A Study of Supervised Spam Detection applied to Eight Months of Personal E-Mail

Gordon Cormack and Thomas Lynam

Introduction

In the last year or two, unwelcome email has grown to the extent that it is inconvenient, annoying and wasteful of computer resources. More significantly, its volume threatens to overwhelm our ability to recognize welcome messages. An automatic spam filter can mitigate these problems, provided that it acts in a reliable and predictable manner.

We evaluate ten spam detection methods embodied in six popular open-source spam filters by applying each method sequentially to all of the e-mail received by one individual (X) from August 2003 through March 2004. These 49,086 messages were originally judged in real-time by X. The messages and judgements were recorded, and reproduced so as to provide the same evaluation suite for all the methods. Five of the methods are derived from Spamassassin [spamassassin.org 2004], a hybrid system which includes both static spam-detection rules and a Bayesian statistical learning component. The purpose of intra-Spamassassin comparison is evaluate the relative contributions of the static and learning components in various configurations. The other methods are all "pure" statistical learning systems, in that they contain essentially no spam-detection rules. We compare these – Bogofilter [Raymond 2004], CRM-114 [Yerazunis 2004a], DSPAM [Zdziarski 2004], SpamBayes [Peters 2004], and Spamprobe [Burton 2002a] – against each other and against the learning component of Spamassassin. Recent evaluations have reported very high accuracies – some higher than 99.9% – for several of these filters [Zdziarski 2004; Yerazunis 2004a; Louis 2004; Yerazunis 2004b; Holden 2004; Burton 2002b]. We contrast these studies and others [Sahami et al. 1998; Androutsopoulos et al. 2000; Tuttle et al. 2004] with ours following the presentation of our results.

Our study's objective is to provide a controlled, realistic, statistically meaningful evaluation of several common filters and the methods they embody. While our study is limited to the extent that X's email is typical, and to the extent that the ten subject methods represent the state of the art, we present our methods and analysis in sufficient detail that they may be reproduced with other email streams and filter implementations. In addition, we have archived our evaluation suite so that we may use it to evaluate future spam detection methods[1].

Spam filtering may be effected in a number of configurations, which we categorize as: *manual, static, unsupervised, and supervised*. With manual filtering (see figure 1), the task falls to the email recipient who must examine and classify each message that is received. As mentioned above, this process is tedious and error-prone, especially as the volume of unwanted email *(spam)* approaches or exceeds the volume of wanted email *(ham)*. With static filtering (see figure 2), an automatic filter

[1] As it contains X's private email, we are not at liberty to publish our evaluation suite.

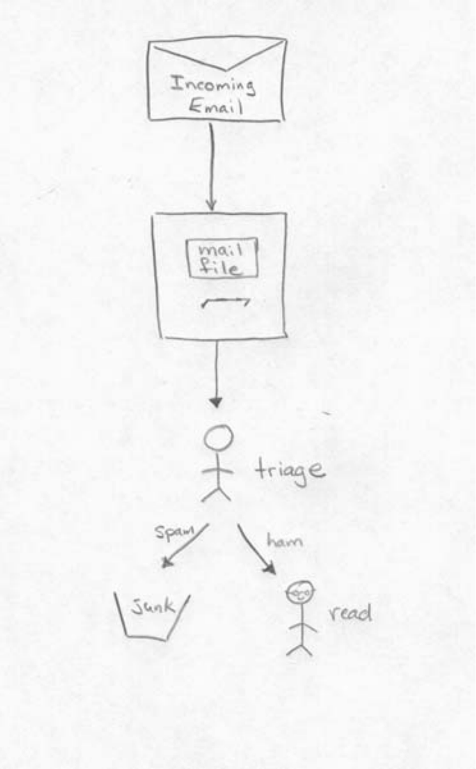

Fig. 1. Manual Configuration

examines each message and sends it to one of two outputs: a *ham file* consisting of messages likely to be ham, and a *spam file* consisting of messages likely to be spam. In the normal course of reading email, the recipient consults only the ham file, performing the same process as for manual filtering, but with a much-reduced volume of spam. The recipient may (or may not) occasionally consult the spam file in an effort to recover ham that may have been misclassified by the automatic filter. Unsupervised filtering (see figure 3) differs from static filtering only in that the automatic filter has *memory*; in addition to classifying email messages, the filter records characteristics of these messages and uses this recorded information to aid in classifying subsequent messages. Supervised filtering (see figure 4) involves the filter and recipient in a closed loop; the recipient regularly examines the ham and spam files and reports misclassifications back to the filter, which updates its memory accordingly.

A spam filter can be considered effective to the extent that two undesirable aspects are minimized: the amount of spam that the filter fails to identify, and the end-to-end probability of losing important email. The first aspect is aptly characterized as a proportion that may be estimated from a sample. The second aspect – the end-to-end probability of loss – is more complex to characterize and to estimate. Important email may be lost if the filter misclassifies it as spam and the

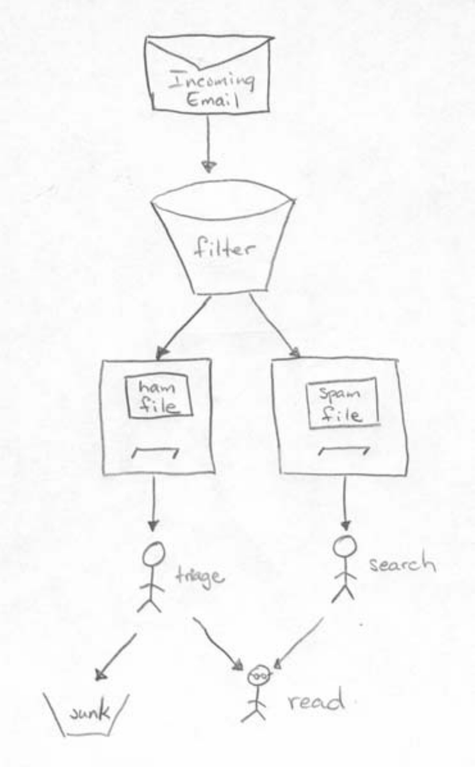

Fig. 2. Static Configuration

recipient fails to retrieve it from the spam file, or if the filter classifies it correctly and the recipient fails to notice it in the ham file. Both eventualities must be considered in assessing the end-to-end probability of loss. Minimizing the filter's proportion of misclassified ham is one component of this effort. In addition we must consider the effect of spam misclassification on the recipient's ability to identify important messages in the ham file, characteristics of ham messages that might cause them to be misclassified, characteristics of misclassified ham messages that might affect the recipient's likelihood of retrieving them from the spam file, and characteristics of misclassified ham messages that determine their *importance* to the recipient.

Method

Test Suite

X has had the same userid and domain name for 20 years; variants of X's email address have appeared on the Web, and in newsgroups. X has accounts on several machines which are forwarded to a common spool file, where they are stored permanently in the order received.

X began using a spam filter in 2002 when the proportion of spam in his email began to exceed 20%, causing X to overlook two important messages which arrived amongst bursts of spam. Since August 2003, X has used Spamassassin 2.60 in a

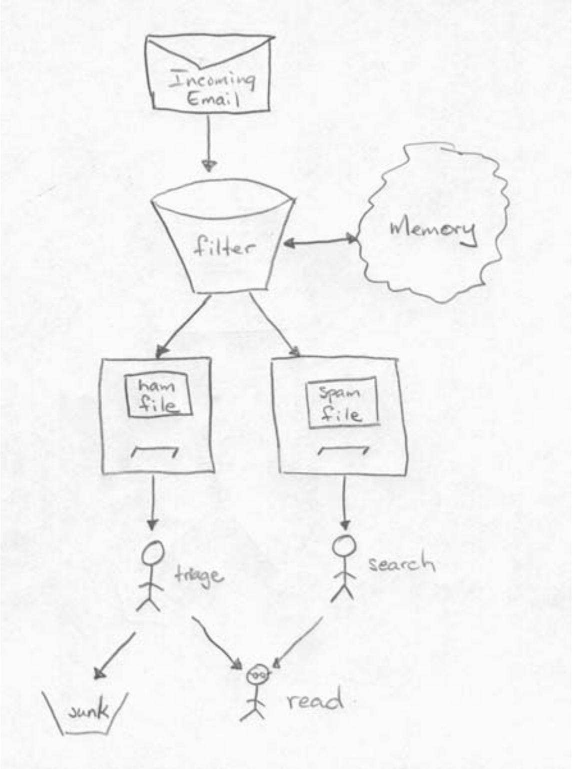

Fig. 3. Unsupervised Configuration

supervised configuration to classify this incoming mail. It was necessary to modify Spamassassin to incorporate this use, as Spamassassin was designed to be used primarily in the unsupervised configuration. User feedback was facilitated by two macros (*No, it's Spam*, and *No, it's Ham*) added to X's mail client. Spamassassin records every judgement (whether automatic or due to user feedback) in its learning database, so it was possible to recover preliminary *gold standard* judgements from this database.

