$$

where N_L is the total number of legitimate messages and N_S denotes the total number of spams. WAcc treats each legitimate message as if it were λ messages: when false positive occurs, it is counted as λ errors; and when it is classified correctly, this counts as λ successes. The higher λ is, the more cost is penalized on false positives.

[Androutsopoulos et al., 2000b] also introduced three different values of λ: $\lambda = 1, 9,$ and 999. When λ is set to 1, spam and legitimate mails are weighted equally; when λ is set to 9, a false positive is penalized nine times more than a false negative; for the setting of $\lambda = 999$, more penalties are put on false positive: misblocking a legitimate mail is as bad as letting 999 spam messages pass the filter. Such a high value of λ is suitable for scenarios where messages marked as spam are deleted directly.

In practice, when λ is assigned a high value (such as $\lambda = 999$), WAcc can be so high that it tend to be easily misinterpreted. To avoid this problem, it is better to compare the weighted accuracy and error rate to a simplistic baseline. As suggested in [Androutsopoulos et al., 2000c], we use the case where no filter is present as baseline: legitimate messages are never blocked and spams can always pass the filter. Then the baseline versions of weighted accuracy and weighted error rate are:

$$WAcc^b = \frac{\lambda \cdot N_L}{\lambda \cdot N_L + N_S},\ WErr^b = \frac{N_S}{\lambda \cdot N_L + N_S} \tag{6}$$

To allow easy comparison with the baseline, [Androutsopoulos et al., 2000c] introduces the Total Cost Ratio (TCR) as a single measurement of the spam filtering effects:

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda \cdot n_{L \to S} + n_{S \to L}} \tag{7}$$

Here greater TCR values indicate better performance. If TCR is less than 1.0, then the baseline (not using the filter) is better. An effective spam filter should be able to achieve a TCR value higher than 1.0 in order to be useful in real world applications.

3 Methods Evaluated

In this section, we will give a brief description to each of the five supervised learning methods investigated in this work. We begin with two probabilistic classifiers: Naive Bayes and Maximum Entropy Model. They are probabilistic in the sense that they are able to estimate the probability of each category being predicted. Then we proceed with a memory-based approach that labels a new message according to its similarity between stored instance base. After that, we introduce Support Vector Machine, a learning paradigm that is based on Structured Risk Minimization principle [Vapnik, 1995, Cortes and Vapnik, 1995]. Finally, we discuss AdaBoost [Freund and Schapire, 1996], a relatively new framework for boosting a weak learner into a strong one. These particular techniques are chosen because of their excellent performance reported in previous studies.

3.1 Naive Bayes

Naive Bayes (NB) is a widely used classifier in Text Categorization task [Lewis, 1998, McCallum and Nigam, 1998]. It also enjoys a blaze of popularity in anti-spam researches [Sahami et al., 1998, Pantel and Lin, 1998, Androutsopoulos et al., 2000a, Schneider, 2003], and often serves as baseline method for comparison with other approaches.

Given a feature vector $\vec{d_j} = \{w_{j1}, w_{j2}, \ldots, w_{j|\vec{d_j}|}\}$ of a message, where w_{ji} is the weight of feature i, and let c denote the category to be predicted ($c \in \{$`spam, legitimate`$\}$), by Bayes law the probability that $\vec{d_j}$ belongs to c is:

$$P(c \mid \vec{d_j}) = \frac{P(c) \cdot P(\vec{d_j} \mid c)}{P(\vec{d_j})} \tag{8}$$

where $P(\vec{d_j})$ is the probability that a randomly picked document has vector $\vec{d_j}$ as its representation, $P(c)$ is the prior probability of a randomly picked document with label c and $P(\vec{d_j} \mid c)$ denotes the probability of a random picked document with label c has $\vec{d_j}$ as its representation.

The denominator $P(\vec{d_j})$ is the same for all categories under consideration and can be dropped safely. The parameter $P(c)$ can be estimated from training data through relative frequency. However, it is usually infeasible to compute the term $P(\vec{d_j} \mid c)$ directly. Since the number of possible vectors $\vec{d_j}$ is too high. In order to alleviate this problem it is common to make the assumption that any two coordinates of the document vector are, when viewed as random variables, statistically independent of each other. So $P(\vec{d_j} \mid c)$ can be decomposed to:

$$P(\vec{d_j} \mid c) = \prod_{i=1}^{|\mathcal{V}|} P(w_{ji} \mid c) \tag{9}$$

This is where the name "Naive" comes from. Despite the fact that this kind of assumption is often violated in real world data, Naive Bayes classifier performs fairly well on spam filtering task [Sahami et al., 1998, Androutsopoulos et al., 2000a]. In our implementation of Naive Bayes we also use the Laplacean prior to smooth the estimate of $P(w_{ji} \mid c)$, as suggested in [McCallum and Nigam, 1998].

Once the classifier is built, we can classify a message $\vec{d_j}$ to be spam if $P(\text{spam} \mid \vec{d_j})$ exceeds a predefined threshold t. In our experiment where mis-classifying a legitimate message is λ times more costly than wrongly labeling a spam as legitimate, Naive Bayes classifier is expected to achieve an optimal result by setting $t = \lambda/(1 + \lambda)$, as long as the probability estimates are accurate [Lewis, 1995].

3.2 Maximum Entropy Model

Maximum Entropy (ME or MaxEnt) Model [Berger et al., 1996] is a general-purpose machine-learning framework that has been successfully applied to a wide range of text processing tasks such as Statistical Language Modeling [Rosenfeld, 1996], Language Ambiguity Resolution [Ratnaparkhi, 1998] and Text Categorization [Nigam et al., 1999].

Given a set of training samples $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ where x_i is a real value feature vector and y_i is the target class, the maximum entropy principle [Jaynes, 1983] states that we should summarize data $\mathcal{T}$ with a model that is maximally noncommittal with respect to missing information. Among distributions consistent with the constraints imposed by $\mathcal{T}$, there exists a unique model with highest entropy in the class of exponential models of the form:

$$P_\Lambda(y \mid x) = \frac{1}{Z_\Lambda(x)} \exp\left[\sum_{i=1}^{n} \lambda_i f_i(x, y)\right] \tag{10}$$

where $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ are parameters of the model, $f_i(x, y)$'s are arbitrary feature functions the modeler chooses to model, and $Z_\Lambda(x) = \sum_y \exp\left[\sum_{i=1}^{n} \lambda_i f_i(x, y)\right]$ is the normalization factor to ensure $P_\Lambda(y \mid x)$ is a probability distribution. Moreover, it has been shown that the Maximum Entropy model is also the Maximum Likelihood solution on the training data that minimizes the Kullback-Leibler divergence between P_Λ and the uniform model [Della Pietra et al., 1997].

Since the log-likelihood of $P_\Lambda(y \mid x)$ on training data is convex in the model's parameter space Λ, a unique Maximum Entropy solution is guaranteed and can be found by maximizing the log-likelihood function:

$$L_\Lambda = \sum_{x,y} \tilde{p}(x, y) \log p_\Lambda(y \mid x) \tag{11}$$

where $\tilde{p}(x, y)$ are empirical probability distribution. In practice, the parameter Λ can be computed through numerical optimization method. In our experiment, we use the Limited-Memory Variable Metric method [Liu and Nocedal, 1989], a Limited-memory version of *quasi-newton* method (also called L-BFGS) to find Λ. Applying L-BFGS requires evaluating the gradient of object function L_Λ in each iteration, which can be computed as:

$$\frac{\partial L}{\partial \lambda_i} = E_{\tilde{p}} f_i - E_p f_i \tag{12}$$

here $E_{\tilde{p}f_i}$ and E_{pf_i} denote the expectation of f_i with respect to the empirical distribution $\tilde{p}$, and model p, respectively.

The advantage of ME model is the ability to freely incorporate features from diverse sources into a single, well-grounded statistical model. [Zhang and Yao, 2003] applied Maximum Entropy Model to a junk filtering task and reported ME model outperforms a baseline Naive Bayes classifier. In this evaluation, a publicly available ME toolkit from the first author was used.[2]

[2]Availability at: http://www.nlplab.cn/zhangle/maxent_toolkit.html.