Algorithm 1 Supervised Email Filtering

```
$filterinit()$

Foreach $msg_i$ in $Archive$
  $result_i, score_i \leftarrow filtereval(msg_i)$
  if $goldstandard_i = \text{ham} \;\&\; result_i \neq \text{ham}$
    $filtertrain(msg_i, \text{ham})$
  if $goldstandard_i = \text{spam} \;\&\; result_i \neq \text{spam}$
    $filtertrain(msg_i, \text{spam})$
end.
```

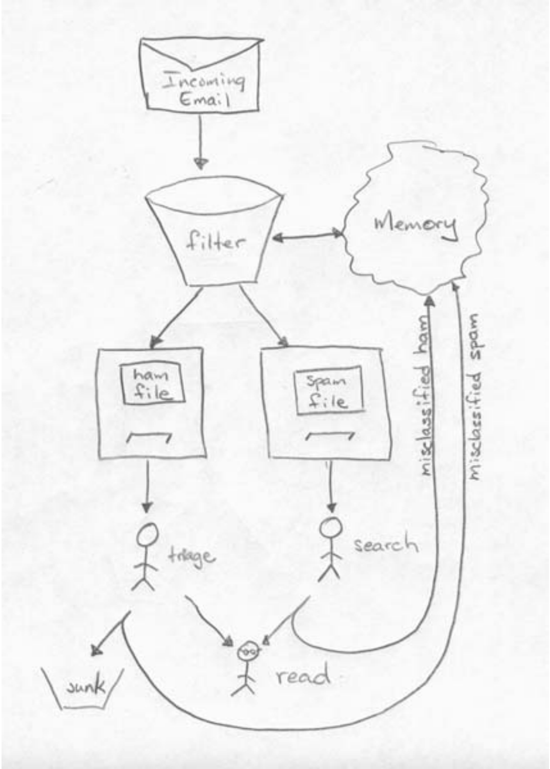

Fig. 4. Supervised Configuration

Algorithm 2 *Train on everything* Email Filtering

```
procedure$filtereval(msg)$
  $result, score \leftarrow localfiltereval(msg)$
  $filtertrain(msg, result)$
  return $result, score$
end.
```

Each trial run is an idealized[2] reproduction of X's behaviour since August 2003, with a different filter in place of Spamassassin 2.60. As shown in algorithm 1, the subject filter is presented with each message from the spool in the original order of arrival. Each filter was encapsulated using three common interface procedures: *filterinit, filtereval,* and *filtertrain. filterinit* sets the filter's memory to a clean initial state; *filtereval* is given an email message and returns a pair consisting of a classification and a score; *filtertrain* is given an email message and a new classification, if the gold standard differs from the result of *filtereval.* We note that this interface implements a *train only on errors* strategy; filters requiring a *train*

[2] Idealized in that feedback to the filter is immediate and completely accurate.

on everything strategy were accommodated by incorporating self-training into the implementation of the *filtereval* access function as shown in algorithm 2.

We conducted one pilot run using a static configuration of Spamassassin 2.63 (i.e. with the learning feature disabled). This run allowed us to double-check the veracity of X's original judgements. All messages in which the original judgements disagreed with those of the pilot were checked again. The vast majority of spam misclassifications were resolved in favour of Spamassassin (i.e. X had neglected to classify a spam message as such), and the gold standard was corrected. All subsequent disagreements between the gold standard and later runs were also manually adjudicated, and all runs were repeated with the updated gold standard. The results presented here are based on this revised standard, in which all cases of disagreement have been vetted manually.

After the runs were complete, each misclassified ham message was examined and assigned one of seven *genres* that we believed might be associated with the likelihood of misclassification and the importance of the email to the recipient. We also assigned a genre to each of a random sample ($n = 352$)of all incoming ham. Similarly, we assigned one of five different genres to each spam message misclassified by one or more of the four best-performing systems, and also to a random sample of spam messages ($n = 100$) misclassified by each of the other systems. We also assigned a genre to each of a random sample ($n = 142$) of all incoming spam.

Subject Configurations

We chose to evaluate open-source systems representing best practice in spam detection. Two systems [mozilla.org 2004; Graham-Cumming 2004] were initially chosen for evaluation but later excluded because a prohibitive effort would have been required to isolate the interfaces necessary to conform to the interface defined in algorithm 1.

Ten runs were conducted. Five of the runs evaluated different configurations of Spamassassin; five of the runs evaluated the other systems. In addition, X's in situ judgements were evaluated as if they had been the results of an eleventh run. For evaluation purposes, this "eleventh run" is grouped with the Spamassassin runs.

The Spamassassin Runs

Spamassassin's static component uses a database of user-submitted patterns and procedures that recognize mail features likely to indicate spam. Through a distributed statistics-gathering effort, scores are deduced for the features. These scores are fixed for a particular release of Spamassassin. Although most of the scores may be specified as configuration parameters, no adjustment (and hence no training) occurs as a consequence of filtering incoming messages.

In classifying a particular message, the scores for the features found in the message are added to yield an overall score, which is compared to a threshold. In all runs, the default value of 5.0 was used for this threshold.

Spamassassin 2.5 introduced a Bayes filter based on Graham's *A Plan for Spam* [2002; 2004]. Spamassassin 2.6 incorporated improvements due to Robinson [2004; 2003]. Spamassassin's Bayes filter operates independently of the other tests, and contributes a (possibly negative) value which is added to the overall score in classifying a message.

By default, Spamassassin's Bayes filter is configured in a conservative manner appropriate for unsupervised learning as might occur in a mail server. It self-trains only on messages which receive an extreme score from the static component alone, excluding the contribution of the Bayes filter. Furthermore, training takes place only if there are three *body hits* and three *header hits* from among the scores, and only if the Bayes filter did not already classify the message correctly. These exclusions from self-training are hard-coded into Spamassassin and cannot be overridden by user parameters. For the runs that used supervised learning, we modified Spamassassin to self-train on *every* message, as in algorithm 2.

The Spamassassin runs are enumerated below.

(1) **SA-Nolearn** - *Spamassassin 2.63* with the learning component disabled.

(2) **SA-Standard** - *Spamassassin 2.63* with default unsupervised learning parameters.

(3) **SA-Supervised** - *Spamassassin 2.63* with supervised learning, modified as described above to self-train on every judgement. Run in the context of algorithm 1, so as to retrain the system for each misclassified message.

(4) **SA-Unsupervised**. *Spamassassion 2.63* with *train on everything* unsupervised learning. The system was used as modified for SA-Supervised, but no retraining was done for misclassified messages.

(5) **SA-Bayes** - *Spamassassin 2.63* learning component only. The system was modified and used as for SA-Supervised, but the scores for all static rules were set to 0, and the threshold value was set to 0. Any positive score was deemed to indicate spam.

(6) **SA-Human** - *X,* aided by Spamassassin. The judgements originally rendered by X, in the normal course of reading email and supervising Spamassassin 2.60 (modified to train on everything, as for SA-Supervised). The judgements were captured from the memory of the system (c.f. figure 4).

Pure Learning Filter Runs

Except as noted, the learning filters were installed using default threshold and tuning parameters. Prior training was not used; *filterinit* initialized the filter's memory to the empty state.

(1) **CRM114** - *The Controllable Regex Mutilator*, version *20040328-Blame St. Patrick-auto.1.* CRM114 uses Markov modelling and a sliding window to find the features in messages must likely to indicate spam. CRM114 classifies each message by computing a score which it compares to a threshold. CRM114 does not self-train; we trained the system only after misclassifications, as suggested in the documentation. We did not use the whitelist or blacklist facilities supplied with CRM114. Filter memory size was set at 10000001 *buckets* for both ham and spam.

(2) **DSPAM** - *The DSPAM Project: Statistical SPAM Protection*, version *2.8.3.* DSPAM is a Statistical-Algorithmic Hybrid filter whose designer strives for simplicity. DSPAM 2.8.3 self-trains on every message it classifies, and annotates the message with a signature that contains information necessary for it

to reverse this self-training. We augmented algorithm 1 to supply this annotated message, rather than the original, to *filtertrain.* We did not use the *purge* facility, which reduces the size of the statistical table maintained by DSPAM.