3.3 Memory Based Learning

Memory-based Learning (also called instance based learning) is a non-parametric inductive learning paradigm that stores training instances in a memory structure on which predictions of new instances are based. The approach is based on the assumption that reasoning is based on direct reuse of stored experiences rather than on the application of knowledge (such as rules or decision trees) abstracted from experience. The similarity between the new instance X and example Y in memory is computed using a *distance metric* $\delta(X, Y)$. In our experiment, we used IB1-IG, a k-NN classifier that uses an Information Gain (IG) weighted overlap metric, which is defined as:

$$\triangle(X, Y) = \sum_{i=1}^{n} w_i \delta(x_i, y_i) \tag{13}$$

where w_i is the Information Gain of feature i, and $\delta(x_i, y_i)$ is defined to be:

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases} \tag{14}$$

Memory-based Learning has been applied to spam filtering task with comparable results to a Naive Bayes classifier [Androutsopoulos et al., 2000b]. The memory-based learner used in our work is TiMBL [Daelemans et al., 1999], a software package developed by the ILK Research Group, Tilburg University. TiMBL is a collection of memory-based learners that sit on top of the classic k-NN classification kernel with added metrics, algorithms, and extra functions.

3.4 Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised learning paradigm based on the Structured Risk Minimization principle from computational learning theory [Vapnik, 1995, Cortes and Vapnik, 1995].

Given a set of training data $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where $x_i \in \mathbb{R}^n$ and $y \in \{+1, -1\}$ (here $+1$ denotes spam and -1 stands for legitimate mail). Training a support vector machine equals to finding the solution of the following optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} W^T \cdot W + C \sum_{i=1}^{N} \xi_i \tag{15}$$

$$\text{subjected to } y_i(W^T \phi(x_i) + b) \geq 1 - \xi_i, \ \xi_i \geq 0 \tag{16}$$

here training vectors x_i's are mapped into a higher (maybe infinite) dimensional space by the function ϕ. W is a weight vector that should be minimized in finding an optimal linear separating hyperplane in this higher dimensional space. ξ_i's are slack variables and are used together with constant $C \geq 0$ to find solution of (15) in non-separable cases.

SVM has been reported remarkable performance on Text Categorization task with many relevant features [Joachims, 1998b]. It has also been applied to spam filtering task with excellent filtering accuracy [Kolcz and Alspector, 2001, Drucker et al., 1999]. In our evaluation, we used an off-the-shelf SVM implementation: SVM^{light} [Joachims, 1998a] to build SVM models. The output of SVM is the signed distance between the instance to be classified and the classification hyperplane. It is possible to normalize the real-value output from SVM into values in [0, 1] via sigmoid transformation [Platt, 1999], so that the result can be evaluated in terms of cost-sensitive measures. We tried the sigmoid transformation but found the normalized output fail to work well when setting threshold with $t = \lambda/(1 + \lambda)$. Therefore, in our experiment we took the approach of adjusting spam threshold θ directly on the output of SVM through cross-validation. When an instance has a distance greater than θ it is labeled as a spam, otherwise it is treated as a legitimate mail. Two different thresholds θ_9 and θ_{999} are used for $\lambda = 9$ and 999, respectively.

3.5 Boosting

AdaBoost [Freund and Schapire, 1996] is a relatively new learning framework for constructing a highly accurate classification rule by combining many weak hypotheses, each of which may be only moderate accurate. The AdaBoost algorithm can be described as follows:

Given: training examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where x_i is a real value vector, $y_i \in \{+1, -1\}$ (+1 denotes spam and -1 denotes legitimate); the number of training rounds T.

Initialize: $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to [-1, +1]$ and its error ϵ_t.
- Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{17}$$

  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution) that should be minimized in each round of training.

Output the final hypothesis:

$$H(x) = \operatorname{sign}(f(x)) = \operatorname{sign} \left[\sum_{t=1}^{T} \alpha_t h_t(x) \right]. \tag{18}$$

here D_t is the distribution at round t, $h_t(x) : X \to \mathbb{R}$ is the weak rule acquired according to D_t.

The AdaBoost algorithm boosts the accuracy of a weak learner by efficiently simulating the weak learner on multiple distributions over the instance space and taking the majority vote of the resulting output hypotheses. The algorithm maintains a set of weights over the training samples and updates these weights in each round of training. Here the weight-updating rule is chosen so that the weight of samples that are misclassified are increased, and the weights of correctly classified samples are decreased. This effectively makes the base learner concentrate on the "harder" samples as training goes on. [Carreras and Márquez, 2001] used AdaBoost algorithm to induce classifiers from decision trees and reported good result on PU1 corpus (described in next section).

There are four different versions of AdaBoost algorithms introduced in [Schapire and Singer, 2000], among which the real AdaBoost.MH algorithm proved to be the most effective one on Text Categorization task. In our experiment, we implement a version of real AdaBoost.MH learner that boosts a weak learner of decision stump. A decision stump is a one level decision tree of the form:

$$h(x) = \begin{cases} c_0 & \text{if } p \text{ holds in } x \\ c_1 & \text{otherwise} \end{cases} \tag{19}$$

where c_0 and c_1 are real numbers, p is the predicate of the tree's root node. In each round of training, a stump (p, c_0, c_1) is built so that $Z_t = \sum_{i=1}^{m} D_t(i) \exp\left[-\alpha_t y_i h_t(x_i)\right]$ is minimized. The reader is referred to [Schapire and Singer, 2000] for details on AdaBoost.MH algorithm. Like SVM, the final decision is made by evaluating the real output of AdaBoost.MH against a threshold θ chosen empirically through cross-validation. Two thresholds θ_9 and θ_{999} are used for different λs.

4 Testing Corpora

Unlike general Text Categorization task where many standard benchmark collections exist, relatively few spam corpora are available, and the sizes are often small. This is probably because while it is easy to collect spam messages (from internet resources like http://spamarchieve.org), it is much harder to collect legitimate mails for the reason of protecting personal privacy. A common approach to compile spam corpus is to mix legitimate mails from one source with spam messages collected from another place. This may lead to a bias when training the classifier: the corpus compiled may not reflect the true distribution of spam mails and tend to be easily separable. A better way may be collecting both legitimate and spam messages from the same source (for instance, a publicly available mailing list that is heavily spammed). To make a thorough evaluation, four publicly available spam corpora are used in this work. Table 2 summarizes the basic properties of each corpus.[3]

[3]When counting vocabulary size, we treat the same word occurs in mail subject and body part as two distinct words.

Table 2: Summary of four spam corpora used in this evaluation

Corpus	No. of spam	No. of legitimate	Spam Rate	Vocabulary Size
PU1	481	618	43.77%	24748
Ling	481	2412	16.63%	65723
SA	1897	4150	31.37%	134850
ZH1	1205	428	73.79%	43134

- **PU1 Corpus**
  The PU1 [4] corpus consists of 1099 messages, 481 of which are marked as spam and 618 are labeled as legitimate, with a spam rate of 43.77%. The messages in PU1 corpus have header fields and html tags removed, leaving only subject line and mail body text. To address privacy, each token was mapped to a unique integer. The corpus comes in four versions: with or without stemming and with or without stop word removal.

- **Ling-spam Corpus**
  The Ling-spam corpus was collected by the same author of PU1 corpus, which includes: 2412 legitimate messages from a linguistic mailing list and 481 spam messages collected by the author with a 16.63% spam rate. Like PU1 corpus, four versions are available with header fields, html tags, and attachments removed.

  Since the Ling-spam corpus was compiled from different sources: the legitimate messages came from a spam-free, topic-specific mailing list, and the spam mails were collected from a personal mailbox, the mail distribution is less like that of the normal user's mail stream, which may make messages in Ling-spam corpus easily separable.