(3) **Bogofilter** - *Bogofilter*, version *0.17.5.* Bogofilter, due to Eric Raymond, is a Bayes filter, like Spamassassin's, modelled after the proposals by Graham and Robinson. Bogofilter emphasizes simplicity and speed.

(4) **SpamProbe** - *SpamProbe* version *0.9h.* A C++ Bayes filter inspired by Graham's proposal.

(5) **SpamBayes** - *Spambayes* version *1.061.* A Python Bayes filter inspired by the proposals of Graham and Robinson.

Analysis

A contingency table (table I) enumerates the possible outcomes of applying a filter to a mail stream. The primary measures of interest are the *ham misclassification fraction,* $hm = \frac{c}{a+c}$, and the *spam misclassification fraction* $sm = \frac{b}{b+d}$. We also report the (overall) *misclassification fraction,* $m = \frac{b+c}{a+b+c+d}$, because it is equivalent to *accuracy* ($m = 1 - accuracy$)[3], which is commonly reported.

	Gold Standard		
		ham	spam
Filter	ham	a	b
	spam	c	d

Table I. Contingency Table

The result of applying a filter to a spam message is a dichotomous variable *result* taking on the value *ham* or *spam.* In our primary analysis, we estimate the probability of each outcome as a function of *true*, the true classification, also a dichotomous value taking on the value *ham* or *spam.* In particular, *hm* estimates $Pr(result = \text{spam} \,|\, true = \text{ham})$ and *sm* estimates $Pr(result = \text{ham} \,|\, true = \text{spam})$. *m*, on the other hand, estimates $Pr(result \neq true)$. These estimates are the result of three separate sets of Bernoulli trials; one for ham messages ($true = \text{ham}$), one for spam messages ($true = \text{spam}$), and one for all messages.

Each set of trials consists of n observations, x of which exhibit the truth value whose probability P is to be estimated. Given P and n, the probability of any particular value of x is determined exactly by the binomial distribution. The cumulative probability over all $t \leq x$ is the sum of $x+1$ discrete binomial probabilities. Since $x \leq n < 50,000$ for each of the three sets, we were able to calculate cumulative probabilities with minimal computational cost.

Given n and x, the maximum likelihood estimate for P is simply $\frac{x}{n}$. 95% confidence limits are computed as follows. When $x = 0$, the lower confidence limit is 0

[3] We quantify misclassifications rather than accuracy so as to avoid presenting nearly equal numbers that represent large differences in performance. Graphical results are displayed using the logistic transformation, $logit(p) = log(\frac{p}{1-p})$, which maps the range $[0 : 1]$symmetrically to the range $-\infty : \infty$.

and the upper confidence limit is the smallest P such that the cumulative binomial probability over all $t \leq x$ is less than 0.05. When $x > 0$, the lower confidence limit is the largest P such that the cumulative binomial probability over all $t \geq x$ is less than 0.025; the upper confidence limit is the smallest P such that the cumulative binomial probability over all $t \leq x$ is less than 0.025. Each P was computed using binary search.

Because all filters are applied to the same messages, we are able to use exact paired tests to evaluate differences that might not be apparent from comparing misclassification proportions. For a pair of filters, A and B, we count each of the four possible pairs of results when A and B are applied to the same message. Table II illustrates the four possible outcomes: a is the number of times that the filters both return *ham*; d is the number of times they both return *spam*; b is the number of times A returns *spam* while B returns *ham*; c is the number of times A returns *ham* while B returns *spam.* a and d, the cases of agreement, do not differentiate the systems and may be ignored. b and c, the cases of disagreement, are the cases of interest. The disagreement cases constitute a set of Bernoulli trials with $n = b + c$, $x = b$. Under the null hypothesis (that A and B exhibit the same performance), $P = 0.5$, and $E(x) = \frac{n}{2}$. Any non-zero difference $|x - \frac{n}{2}| > 0$ must be due either to chance or to the falsity of the null hypothesis. p, the chance probability, is computed as the sum of binomial probabilities for all t such that $|t - \frac{n}{2}| \geq |x - \frac{n}{2}|$.

		Filter A	
Filter B		ham	spam
	ham	a	b
	spam	c	d

Table II. Matched-Pair Result Table

In this study, we test several hypotheses. For those that are amenable to statistical inference we state confidence intervals and declare significant differences based on the error probability $\alpha = 0.05$. As for any set of statistical inferences, whether from the same or separate studies, we must be aware that some results reported as significant will in fact be due to chance. According to Streiner [1986]:

> Of course, most statistical analysis uses an α-level of 0.05, which means that there is one chance in 20 that they will conclude there is some difference when there isn't. This also means that of every 20 "significant" differences reported in the literature, one is wrong. Wonder which one it is!

That said, we shall avoid a discussion of the philosophy of statistics and defer to common practice.

A fallacy not admitted by common practice is to perform several hypothesis tests and to report only those yielding a "significant" result. If we perform n tests, we expect about αn of them to show $p < \alpha$, even if the null hypothesis holds in every case. On the other hand, only α of the tests would show $n \cdot p < \alpha$ under the null hypothesis; in other words, the chance of some test showing $n \cdot p < \alpha$ is α.

Bonferroni correction captures this effect: when selected from a set of n tests, any test showing $n \cdot p < \alpha$ is significant with (Bonferroni corrected) $p < \alpha$. Bonferroni correction may be applied repeatedly using Holm's stepdown method [Holm 1979]: the result with smallest p is selected from the set; if it is significant after Bonferroni correction, the test is removed from the set and the process repeated with the remaining $n - 1$ tests. If the result with the smallest p is not significant, none of the remaining results is considered significant. When we rank the results of n tests, we are in effect performing $\frac{n(n-1)}{2}$ paired tests, which we correct using Holm-Bonferroni stepdown method.

Receiver operating characteristic (ROC) analysis [Park et al. 2004] is used to evaluate the trade-off between ham and spam misclassification probabilities. Using each of the numerical scores returned by a given filter, we conduct a hypothetical run to determine the ham and spam misclassification fractions that would have resulted had that score been used as a threshold. The set of pairs *(hm, 1-sm)* resulting from the hypothetical runs define a monotone non-decreasing function that is plotted as an ROC curve. As a summary measure of the relationship between ham and spam misclassification fractions over all possible thresholds, we present $1 - ROCAC$, where ROCAC is the area under the ROC curve. $1 - ROCAC$ estimates the probability that a random spam message is (incorrectly) given a lower score than a random ham message. ROCAC estimates and 95% confidence intervals were computed using SPSS 12.

Logistic regression [Agresti 1996] is used to evaluate the effect of the number n of messages processed on the probability P of ham or spam misclassification (i.e. the *learning curve).* P and n are assumed to be related by the formula $logit(P) =_{def} log(\frac{P}{1-P}) = \alpha + n\beta$ (alternatively, $\frac{P}{1-P} = e^{\alpha}e^{n\beta}$) for some α and β. Maximum likelihood estimates for α and β, 95% confidence limits, and p-values (for the null hypothesis that $\beta = 0$) were computed using SPSS 12. $\frac{P}{1-P}$ is the *odds* (as opposed to the probability) of misclassification; i.e. the ratio of incorrect to correct classifications. e^{α} is the *initial odds* when $n = 0$, and $e^{n\beta}$ is the *odds ratio*; for every n messages the odds increase (or decrease) by a factor of $e^{n\beta}$. For small P, odds and probability are nearly equal, so we may consider $e^{n\beta}$ also to be the *risk ratio;* for every n messages the probability of misclassification changes by this same constant factor.

A piecewise graphical estimate of $logit(P)$ vs. n is juxtaposed with the logistic regression curve as a visual indicator of the appropriateness of the logistic model. Estimates of initial and final misclassification rates, as well as the odds ratio, are tabulated with 95% confidence limits.

Within each genre of ham identified in our post-hoc classification, we estimated the proportion of incoming ham messages and, for each filter, the proportion messages misclassified by that filter. The ratio of these proportions provides an estimate the relative difficulty that each filter has in classifying messages of different genres, and an estimate of the maximum likely confounding effect due to each particular genre.

Results

The test sequence contained 49,086 messages. Our gold standard classified 9,038 (18.4%) as ham and 40,048 (81.6%) as spam. The gold standard was derived from X's initial judgements, amended to correct errors that were observed as the result of disagreements between these judgements and the various runs.

Classification Performance - Spamassassin Variants

Table III and figure 5 report the performance of our Spamassassin subject runs.