- **SpamAssassin Corpus**
  The SpamAssassin (SA) corpus is a larger collection made available by spamassassin.org. [5] SA corpus includes 1897 spam messages and 4150 legitimate messages with a spam rate of 31.37%. All messages come in raw email format. This makes it possible to evaluate the contribution of header fields in classifying spams.

- **ZH1 Chinese Spam Corpus**
  While good spam filtering results have been reported on several English corpora, it is still unclear whether the same technique is effective in other language settings. To explore this point, we compiled a publicly available Chinese spam corpus named ZH1 [6]. The ZH1 corpus is made up of 1205 spam messages and 428 legitimate messages collected by the first author, with a spam rate of 73.79%. The messages in ZH1 corpus are all simplified Chinese text with GB2312/GBK encoding.

  Unlike English where there are explicit boundaries between words, Chinese text is written continuously without word delimitation. To apply the word based spam filtering techniques to Chinese, we segmented the Chinese text into words with a Chinese word segmenter developed by the Natural Language Processing Lab, Northeastern University [7]. All Chinese texts in header fields, message body, sender, and recipient's names are tokenized into words before further processing.

4.1 Corpus Preprocessing

The purpose of corpus preprocessing is to transform messages in mail corpus into a uniform format that can be understood by the learning algorithms. In our experiment we transformed features found in mails into a Vector Space in which we define each dimension of the space as corresponding to a given feature in the entire corpus of messages seen. Each individual message can be represented as a binary vector denoting which features were present or absent in the message. This is frequently referred to as the "bag of words" approach.

Messages in PU1 and Ling-spam corpora have already been parsed and tokenized into individual words with binary attachments and HTML tags removed. The other two corpora SA and ZH1 contain raw messages in a well-formed format: each message has several header fields recording information like the mail server used, the sender and receiver's addresses, the mail agents used and so on, followed by a body part (with several attachments in some cases). In tokenizing the raw messages, we chose to use an attribute-value representation to map tokens into features: each token consists of an attribute field indicating the token's position in the message paired with the token's value. The

[4] The PU1 Corpus and the Ling-spam Corpus are freely available from: http://www.aueb.gr/users/ion/publications.html.

[5] Availability: http://spamassassin.org/publiccorpus. Several versions of the corpus exist on that site. We use the version labeled as 20030228 in this experiment.

[6] ZH1 Corpus is freely available from: http://www.nlplab.cn/zhangle/spam/zh1.tar.bz2.

[7] see http://www.nlplab.cn.

attribute set includes common parts found in mail: subject, body, return-path, received server, from, to, and so on. Under this representation, the word "Free" occurs in message subject line and body text are treated as two distinct tokens: (subject:Free) and (body:Free). Html tags and attachments were removed.

Word stemming and stop-word removal are two other issues need to be considered in tokenizing emails. Word stemming refers to converting words to their morphological base forms, for example, both "clicking" and "clicked" are reduced to root word "click". Stop-word removal is a procedure to remove words that are found in a list of frequently used words like "and, for, a". The main advantages of applying word stemming and stop-word removal are the reduction of feature space dimension and possible improvement on classifiers' prediction accuracy by alleviating the data sparseness problem. [Androutsopoulos et al., 2000c] have investigated the use of word stemming and stop-word list on the performance of a Naive Bayes classifier using PU1 corpus. Their result shows that most of time stemming and stop-word removal do not have a statistically significant improvement over the non-stemming, no-stop-word removal filter. Furthermore, word stemming only applies to certain Latin-like languages, and is nonsensical to Asian languages like Chinese or Japanese. For this reason we only use the non-stemming, no-stop-word removal versions of PU1 and Ling-Spam corpus and do no special treatment on words in SA and ZH1 corpus.

5 Cost-Sensitive Evaluation

5.1 Experiment I: The Impact of Feature Selection Method

The first experiment was designed to reveal the impact of different feature pruning strategies and feature space sizes on the performance of each classifier. Our primary interest is to find whether the three feature selection methods: document frequency (DF), information gain (IG) and χ^2-test (CHI), which were found to be especially effective on general multi-class Text Categorization tasks [Yang and Chute, 1994], perform similarly on two-class spam filtering task with cost-sensitive measures in mind. Beside this goal, we also want to identify the scalability of each learner empirically by varying the feature set sizes.

To this end, we gradually increased the feature set size of each corpus by selecting top N features ranked with DF, IG, and CHI respectively. We then trained and tested each classifier on the same training/testing split using ten-fold cross validation. When it comes to evaluation criteria, we chose to focus on Total Cost Ratio (TCR) with λ set to 9 and 999. Since the main difference between spam filtering and general Text Categorization task is that in classifying spams, the two types of error (mis-classify spam as legitimate and wrongly mark legitimate as spam) should be weighted differently. In general, people are more interested in scenarios where wrongly labeling a legitimate mail (false positive) is penalized much more than letting a spam pass the filter (false negative). The commonly used *precision, recall* and F_1 measures fail to incorporate cost-sensitive penalties in evaluation, thus do not fill the bill here. TCR addresses this issue by providing a single cost-sensitive measure that allows the classifiers' performances to be compared with a baseline easily.

To make a fair comparison, we tried different parameter settings of each learner several times and reported the results under optimal settings. The similarity metric used in TiMBL was IB1-IG, with k nearest neighbor set to 1. For SVM^{light} package, we found default setting (linear kernel) worked well. Switching to other kernels and parameter settings usually did not gain any improvement. The Maximum Entropy Model was trained up to 100 L-BFGS iterations with a global gaussian prior set to 1.0. Each Boost Stumps was built from 100 rounds of boosting. More rounds in training did not lead to a statistically manifest improvement in accuracy but required substantially longer training time.

To save space, we only plot the five classifiers' TCR9 and TCR999 values obtained on SA corpus (Figure 1 to 5). The X-Axis indicates different feature set sizes [8]; the Y-axis is the TCR measure [9]. All the p-values reported in the rest of this paper were obtained with paired single-tailed t test [Dietterich, 1998].

We can make several useful observations when putting the results together (Figures 1 to 5). The first thing we notice on the figures is that the performance of most classifiers (except Timbl) increased as more features were used (similar observations were made on other corpora though not plotted here). This corresponds to the intuition that the more relevant features selected, the more prediction power a classifier can produce. Yet as we will see later, the gain from increasing feature set size varies greatly from classifier to classifier.

From the TCR9 measure of Naive Bayes (Figure 1-a) we find IG performed slightly better than CHI, and CHI worked much better than DF. CHI outperformed DF on 22 attribute set sizes, confirmed with $p < 0.05$. The difference

[8]The feature set sizes were gradually increased from 50 to 500 with a step of 50, then 600 to 1000 with a step of 100, and finally 2000 to 10000 with a step of 1000. Total 24 different sizes.

[9]When computing the TCR value, it's important not to average the TCRs from results of 10-CV experiment. Since some TCR values obtained can be undefined (WErr is zero), and the small value of TCR can introduce bias. We choose to calculate the final TCR as averaged $WErr_b$ divided by averaged $WErr$, as was done in [Androutsopoulos et al., 2000c].

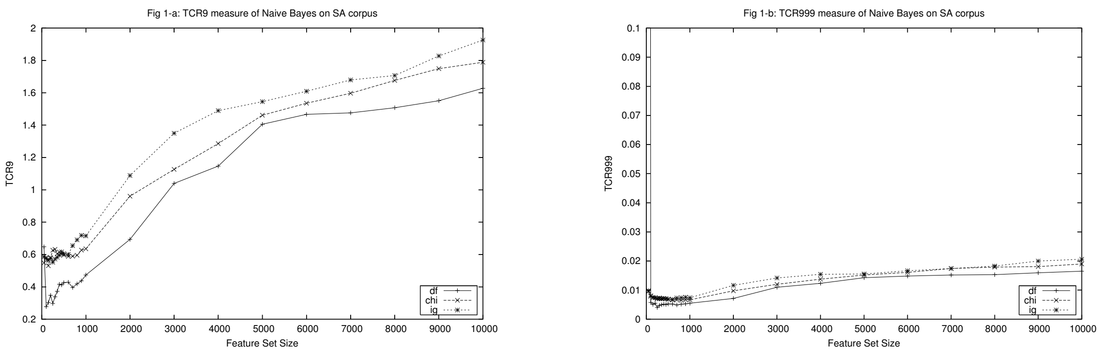


Figure 1: TCR performance curves of Naive Bayes on SA corpus. The threshold t is 0.9 for TCR9 measure (left) and 0.999 for TCR999 measure (right).

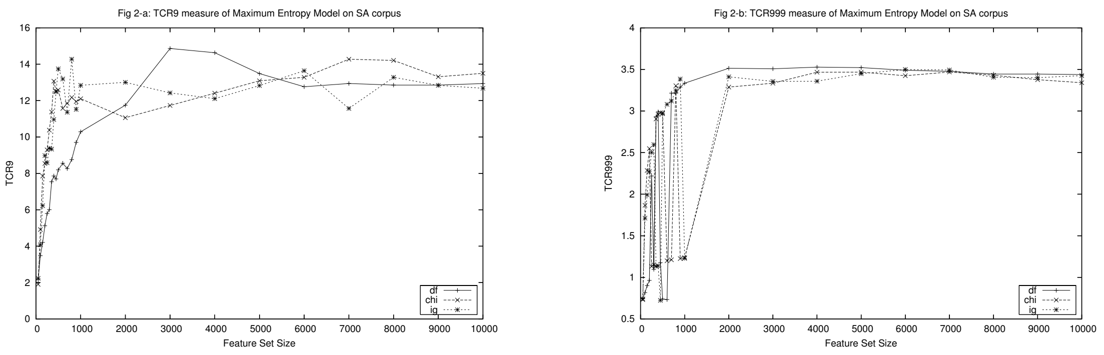


Figure 2: TCR performance curves of Maximum Entropy Model on SA corpus. The threshold t is 0.9 for TCR9 measure (left) and 0.999 for TCR999 measure (right).

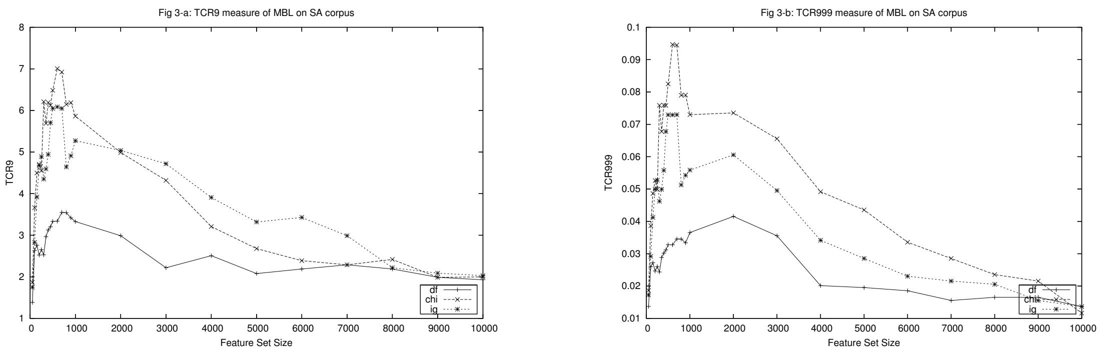


Figure 3: TCR performance curves of Memory Based Learner on SA corpus.

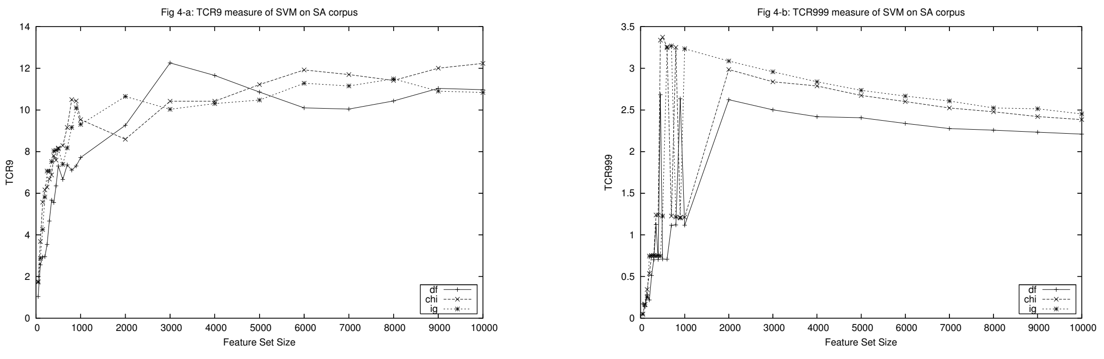


Figure 4: TCR performance curves of SVM^{light} on SA corpus, with spam threshold θ_9 set to 0.48 (TCR9) and θ_{999} set to 1.38 (TCR999).

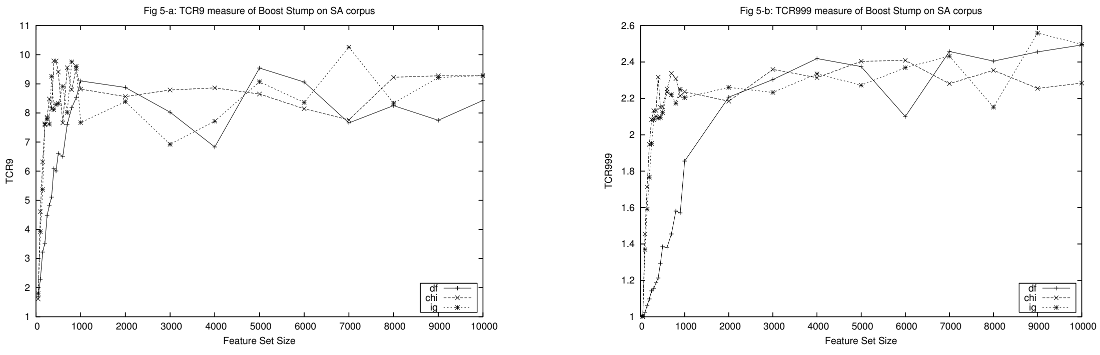


Figure 5: TCR performance curves of Boost Stumps on SA corpus, with spam threshold θ_9 set to 1.95 (TCR9) and θ_{999} set to 6.82 (TCR999).

between DF and IG, CHI is notably big when feature size is small. When less than 1000 features were used, CHI and IG surpassed DF on 15 attribute set sizes at $p < 0.01$. As more features were used, Naive Bayes's accuracy increased gradually in terms of TCR9, obtaining a filtering performance better than baseline (TCR9 > 1.0).

The TCR999 result of Naive Bayes on SA corpus (Figure 1-b) is not exciting. All data points (except X-Axis = 50) fall far below 1.0, and the majority TCR999 values are below 0.02. This implies that in the scenario of $\lambda = 999$ the baseline (no filter) is much better than a Naive Bayes filter is present. Manually checking the output of Naive Bayes shows NB wrongly assigned a small handful of legitimate mails a spam probability very close to 1.0. This can account for the low TCR999 value of NB when false positives are penalized heavily. It is worth noting that Naive Bayes still performed well if we chose to use a small λ, witnessed by a TCR1 value of 10.0 on this corpus (not shown here).

For Maximum Entropy model (Figure 2), all feature selection methods work well on a wide range of feature set sizes. And it seems ME model can benefit from using larger feature sets: the TCR9 values obtained on big feature sets (>2000) are higher than that of smaller ones. DF performed worse than CHI and IG only when less then 300 features were used, confirmed with $p < 0.01$, but worked comparably with CHI and IG on larger feature set. When λ is set to 999 (Figure 2-b), ME was able to maintain a stable TCR999 value well above 1.0 baseline, which indicates ME could be used somewhat safely at this level of λ. The difference between the three feature selection methods is not statistically significant except when less than 300 features were used, where DF performed much worse ($p < 0.01$). We further notice that the TCR999 values of ME model are not stable and fluctuate between 0.8 and 3.0 until more than 1000 features were used.