Filter	Ham Misc. (%)	Spam Misc. (%)	Overall Misc. (%)	1-ROC Area (%)
SA-Supervised	0.07 (0.02-0.14)	1.51 (1.39-1.63)	1.24 (1.15-1.35)	0.06 (0.04-0.07)
SA-Bayes	0.17 (0.09-0.27)	2.10 (1.96-2.24)	1.74 (1.63-1.86)	0.15 (0.11-0.18)
SA-Nolearn	0.19 (0.11-0.30)	9.49 (9.21-9.78)	7.78 (7.54-8.02)	0.80 (0.74-0.86)
SA-Standard	0.07 (0.02-0.14)	7.49 (7.23-7.75)	6.12 (5.91-6.34)	1.00 (0.93-1.06)
SA-Unsupervised	0.11 (0.05-0.20)	8.11 (7.84-8.38)	6.63 (6.41-6.86)	0.82 (0.76-0.88)
SA-Human	0.09 (0.04-0.18)	1.06 (0.97-1.17)	0.88 (0.80-0.97)	-

Table III. Filter Misclassification - Spamassassin Components

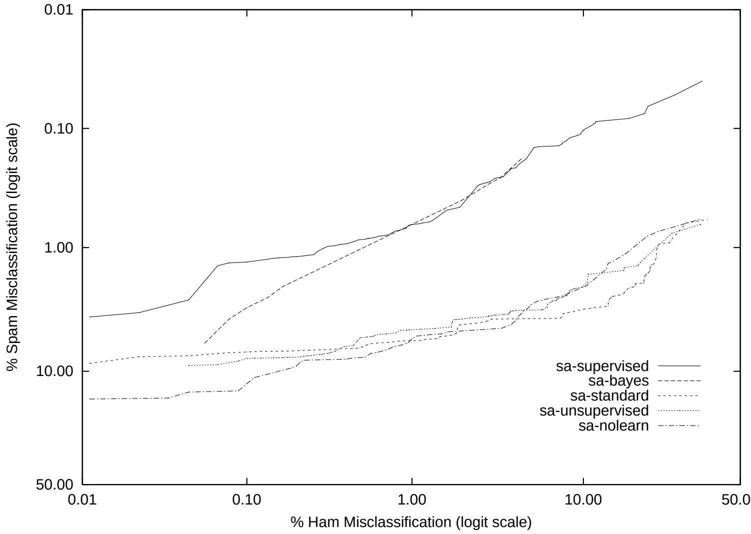

Fig. 5. ROC Curves for Spamassassin Variants

SA-Supervised, our baseline run, misclassifies 6 of 9,038 ham messages (0.07%) and 605 of 40,048 spam messages (1.51%). Overall, SA-Supervised misclassifies 611 of 49,086 messages (1.24%). The area under the ROC curve, *ROCAC*, is 0.9994 which we report as 1-ROCAC (%) or 0.06.

The SA-Supervised filter is a committee of two distinct components: *SA-Nolearn,* a static rule-based filter, and *SA-Bayes*, a pure learning filter. Taken separately, each component shows inferior performance to the baseline according to all four measures. We note in particular that SA-Supervised shows 2.5 times fewer ham misclassifications than either SA-Bayes ($p < .004$) or SA-Nolearn ($p < .035$), two-thirds as many spam misclassifications as SA-Bayes ($p < 10^{-10}$) and 6 times fewer spam misclassifications than SA-Nolearn ($p < 10^{-10}$).

SA-Standard uses Spamassassin's default configuration: the same static and learning filter, but with the filter trained only on errors, as adjudicated by the difference in results between the learning filter and a separate (more conservative) internal invocation of the static filter. In contrast, SA-Unsupervised trains on every judgement returned by *filtereval*. Both runs are unsupervised in that they operate autonomously with no human intervention. As with SA-Supervised, both runs show fewer ham and spam misclassifications than either SA-Bayes or SA-Nolearn taken separately. Of the differences in ham misclassifications only the difference between SA-Standard and SA-Nolearn may be interpreted as significant ($p < .035$). All differences in spam misclassification are significant ($p < 10^{-10}$).

SA-Human uses essentially the same configuration as SA-Supervised, but the system was supervised by X in real-time. That is, for every misclassification observed by X, the system was retrained and the human-corrected classification was recorded as the result for SA-Human. While SA-Human resulted in two more ham misclassifications than SA-Supervised (i.e. 8 vs. 6) no significant difference can be inferred. SA-Human resulted in two-thirds as many spam misclassifications ($p < 10^{-10}$).

We note that ham, spam, and overall misclassification rates rank the six runs in the same order. ROCAC inverts SA-Standard and SA-Unsupervised, and is inapplicable to SA-Human. Nevertheless, ROCAC ranking is consistent with the overall effect: that all tested combinations of static and learning filters outperform these individual components in isolation. The ROC curves show that SA-Supervised dominates the other runs, performing better than SA-Bayes when ham misclassification is minimized and as well when spam misclassification is minimized. SA-Supervised and SA-Bayes both dominate the remaining runs. These runs, SA-Nolearn, SA-Standard, and SA-Unsupervised, show ROC curves that intersect many times, indicating that their relative ROCAC scores are likely to be uninformative.

Classification Performance - Pure Learning Filters

Table IV and figure 6 show the classification performance of six pure learning filters (including SA-Bayes, the learning component of Spamassassin, also reported above). For this group of runs we have no baseline, and wish instead to evaluate their relative performance. The columns labelled *ham misclassification* and *spam*

Filter	Ham Misc. (%)	Spam Misc. (%)	Overall Misc. (%)	1-ROC Area (%)
Bogofilter	0.08 (0.03-0.16)	6.63 (6.39-6.88)	5.43 (5.23-5.63)	0.08 (0.05-0.10)
SpamBayes	0.17 (0.09-0.27)	5.86 (5.63-6.10)	4.81 (4.63-5.01)	0.16 (0.12-0.20)
SA-Bayes	0.17 (0.09-0.27)	2.10 (1.96-2.24)	1.74 (1.63-1.86)	0.15 (0.11-0.18)
SpamProbe	0.34 (0.23-0.49)	1.03 (0.93-1.14)	0.90 (0.82-0.99)	0.09 (0.05-0.13)
DSPAM	1.28 (1.06-1.54)	1.98 (1.84-2.12)	1.85 (1.73-1.97)	1.03 (0.90-1.17)
CRM-114	3.26 (2.91-3.65)	0.99 (0.90-1.09)	1.41 (1.31-1.52)	1.10 (0.94-1.27)

Table IV. Filter Misclassification - Pure Learning Filters

Ham Misc.	Spam Misc.
Bogofilter SpamBayes SA-Bayes	CRM-114 SpamProbe
SpamProbe	DSPAM SA-Bayes
DSPAM	SpamBayes
CRM-114	Bogofilter

Table V. Significant divisions ($p < .05$, Bonferroni-Holm corrected)

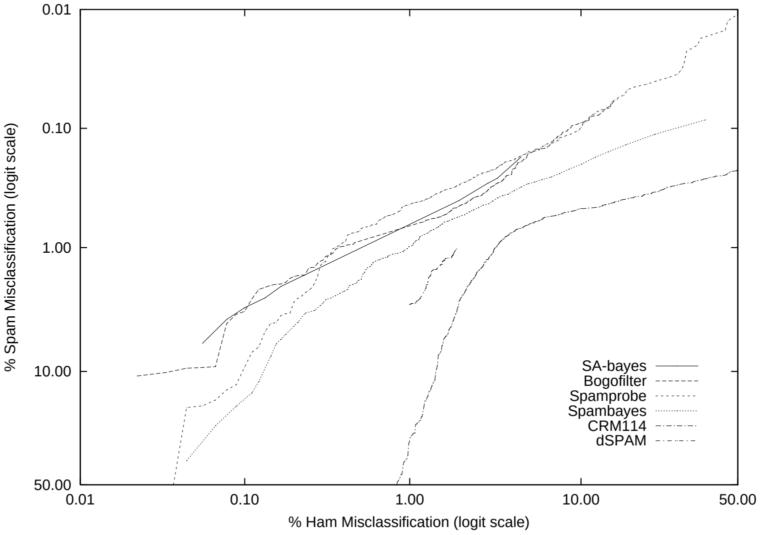

Fig. 6. ROC Curves for Supervised Learning Filters

misclassification show nearly opposite effects. Bogofilter offers the least number of ham misclassifications and the greatest number of spam misclassifications, while CRM shows the opposite.

To divide the filters into groups separated by significant differences in classification performance, we considered ham and spam separately; for each we performed a paired test between every pair of runs. Each column in table V summarizes the results of these 30 tests, corrected using Holm's stepdown method. Every pair of runs from different boxes shows a significant different difference ($p < 0.05$), while every pair in the same box does not. We see that the runs are divided into four groups with respect to ham classification performance, and four (different) groups with respect to spam classification performance. Although ham and spam misclassification performance yields nearly opposite rankings, we note that Bogofilter, SpamBayes, and SA-Bayes are distinguished by their spam performance but not by their ham performance. Similarly, CRM-114 and SpamProbe; and also DSPAM and SA-Bayes, are distinguished by their ham performance but not by their spam performance.

Overall Misclassification results are largely reflective of spam misclassification results, and are not analyzed further. The ROC curves show that the curves for Bogofilter, SpamProbe, and SA-Bayes intersect one another in many places throughout the operating range, but SA-Bayes and Bogofilter appear to have a lower spam misclassification proportion when the ham misclassification proportion is low (i.e. less than 0.3%). All three dominate SpamBayes by a narrow margin and dominate DSPAM and CRM-114 by substantial margins. ROCAC scores largely reflect the major differences observable in the curves, but fail to provide a meaningful distinction among Bogofilter, SpamProbe, SA-Bayes, and SpamBayes.

Effects of Learning on Classification Performance

Table VI summarizes the fraction of spam received by X as a function of the number of messages received. Although the overall spam fraction is 81.6%, logistic regression indicates that this fraction increased from 75.7% to 86.6% (an odds ratio of 2.07, $p < .001$) over the eight months during which our email stream was collected. Figure 7 shows a piece-wise approximation of this function juxtaposed with the regression line.

Initial Spam %	Final Spam %	Odds Ratio	p
75.7 (75.0, 76.6)	86.6 (86.0, 87.1)	2.07 (2.04, 2.10)	0.00

Table VI. Spam as a fraction of incoming messages

Tables VII and VIII summarize the ham and spam misclassification fractions as functions of the number of messages processed. Each row estimates the initial misclassification proportion, the final misclassification proportion, and the odds ratio between the two. 95% confidence limits and p-values are given for each. Figures 8 through 18 provide graphical representations of these functions.

Filter	Initial Misc. (%)	Final Misc. (%)	Odds Ratio	p
Bogofilter	0.19 (0.06, 0.62)	0.02 (0.00, 0.17)	0.08 (0.00, 1.98)	0.12
CRM-114	4.53 (3.71, 5.52)	2.08 (1.56, 2.75)	0.45 (0.29, 0.69)	0.00
DSPAM	1.52 (1.09, 2.12)	1.03 (0.67, 1.58)	0.68 (0.35, 1.33)	0.26
SA-Bayes	0.31 (0.13, 0.72)	0.06 (0.02, 0.26)	0.21 (0.03, 1.52)	0.12
SA-Human	0.01 (0.00, 0.09)	0.45 (0.15, 1.38)	54 (2, 1222)	0.01
SA-Nolearn	0.32 (0.14, 0.71)	0.09 (0.02, 0.31)	0.27 (0.04, 1.72)	0.17
SA-Standard	0.38 (0.12, 1.19)	0.00 (0.00, 0.07)	0.00 (0.00, 0.40)	0.02
SA-Supervised	0.19 (0.06, 0.66)	0.01 (0.00, 0.15)	0.05 (0.00, 1.80)	0.10
SA-Unsupervised	0.39 (0.15, 0.98)	0.01 (0.00, 0.10)	0.02 (0.00, 0.47)	0.01
SpamBayes	0.23 (0.10, 0.58)	0.10 (0.03, 0.37)	0.44 (0.07, 2.96)	0.40
SpamProbe	0.96 (0.56, 1.65)	0.05 (0.01, 0.17)	0.05 (0.01, 0.26)	0.00

Table VII. Ham Learning Performance

Of particular interest is the "learning" performance of SA-Nolearn; as this system has no learning component, its performance may be used to gauge any change in 'difficulty' of the spam messages over the eight months. Table VIII shows that SA-Nolearn's spam misclassification fraction increases from 7.73% to 11.37% ($p < .001$),

Filter	Initial Misc. (%)	Final Misc. (%)	Odds Ratio	p
Bogofilter	7.95 (7.41, 8.53)	5.50 (5.10, 5.94)	0.68 (0.59, 0.77)	0.00
CRM-114	1.90 (1.61, 2.24)	0.45 (0.35, 0.57)	0.23 (0.16, 0.33)	0.00
DSPAM	7.02 (6.33, 7.77)	0.23 (0.18, 0.30)	0.03 (0.02, 0.04)	0.00
SA-Bayes	2.51 (2.21, 2.85)	1.74 (1.52, 2.00)	0.69 (0.55, 0.87)	0.00
SA-Human	1.67 (1.40, 1.98)	0.64 (0.52, 0.79)	0.38 (0.27, 0.53)	0.00
SA-Nolearn	7.73 (7.25, 8.25)	11.37 (10.76, 12.02)	1.53 (1.37, 1.72)	0.00
SA-Standard	16.07 (15.22, 16.96)	2.67 (2.43, 2.92)	0.14 (0.13, 0.16)	0.00
SA-Supervised	1.68 (1.44, 1.96)	1.36 (1.16, 1.59)	0.81 (0.61, 1.07)	0.13
SA-Unsupervised	18.03 (17.13, 18.98)	2.67 (2.44, 2.92)	0.12 (0.11, 0.14)	0.00
SpamBayes	5.91 (5.46, 6.39)	5.82 (5.38, 6.29)	0.99 (0.85, 1.14)	0.82
SpamProbe	1.29 (1.08, 1.56)	0.81 (0.67, 1.00)	0.63 (0.45, 0.88)	0.01

Table VIII. Spam Learning Performance

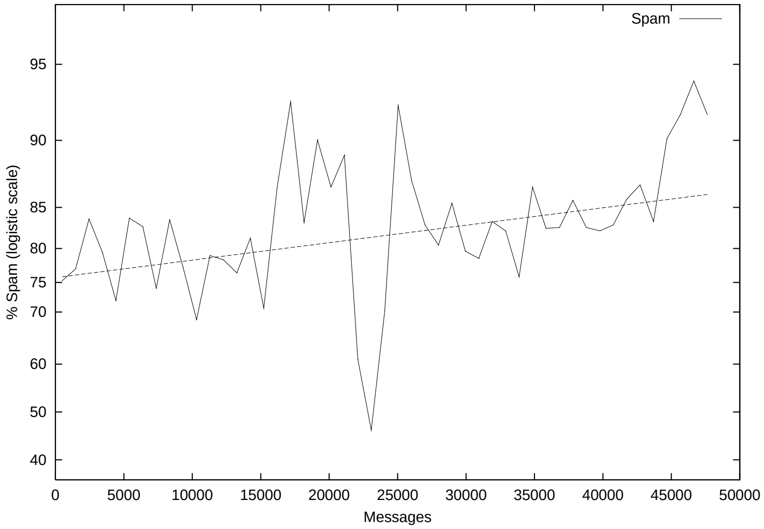

Fig. 7. Spam Growth

indicating that the nature of spam has changed so as to make it 'more difficult.' Figure 8 confirms this trend, but also shows anomalous spikes in misclassifications centred at about 6,000 and 17,000 messages. SA-Nolearn's ham misclassification fraction shows no significant slope over the eight-month interval.