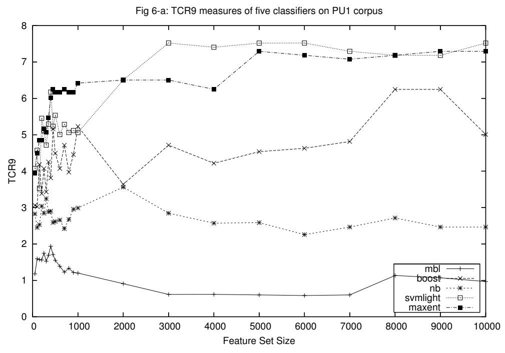
The TCR9 value of the memory-based learner Timbl (Figure 3-a) peaked at 600 features filtered with CHI, achieving a maximum of 7.0. With more features added (>700), Timbl's performance started to degrade. It is observed that CHI is slightly better than IG, which works much better than DF. The difference is notably big when about 500 features were used.

The TCR999 curve of Timbl (Figure 3-b) is rather disappointing: all feature selection methods failed to obtain a TCR999 greater than 0.1. This makes Timbl totally unusable when λ is set to 999. We feel the low TCR score of Timbl can be attributed to the categorical output from the underlying K-NN classifier, which lacks an easy way to adjust the threshold of class labels in accord with different λs [10]. Thus Timbl is difficult to be optimized for cost-sensitive evaluation, despite the fact that Timbl can achieve an F_1 value of 95% on PU1 corpus and an F_1 value of 97.5% on SA corpus (not shown here).

In the case of SVM (Figure 4), good results were achieved on all three feature selection methods with the threshold θ_9 set to 0.48 and θ_{999} set to 1.38 empirically. DF is inferior to CHI and IG on small feature sets (<300), confirmed with $p < 0.01$. However, this difference becomes insignificant as more features were added. When using more than 2000 features, there is no statistically manifest difference between the three feature pruning methods, conforming the early observation of [Joachims, 1998b] that SVM can tolerate many noisy features. The good filtering result achieved by the linear kernel also implies that the spam filtering tasks conducted in this work are mostly linear separable.

Similar to the TCR999 curve of Maximum Entropy Model, the TCR999 values of SVM are fluctuant at small feature sizes, and become stable when more than 2000 features were used. On feature set larger than 700 features, SVM's TCR999 values are constantly above the 1.0 baseline (Figure 4-b). This suggests that it may be viable to use SVM in this scenario safely.

[10] [Androutsopoulos et al., 2000c] circumvented this limitation by adding a post-processing step to Timbl's output which multiplied the number of legitimate neighbors by λ before deciding on the majority class in the neighborhood. Such a post-processing was not used in our experiments with Timbl.



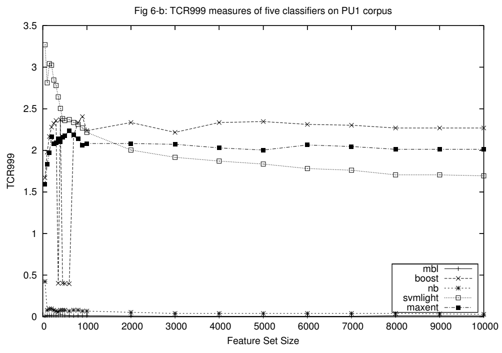


Figure 6: TCR measures of five classifiers on PU1 corpus with features filtered with IG. The spam thresholds are $t_9 = 0.9$ and $t_{999} = 0.999$ for Naive Bayes and Maximum Entropy Model; $\theta_9 = 0.45$ and $\theta_{999} = 1.33$ for SVM^{light}; $\theta_9 = 3.6$ and $\theta_{999} = 8.1$ for AdaBoost.

Looking at the result of AdaBoost (Figure 5) we notice that, despite the weak learner used is quite simple, Boost Stump achieved excellent TCR9 value on SA corpus and good TCR999 value (>2) when more than 2000 features were used (Figure 5-b). Still, DF is found to be inferior than IG and CHI on small feature sets (<1000 features), confirmed with $p < 0.01$.

The good TCR values achieved by Boost Stumps bring up an interesting point on the underlying mechanism of the learner. In each round of boosting, a weak learner (decision stump) is built from training samples by minimizing the normalized factor Z_t. This process can be regarded as a special kind of feature selection, which implies that when feature set is large the Boost algorithm is able to deduce the right feature to use on its own. Figure 5-a and Figure 5-b confirm this hypothesis where all feature selection methods performed equally well on large feature set (the differences among different feature pruning methods are not statistically significant at $p < 0.05$ when more than 2000 features were used).

In sum, we observe that the significance of feature selection varies greatly from classifier to classifier. For all the classifiers we tried, IG and CHI perform much better than DF when feature set sizes are small, usually confirmed with $p < 0.01$. This suggests DF is not a suitable choice for aggressively feature pruning on this task. Although all classifiers are somewhat effective in terms of TCR9 on SA corpus, where λ is assigned to 9, only three classifiers: Maximum Entropy Model, SVM and AdaBoost are found to be effective when λ is set to 999, achieving a TCR999 value greater than the 1.0 baseline on large feature sets. In contrast, Naive Bayes failed to function well when λ is set to 999 because of errors in its probability estimates. Timbl achieved a performance approaches that of Naive Bayes on small feature sets, but failed to scale well in the presence of many noisy features. In addition, we notice that in order to achieve a TCR999 constantly above 1.0 at least 2000 features should be used for the three top-performance classifiers on SA corpus. This suggests what the higher λ is, the more features should be used to allow the classifiers to produce a reliable prediction. Similar patterns are also observed on other corpora (PU1, Ling-Spam and ZH1) though not plotted here, where DF is still inferior to IG and CHI on small feature sets.

5.2 Experiment II: Cross Classifier Evaluation

After investigating the influence of features on classifiers' performance, we now move to the cross classifier evaluation where pairwise comparisons were made on four spam corpora. The training/testing split were prepared as described in Experiment I. We chose to use IG as the single feature pruning method in this experiment, for it has been proved to be an effective feature selection method for this task in Experiment I. Again, we use TCR9 and TCR999 as a unified cost-sensitive measure.

To allow for a rigorous cross-classifier comparison we construct win-tie-loss count tables [11] for all pairs of learning algorithms on the four corpora. For a particular attribute set size (X-Axis) and algorithms A and B, if A's TCR value is greater than B's, and if the difference is statistically significant at the 95% confident level of a paired one-tailed t test, we increase A's win count and B's loss count by one. Otherwise A and B's tie counts will be increased by one. Table 3 and 4 summarize the pairwise win-tie-loss results on all datasets. The TCR9 and TCR999 performance curves are plotted in Figure 6 to Figure 9.

The TCR9 results on PU1 corpus (Figure 6-a) show that Maximum Entropy Model and SVM worked best on this

[11]The table is similar to the one used in [Dietterich, 2000], where pairwise comparisons were made among three methods for constructing ensembles of decision trees.

Table 3: Pairwise combinations of the five classifiers using TCR9 measure on four spam corpora. Each cell contains the number of wins, ties and losses between the algorithm in that row and the algorithm in that column.

PU1 Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-9-15	0-0-24	20-4-0	0-1-23
MaxEnt	17-7-0	0-24-0	24-0-0	
TiMBL	0-0-24	0-0-24		
SVM	13-11-0			

Ling Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-10-14	0-6-18	11-13-0	0-7-17
MaxEnt	4-20-0	0-22-2	24-0-0	
TiMBL	0-0-24	0-0-24		
SVM	4-20-0			

SA Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-0-24	0-0-24	0-3-21	0-0-24
MaxEnt	13-11-0	14-10-0	23-1-0	
TiMBL	0-1-23	0-1-23		
SVM	3-18-3			

ZH1 Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-1-23	0-1-23	9-7-8	0-0-24
MaxEnt	4-19-1	8-8-8	21-3-0	
TiMBL	1-3-20	2-7-15		
SVM	1-14-9			

Table 4: Pairwise combinations of the five classifiers using TCR999 measure on four spam corpora. Each cell contains the number of wins, ties and losses between the algorithm in that row and the algorithm in that column.

PU1 Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-1-23	0-1-23	23-1-0	0-1-23
MaxEnt	0-8-16	8-1-15	24-0-0	
TiMBL	0-0-24	0-0-24		
SVM	7-7-10			

Ling Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-4-20	1-4-19	3-14-7	0-3-21
MaxEnt	0-6-18	1-14-9	24-0-0	
TiMBL	0-0-24	0-3-21		
SVM	12-10-2			

SA Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-0-24	0-0-24	0-0-24	0-0-24
MaxEnt	18-6-0	12-10-2	24-0-0	
TiMBL	0-0-24	0-2-22		
SVM	8-13-3			

ZH1 Corpus	Boost	SVM	TiMBL	MaxEnt
NB	0-0-24	0-1-23	14-10-0	0-0-24
MaxEnt	4-20-0	3-7-14	24-0-0	
TiMBL	0-0-24	0-0-24		
SVM	12-9-3			

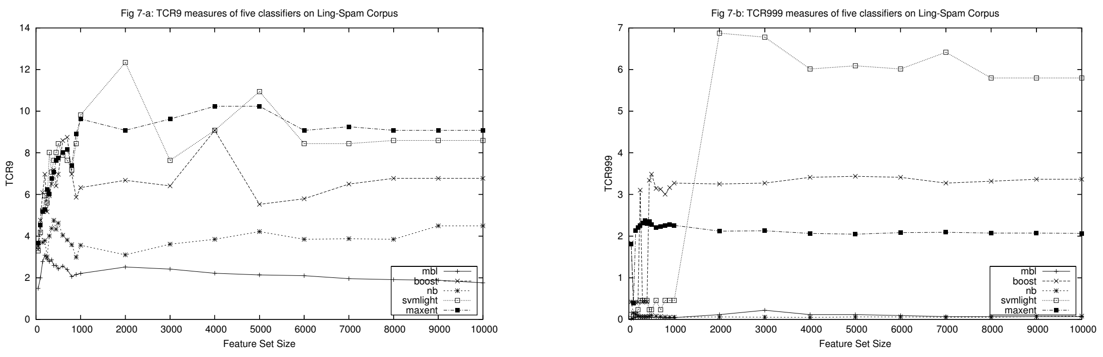


Figure 7: TCR measures of five classifiers on Ling-Spam corpus with features filtered with IG. The spam thresholds are $t_9 = 0.9$ and $t_{999} = 0.999$ for Naive Bayes and Maximum Entropy Model; $\theta_9 = 0.21$ and $\theta_{999} = 0.41$ for SVM^{light}; $\theta_9 = 2.4$ and $\theta_{999} = 7.9$ for AdaBoost.

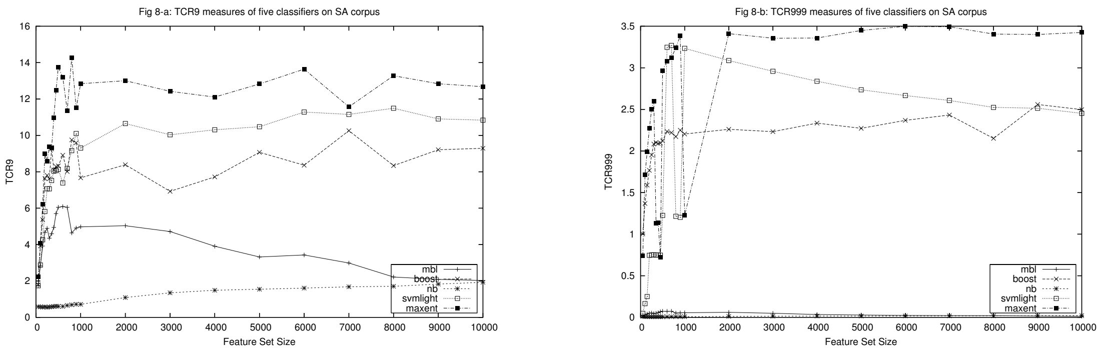


Figure 8: TCR measures of five classifiers on SA corpus with features filtered with IG. The spam thresholds are $t_9 = 0.9$ and $t_{999} = 0.999$ for Naive Bayes and Maximum Entropy Model; $\theta_9 = 0.48$ and $\theta_{999} = 1.38$ for SVM^{light}; $\theta_9 = 1.95$ and $\theta_{999} = 6.82$ for AdaBoost.

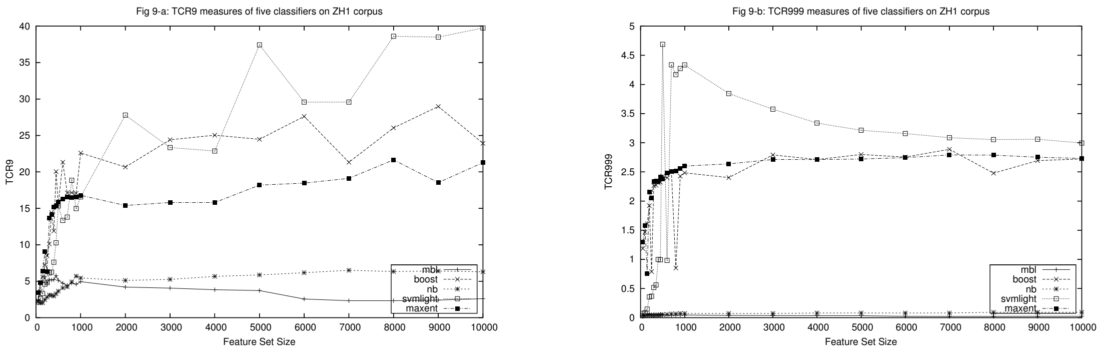


Figure 9: TCR measures of five classifiers on ZH1 corpus with features filtered with IG. The spam thresholds are $t_9 = 0.9$ and $t_{999} = 0.999$ for Naive Bayes and Maximum Entropy Model; $\theta_9 = 0.36$ and $\theta_{999} = 1.15$ for SVM^{light}; $\theta_9 = 3.0$ and $\theta_{999} = 9.24$ for AdaBoost.

dataset in terms of TCR9 measure, followed by AdaBoost and Naive Bayes, with memory-based learner near the 1.0 baseline. Similar patterns are found on the other three corpora. However, when λ is set to 999 (Figure 6-b), SVM, Maximum Entropy Model and AdaBoost are the clear winners of this game, outperformed Naive Bayes and Timbl by a wide margin. AdaBoost worked especially well on PU1 with $\lambda = 999$. Its TCR999 measure beats Maximum Entropy Model on 16 points and outperforms SVM on 10 points (Table 4).

For Ling-Spam corpus, we observe that Maximum Entropy Mode performed comparably with SVM in terms of TCR9 (Figure 7-a), but achieved a much lower TCR999 value than AdaBoost and SVM on large feature sets (Figure 7-b). SVM worked particularly well on this corpus with large feature set, considerably better than AdaBoost (12-10-2) and Maximum Entropy (9-14-1) using TCR999 measure.

For SA corpus, we find Maximum Entropy Model is the most effective classifier on this corpus. Its TCR9 measure wins on 74 points, ties on 22 points with 0 loss! Its TCR999 measure is also better than AdaBoost (18-6-0) and SVM (12-10-2). The memory based learner surpasses Naive Bayes on TCR9 measure (21-3-0), but is still inferior to the other classifiers (Figure 8-a).

On the newly compiled ZH1 corpus, good TCR9 values are obtained by all classifiers (Figure 9-a). Again, Naive Bayes and the memory-based approach fail to achieve a better-than-baseline performance when λ is set to 999 (Fig 9-b). SVM shows its advantage over AdaBoost (12-9-3) and Maximum Entropy Model (14-7-3) on this dataset judging from TCR999. In addition, the experimental results on ZH1 corpus verify the hypothesis that "bag of features" filtering model should also work well on Chinese language.