All learning filters show a reduction in both ham and spam misclassification fractions as more messages are processed, though not all reductions are large or significant. In particular, confidence intervals for the ham misclassification odds ratio are very large, due to the fact that the curve is fitted to few points – of the order of ten for the better-performing runs. Subject to this caveat, The plotted curves show a good fit between piecewise approximation and logistic regression. Possible exceptions are DSPAM, SA-Standard, and SA-Unsupervised. DSPAM's spam misclassification curve, shown in figure 16, has a piecewise approximation that appears to be more concave than the regression curve. SA-Standard and

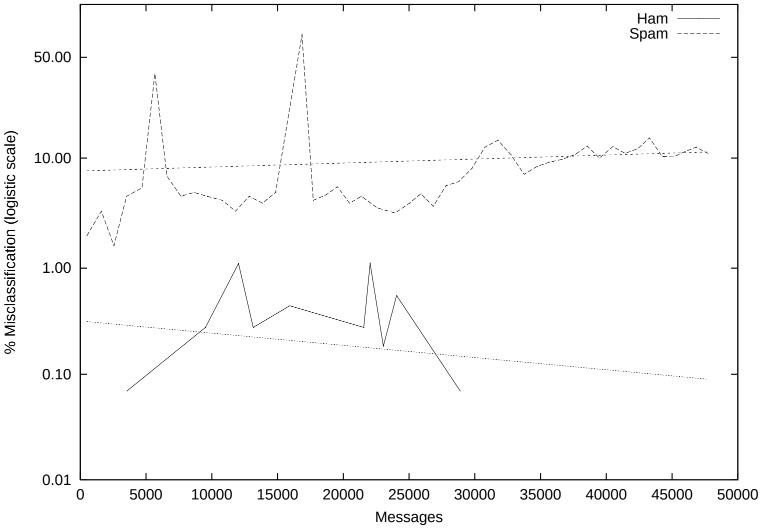

Fig. 8. SA-Nolearn Spam Misclassification as a function of Messages Processed

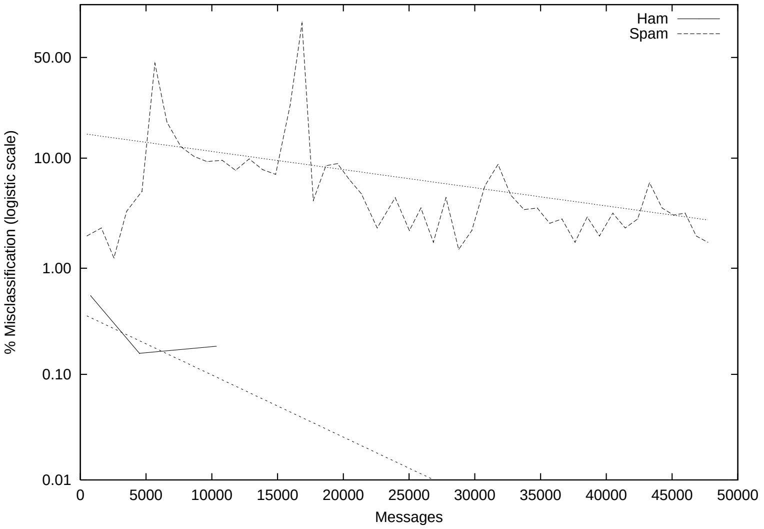

Fig. 9. SA-Standard Misclassification as a function of Messages Processed

SA-Unsupervised (figures 9 and 11) both indicate substantially lower spam misclassification rates prior to the virus-induced anomaly at message 6,000, followed by consistent improvement notwithstanding the backscatter anomaly at message 17,000. We observe that the initial misclassification fraction of a number of systems is substantially better than the final misclassification fraction of others.

For completeness, we included SA-Human in our analysis. SA-Human's ham mis-

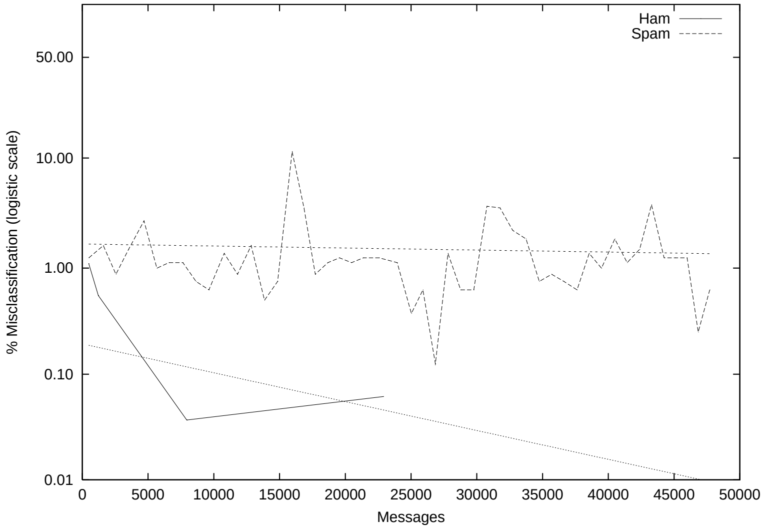

Fig. 10. SA-Supervised Misclassification as a function of Messages Processed

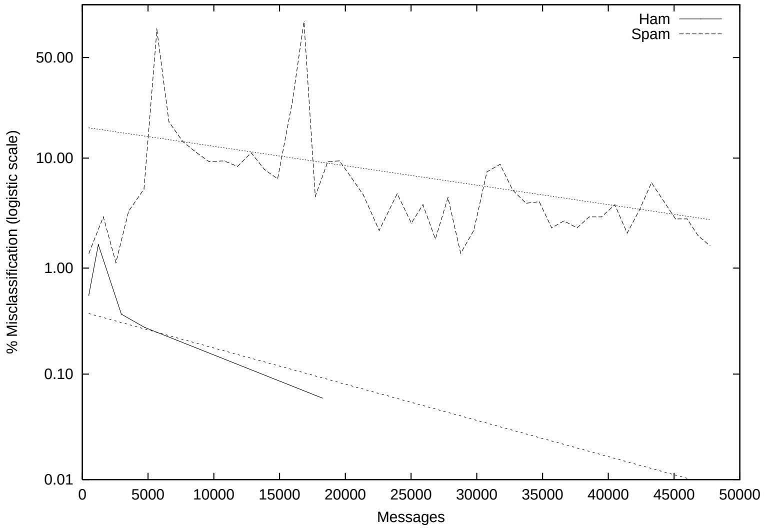

Fig. 11. SA-Unsupervised Misclassification as a function of Messages Processed

classification fraction shows a large significant increase with a huge confidence interval [odds ratio $54\,(2, 1222)$], indicating that this measurement is unstable, rather than that X suffered some degeneration in discriminatory ability. Further investigation reveals that the positive odds ratio may be accounted for entirely by three automated (but legitimate) messages received the same day from the same source. SA-Human's apparent decrease in spam misclassification may also be accounted for

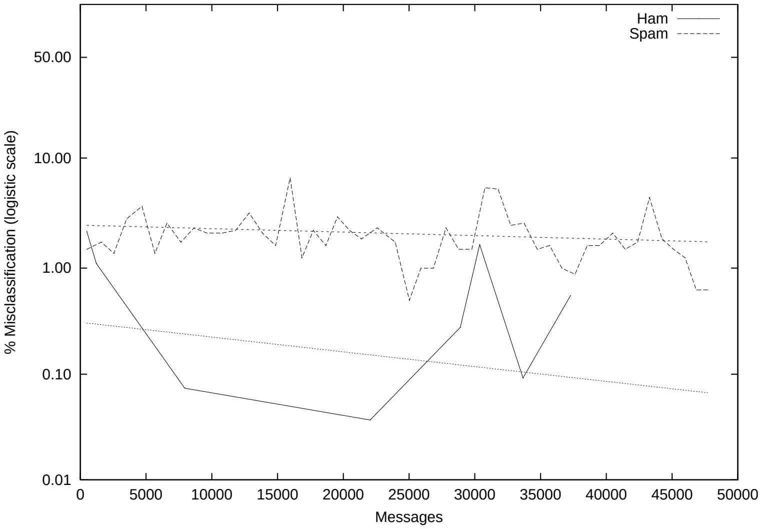

Fig. 12. SA-Bayes Misclassification as a function of Messages Processed

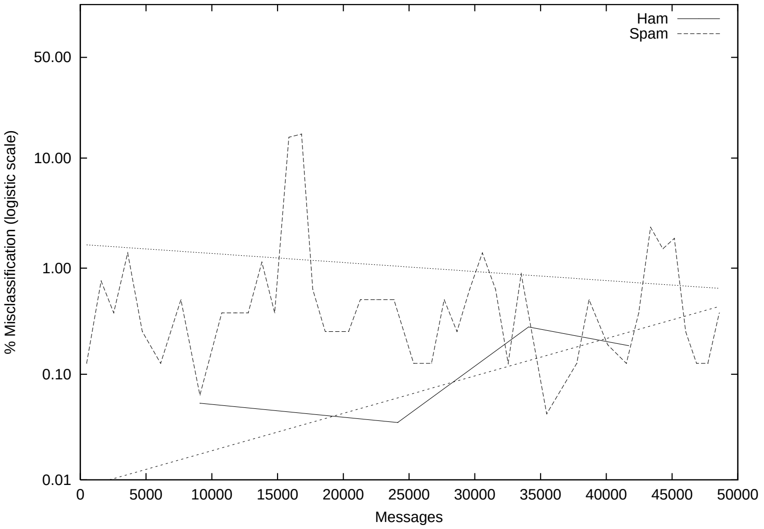

Fig. 13. SA-Human Misclassification as a function of Messages Processed

by the anomalous spike at 17,000 messages.