From the above analysis, we can conclude that Support Vector Machine, AdaBoost and Maximum Entropy Model are effective methods for automated spam filtering task where false positives are assigned a much higher cost than false negatives. In contrast, we find the commonly used Naive Bayes classifier and a memory based learner are not good choices for such applications. Moreover, we notice that the performance of a classifier can vary greatly from one dataset to another, and it seems no learning method can be superior on all datasets. For instance, using TCR999 measure we find Maximum Entropy Model comes out on top on SA corpus with a total 78 wins, 16 ties, and 2 losses, but loses its place on the other three corpora. SVM shows a clear advantage over ME and AdaBoost on ZH1 corpus with a total 73 wins, 17 ties, and 6 losses, but is slightly outperformed by AdaBoost on PU1 corpus (SVM's 69-9-18 against AdaBoost's 73-16-7 win-tie-loss counts).

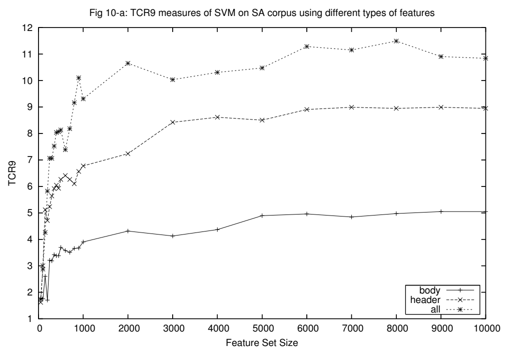
Combined with the performance curves shown from Figures 6 to 9 we notice that the predicting behaviors of SVM, AdaBoost and Maximum Entropy Model are quite similar. The three learners are not sensitive to feature-pruning strategies and can handle very high feature dimensions gracefully with good performances across different attribute sets. Similar observation was made by [Drucker et al., 1999], where AdaBoost and SVM were reported comparable performances on two spam datasets. This calls for a closer comparison of the three learning methods. The similarity between SVM and AdaBoost can be clarified by the fact that the two algorithms seek to maximize the minimum margins of training examples that are only different in norms of instance vector and weight vector [Freund and Schapire, 1999]. Viewed in this way, SVM and AdaBoost are very similar. The difference between large margin classifiers (SVM, AdaBoost) and Maximum Likelihood Estimation of exponential model (Maximum Entropy Model) seems to be obvious at first glance. However, there exists strong connection between AdaBoost and Maximum Entropy Model. The work of [Lebanon and Lafferty, 2001] showed that AdaBoost and Maximum Likelihood of exponential models actually minimize the same Kullback-Leibler divergence objective function subject to identical feature constraints, and they typically yield identical results as the number of features increases to allow the models to fit the training data. Our experiments confirmed this argument.

5.3 Experiment III: Mail Header, Body, or All Features?

The SA and ZH1 corpus both come in raw format: each mail consists of a header part followed by a body part and several optional attachment parts. While previous researches mainly focused on building "pure" content based filtering model that only uses features from message's subject and body parts, some experimental results on using non-textual features that are usually hidden in mail headers have proved to be helpful in classifying spams [Sahami et al., 1998, Zhang and Yao, 2003]. Still, it is unclear to what extent the features in header part could improve the filtering result.

In order to measure the contributions of the different parts of message in spam filtering, we processed the SA and ZH1 corpora into three versions: one uses only terms from message body plus subject line; one with tokens occur in message headers only and one with both mail body and headers tokenized. The tokenization scheme was the same as described in Section 4. Then we ran the SVM^{light} classifier on the two corpora's six variants using the same experimental setting as used in Experiment II. The TCR9 and TCR999 results are plotted in Figures 10 and 11, with θ_9 and θ_{999} tuned for each version of the training data separately.

The first thing that catches our attention in Figures 10 and 11 is the result of header feature: SVM classifier achieved good TCR values using information from mail headers only. The TCR9 values of the "header" curves in



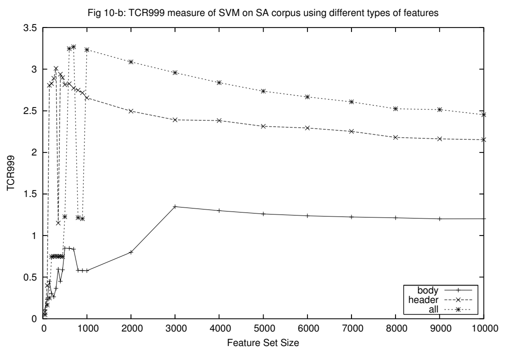
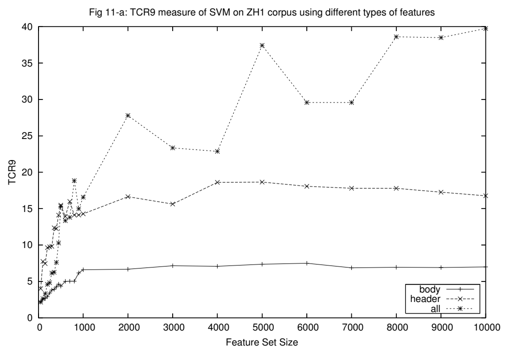


Figure 10: TCR result of SVM on SA corpus. The θ_9 thresholds for body, header and all (body+header) feature types are 0.63, 0.575 and 0.48 respectively. The corresponding θ_{999} values are 2.31, 1.43 and 1.38.



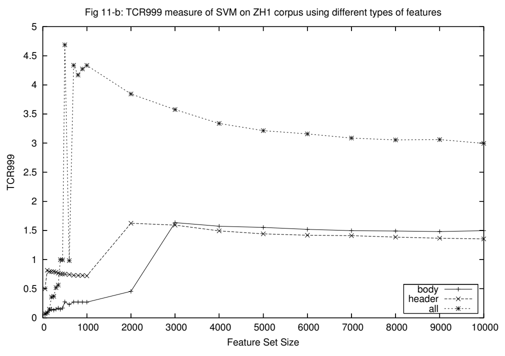


Figure 11: TCR result of SVM on ZH1 corpus. The θ_9 thresholds for body, header and all (body+header) feature types are 0.771, 0.687 and 0.36 respectively. The corresponding θ_{999} values are 1.47, 1.69 and 1.15.

Figure 10-a and Figure 11-a are all significantly better then that of "body" curves, confirmed with $p < 0.05$ on feature sets larger than 200. The TCR999 result on SA and ZH1 corpus exhibits the same trend (Figure 10-b and 11-b) (except on ZH1 where "body" features worked slightly better when more than 3000 features were used). This finding is rather striking, since previous studies only focused on building "pure" content based model that mainly relys on tokens from mail bodies, effectively ignoring the information hidden in mail headers. Our observation implies that features in message headers can be more reliable in discriminating spams than terms occur in message body. Several attempts have been made to incorporate some header specific properties into spam filters [Sahami et al., 1998, Zhang and Yao, 2003]. However, to our knowledge this is the first time showing message headers along can be powerful discriminating sources for spam filtering. The excellent filtering results on header features reflect the fact that spam mail differs from normal user mail not only in the boasting words they use, but also in their abnormal behaviors in various stages of dispatching. Some particular features in the header field of a mail can give strong evidence whether a mail is spam or not. For instance, spam mails are seldom sent through normal email client such as Outlook Express or Mutt. Instead, spammers prefer to use some group mail sending software specially designed for dispatching mails to a large amount of recipients. This can be detected by looking at the X-Mailer field in the header. If a mail has an X-Mailer field indicating some group sending software or does not have X-Mailer field at all, it is very likely to be a spam.

As expected, optimal filtering results followed by combining features from header fields and mail body (the "all" curves in Figures 10 and 11). On the whole, the combined feature sets yielded a consistent improvement over feature sets that used either header or body feature alone. We therefore conclude that message headers are as important as mail bodies in terms of spam discriminating power and should not be treated with ignorance by spam filter designers.

6 Related Work

It is certainly true that there has been an explosion of interest and applications for spam filtering over the last several years, with many machine-learning techniques having been tried to this task. We recall some of them in this section.

[Cohen, 1996] used RIPPER rule learning algorithm to classify personal emails into pre-defined categories with a set of "keyword-spotting rules" automatically learned from corpus. Cohen showed that RIPPER algorithm can achieve comparable performance to a traditional TF-IDF weighting method on a multi-class categorization task, with greater comprehensibility.

Since its first application to spam filtering task [Sahami et al., 1998], Naive Bayes method has been extremely popular in this field, for its simplicity and fairly good performance, and often serves as baseline classifier for comparison with other filtering approaches. [Pantel and Lin, 1998]'s work showed NB can work better than RIPPER algorithm on spam filtering task. [Androutsopoulos et al., 2000b] conducted an extensive comparison of the performance of Naive Bayes and a memory-based learner. However, in this evaluation we find Naive Bayes is not a suitable choice for filtering configuration where false positives are assigned a very high cost (such as the situation of $\lambda = 999$).

[Kolcz and Alspector, 2001] proposed a content-specific cost model for filtering spam, utilizing the SVM learning paradigm. In their experiments the cost of misclassifying legitimate was content-specific and the cost of misclassifying spam was assumed to be uniform. Since their training model needs additional labeling of the training data, we did not use the content-specific cost model in this research. Yet it is worth noting when applied to real world email filtering scenario, where the distribution of misclassification is not uniform and is changing across time, incorporating content-specific misclassification costs during training may be more effective.

[Schneider, 2003] compared two event models for Naive Bayes model: a multi-variate Bernoulli model, where each term is used in a "bag of words" manner not counting the number of time a term is seen in a document, and a multi-nomial model which also considers term counts information. Schneider concluded that the multi-nomial model is less biased towards Bernoulli model, which discards feature count information. While most studies in spam filtering (including experiments conducted in this paper) concentrate on the multi-variate Bernoulli model, future work on multinomial model may yield better result.

Almost all filtering methods proposed so far come from the Text Categorization's point of view: transform data into the input of a machine learning algorithm and evaluate the output produced by that algorithm. Relatively few attempts have been made to exploit the linguistic characters of spams. [Orasan and Krishnamurthy, 2002] compared various linguistic features such as average sentence length, POS distribution, lexical frequencies (N-grams, lemma) in one junk corpus to BNC corpus, showing quite different characters exist between junk mails and the usual written text. Incorporating linguistic features can be a useful direction in further spam filtering research.

7 Conclusion

In this paper, we gave a comprehensive evaluation of five commonly used supervised learning approaches in the context of cost-sensitive spam filtering. Different feature pruning methods and sizes of feature space were tried on three English spam corpora and one Chinese spam corpus. In particular, we found:

- The "bag of features" filtering model can be quite effective on spam filtering task.

  On all the four spam corpora we tried, most classifiers were able to achieve filtering result much better than no filter is present (baseline) using "bag of features" model. Furthermore, The technique also proves to be language independent. We have gained satisfactory filtering results on a newly compiled Chinese corpus with a simple preprocessing of word segmentation.

- SVM, AdaBoost, and Maximum Entropy Model are top performers on cost-sensitive spam filtering task.

  The experiments show that in spam filtering the choice of classifier plays a more important role than the choice of feature selection method. Support Vector Machine, AdaBoost, and Maximum Entropy Model are top performers in cost-sensitive spam filtering task, outperforming Naive Bayes and a Memory-based learner by a wide margin. The three learners share similar characteristics: not sensitive to feature selection strategy, easily scalable to very high feature dimension, and good performances across different datasets. In addition, we notice that the sensitivity to feature selection method varies greatly from classifier to classifier. CHI and IG generally do a better job than DF, especially when feature set is small.

  The experiment also suggests that aggressive feature pruning should be avoided when building filters to be used in scenarios where legitimate mails are assigned a much higher weight than spams (such as $\lambda = 999$), so as to maintain a better-than-baseline result (TCR999 >1). The possible reason may be that the higher λ is, the more features should be used to let the classifier produce a reliable prediction.

- The information from message header is as important as message body.

While previous studies focused on classifying model based on message body only, we find the message header is precious and should not be treated with ignorance. Classifiers trained on message headers actually yielded comparable or better results than the message body solution. Using all information in a message (header + body) typically generates better classifier than using either part alone.

Acknowledgments

The authors are grateful to two anonymous reviewers whose thoughtful comments improved the quality of this paper, especially the use of TCR measure in evaluation. We thank Antal van den Bosch for sharing the information on the internal of the TiMBL learner. This research was partially funded by the National Natural Science Foundation of China and Microsoft Research Asia under grant No. 60203019 and a grant from the National Natural Science Foundation of China (No. 60473140).

References

I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M.n van Someren, editors, *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000a. URL `http://arXiv.org/abs/cs.CL/0006013`.

I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, , and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France, 2000b. URL `http://arXiv.org/abs/cs/0009009`.

Ion Androutsopoulos, John Koutsias, Konstandinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 160–167, Athens, GR, 2000c. ACM Press, New York, US. URL `http://www.acm.org/pubs/articles/proceedings/ir/345508/p160-androutsopoulos/p160-androutsopoulos.pdf`.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001. URL `http://www.lsi.upc.es/ carreras/pub/boospam.ps`.

W.W. Cohen. Learning rules that classify e-mail. In *Spring Symposium on Machine Learning in Information Access*, Stanford, California, 1996.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. Timbl: Tilburg memory-based learner - version 4.3 reference guide, 1999.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Laf ferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998. URL `citeseer.ist.psu.edu/dietterich98approximate.html`.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000. URL `citeseer.nj.nec.com/dietterich98experimental.html`.

Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5), 1999.

R. Fano. *Transmission of Information.* MIT Press, Cambridge, MA, 1961.

Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5) (1999), 771-780. 11(5):771–780, 1999.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

E. Jaynes. *Papers on Probability, Statistics, and Statistical Physics.* D. Reidel Publishing Company, 1983.

T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998a.

Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998b. Springer Verlag, Heidelberg, DE. URL `citeseer.nj.nec.com/joachims97text.html`.

Aleksander Kolcz and Joshua Alspector. SVM-based filtering of e-mail spam with content-specific misclassification costs. In *Proceedings of the TextDM'01 Workshop on Text Mining - held at the 2001 IEEE International Conference on Data Mining*, 2001.

Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems*, 2001.

David D. Lewis. Evaluating and optmizing autonomous text classification systems. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995. ACM Press, New York, US. URL `http://www.research.att.com/ lewis/papers/lewis95b.ps`.

David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. URL `http://www.research.att.com/ lewis/papers/lewis98b.ps`. Published in the "Lecture Notes in Computer Science" series, number 1398.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

C. Orasan and R. Krishnamurthy. A corpus-based investigation of junk emails. In *Language Resources and Evaluation Conference (LREC- 2002)*, Las Palmas, Spain, 2002.

Patrick Pantel and Dekang Lin. Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. URL `http://www.cs.ualberta.ca/ ppantel/Download/Papers/aaai98.pdf`.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schlkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1998.

R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language1996*, 10:187– 228, 1996. Longe version: Carnegie Mellon Tech. Rep. CMU-CS-94-138.

Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. URL `http://robotics.stanford.edu/users/sahami/papers-dir/spam.ps`.

Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

Karl-Michael 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 (EACL'03)*, 2003. URL `http://www.phil.uni-passau.de/linguistik/mitarbeiter/schneider/pub/eacl2003.pdf`.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. URL `http://www.isti.cnr.it/People/F.Sebastiani/Publications/ACMCS02.pdf`.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, NY, USA, 1995.

Yiming Yang and Christopher G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252 – 277, 1994.

Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US. URL `http://www.cs.cmu.edu/ yiming/papers.yy/sigir99.ps`.

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US. URL `http://www.cs.cmu.edu/ yiming/papers.yy/ml97.ps`.

Le Zhang and Tianshun Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages (ICCPOL03)*, pages 446–453, 2003.