Misclassification by Genre

In the course of examining the misclassified messages, we identified several message genres that we suspect might be associated with the filters' performance. Ham messages were classified into seven genres:

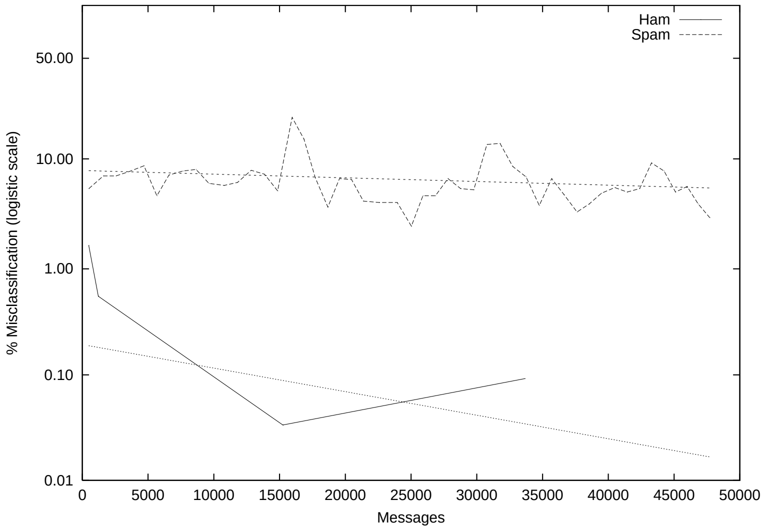

Fig. 14. Bogofilter Misclassification as a function of Messages Processed

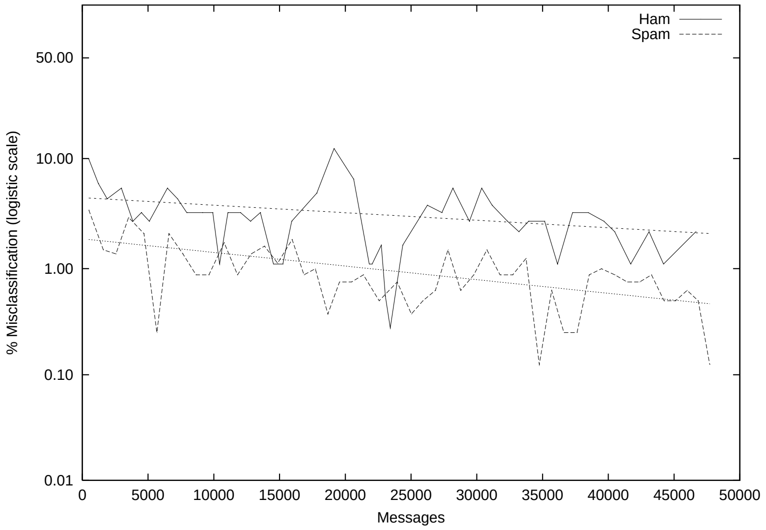

Fig. 15. CRM114 Misclassification as a function of Messages Processed

(1) *Advertising.* Messages from companies or organizations having a relationship with the recipient.

(2) *Cold Call.* Messages from individuals with whom X had no prior correspondence or relationship.

(3) *Delivery.* Messages from an email server pertaining to the delivery of an email message.

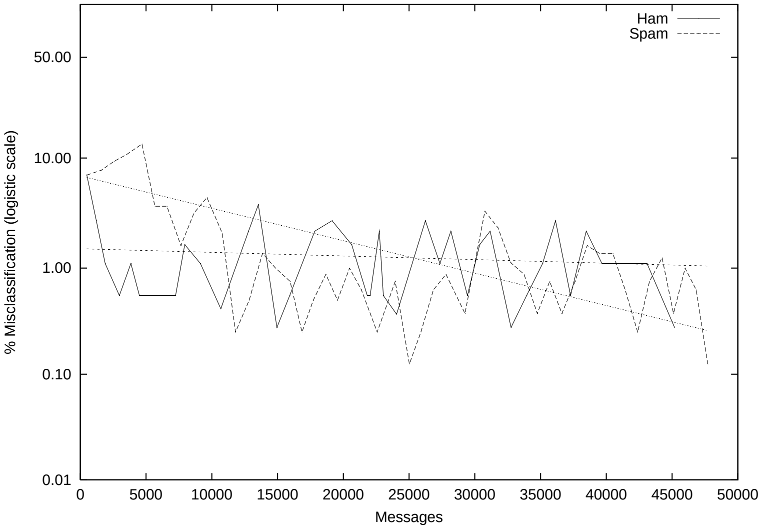

Fig. 16. DSPAM Misclassification as a function of Messages Processed

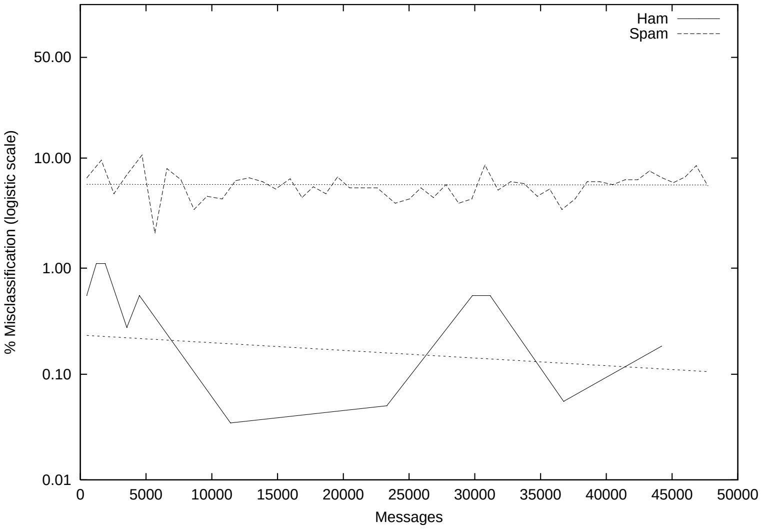

Fig. 17. SpamBayes Misclassification as a function of Messages Processed

(4) *List.* Mailing list messages, broadly defined. This genre includes automated mailing lists, service messages from mailing lists, and ad hoc messages consisting of general information copied to a large number of recipients.

(5) *News.* News clipping and digest services to which X is subscribed.

(6) *Personal.* Mail specifically addressed to X by an individual; the equivalent of *first class mail.*

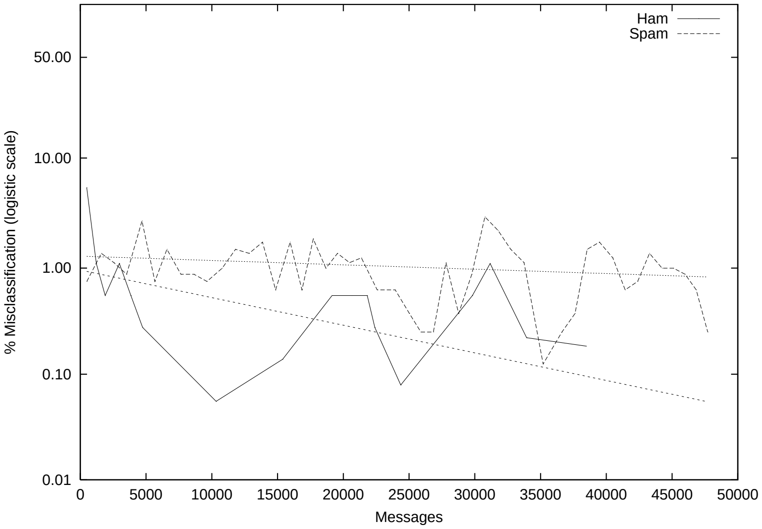

Fig. 18. SpamProbe Misclassification as a function of Messages *Processed*

Filter	Advertising	Cold Call	Delivery	List	News	Personal	Transaction	Total
SA-Standard	4	2	0	0	0	0	0	6
SA-Super	1	0	0	1	1	0	3	6
Bogofilter	1	0	0	2	1	0	3	7
SA-Human	0	0	0	3	4	0	1	8
SA-Unsuper	5	0	0	1	0	1	3	10
SA-Bayes	1	0	0	4	1	1	8	15
SpamBayes	1	0	2	5	1	3	3	15
SA-Nolearn	1	0	4	0	3	9	0	17
SpamProbe	3	2	4	5	1	8	8	31
DSPAM	15	5	9	28	6	35	18	116
CRM-114	7	15	13	78	10	135	37	295
Incoming Ham	0%	1%	17%	13%	14%	51%	4%	9038

Table IX. Ham Misclassification by Genre

(7) *Transaction.* Responses to electronic internet transactions, such as receipts, travel itineraries, shipping information, passwords, acknowledgements, or status information.

Spam messages were classified into five genres:

(1) *Advertising.* Messages sent indiscriminately to X aimed at acquiring some or all of X's wealth.

(2) *Backscatter.* Delivery messages from a third-party server, rejecting a message not sent by X, but forged to appear to have been sent by X. These messages

Filter	Advertising	Backscatter	Demographic	Targeted	Virus	Total
CRM-114	72%	8%	12%	4%	4%	397
SA-Human	14%	66%	10%	7%	4%	413
Spamprobe	48%	17%	17%	7%	12%	421
SA-Super	28%	36%	22%	5%	9%	605
DSPAM	58%	8%	17%	3%	14%	791
SA-Bayes	45%	19%	17%	8%	11%	840
SpamBayes	50%	16%	25%	7%	2%	2348
Bogofilter	68%	14%	10%	2%	6%	2656
SA-Standard	17%	29%	5%	0%	49%	2999
SA-Unsupervised	9%	31%	7%	1%	52%	3246
SA-Nolearn	51%	24%	5%	0%	20%	3802
Incoming Spam	92%	1%	0%	0%	8%	40048

Table X. Spam Misclassification by Genre

are deemed to be spam (as opposed to Delivery ham messages) because they are a direct consequence of spam.

(3) *Demographic.* Advertising messages for goods and services of marginal value sent to a specific demographic group to which X belongs.

(4) *Targeted.* Messages addressed to X for no reason other than X's membership in a broad identifiable group (profession, geographic location, appearance on a subject-related web-page, etc.).

(5) *Virus.* Messages that contain malware.

Table IX shows the number of misclassified ham messages, by genre, for each filter. Also shown is an estimate of the proportion of all ham represented by each genre. Four of the runs have no *personal* misclassifications, a much lower fraction than would be suggested by the fact that this genre comprises 51% of all ham. At the other end of the spectrum, CRM-114 misclassifies 135 *personal* ham messages, or about 3% of all such messages. DSPAM also misclassifies a high number of *personal* messages: 35, or about 0.75% of the total.

In general, *advertising, cold call*, and *delivery* messages each represent a small proportion of overall ham and a disproportionately large number of misclassifications. *Personal* messages represent disproportionately few misclassifications, while *transaction, list,* and *news* fall in between.

Table X shows the estimated fraction of misclassified spam messages, by genre, for each filter, as well as the fraction of all spam represented by each genre. The vast majority of spam messages are *advertising,* with *backscatter* representing a mere 1%. Yet nearly as many *backscatter* messages are misclassified. In particular, we note that SA-Human and SA-Super misclassify a fraction of backscatter messages approaching or exceeding 50%. Three-fifths of all of SA-Human's misclassifications are attributable to misclassified *backscatter.* The reason for this is that X was overwhelmed by the burst of *backscatter* occurring at 17,000, and skipped over many of these messages without recording a judgement[4].

Discussion and Comparison

Evaluation Measures

Although widely reported, *accuracy* has little value in evaluating and comparing spam filters. The consequences of ham and spam misclassification are materially different, while measurements of accuracy conflate them. The computation of accuracy depends directly on the ratio of ham to spam messages in the incoming email, and also on the threshold parameter used by the filter to transform scores into judgements. For a given filter, the problem of optimizing accuracy reduces to the decision-theoretic problem of picking the best threshold [Lewis 1995] for the anticipated ham-to-spam ratio ($hs = \frac{a+c}{b+d}$; a, b, c, d from table I). Tables III and IV include *overall misclassification fraction (1-accuracy)* which reflect influence of the systems' default threshold parameters. Every system in this study, had its threshold been set to optimize accuracy[5], would have yielded an unacceptably high level of ham misclassification (see table XI).

Filter	Ham Misc.	Spam Misc.	Overall Misc.
SpamProbe	0.94	0.44	0.54
SA-Super	1.62	0.49	0.69
SA-Bayes	1.97	0.40	0.69
Bogofilter	1.25	0.59	0.71
SpamBayes	1.60	0.61	0.79
DSPAM	1.90	1.03	1.19
CRM-114	3.95	0.75	1.34
SA-Nolearn	5.53	2.77	3.28
SA-Standard	2.98	3.88	3.71
SA-Unsuper	10.64	1.67	3.32

Table XI. Effect of Optimizing Accuracy

Hidalgo [2002] discusses the use of cost-sensitive evaluation to mitigate these difficulties:

> The main problem in the literature on [spam] cost-sensitive categorization is that the [ham-spam cost ratios] used do not correspond to real world conditions, unknown and highly variable. No evidence supports that classifying a legitimate message as [spam] is 9 nor 999 times worse than the opposite mistake.

This criticism – dependence on highly variable external factors, arbitrary filter parameters, and arbitrary evaluation weights – applies to a large class of combined evaluation measures (cf. Sebastiani[2002]). To this criticism we add a note of caution with respect to the statistical power of filter evaluations. Ham misclassification

[4] X subsequently deployed an ad-hoc filter to identify backscatter messages and to record a judgement automatically.

[5] The results presented here are the result of a hypothetical run for which the optimal threshold was known in advance. Lewis discusses automatic methods of adjusting the threshold so as to optimize error rate (i.e. $1 - accuracy$) and other measures.

rates for good filters are exceptionally low, amounting to only a handful of messages in our sample of nearly 50,000. These rates are even lower when stratified by genre, often yielding 0 occurrences (e.g. four of the runs misclassified *no* personal email messages). The statistical uncertainty due to these small numbers will dominate any weighted score, potentially masking significant differences in spam misclassification rates for filters with comparable ham misclassification rates.

Hidalgo suggests the use of ROC curves, originally from signal detection theory and used extensively in medical testing, as better capturing the important aspects of spam filter performance. In the event that the ROC curve for one filter is uniformly above that of another, we may conclude that there is a parameter setting such that its performance exceeds the other for any combination of external factors and evaluation weights. The area under the ROC curve serves to quantify this difference and, perhaps surprisingly, represents a meaningful quantity: the probability that a random spam message will receive a higher score than a random ham message. In the event that the ROC curves intersect, one may consider the area under only a subset, the *normal operating region.* For a spam filter, this operating region would likely be the fragment of the curve above the range of acceptable ham misclassification fraction values.

Tuttle et al [2004] present spam filter effectiveness using a tabular representation of an ROC curve: hm vs. $(1-sm)$. Further, they choose 1% hm as a proxy for the normal operating region and report sm at this value. More broadly, ROC-based evaluation for machine learning and information retrieval is of current interest. We found that ROC analysis provided us with valuable insight to our results, complementing but not obviating distinct ham and spam misclassification analyses. With one inversion (SA-Standard vs. SA-Unsupervised) ROCAC values agreed with our subjective ranking of the systems. The ROC curves for these two runs intersect; SA-Standard demonstrates superior performance within the normal operating region (small hm) while SA-Unsupervised overtakes it for large hm.

Like the measures described above, *recall, precision,* and *precision-recall* curves evaluate the tension between ham and spam classification performance. Precision and recall originate with information retrieval, in which the objective is to discover relevant documents from a collection. The measures are asymmetric, predicated on the general assumption that there are many fewer relevant than non-relevant documents in the collection. *Recall* is the fraction of all relevant documents retrieved by the system; *precision* is the fraction of retrieved documents that are relevant. Within the context of spam classification, it is necessary to consider either the ham or the spam messages as relevant, and the others as not relevant. This labelling is arbitrary, but must be identified. Ham precision ($hp = \frac{a}{a+b}$) and ham recall ($hr = \frac{a}{a+c}$), in which ham messages are deemed to be relevant, have perhaps the more intuitive meaning within the context of spam filtering. The complementary measures are spam precision ($sp = \frac{d}{c+d}$) and spam recall ($sr = \frac{d}{b+d}$).

Ham recall is the same thing as ham accuracy ($1 - hm$). Spam recall is the same thing as spam accuracy ($1 - sm$). But these two measures are not used as a pair in information retrieval evaluation, which assumes a consistent labelling of relevant and non-relevant documents. Instead, ham precision and ham recall (or spam precision and spam recall) are used together[6]. Ham precision depends on sm but depends also on hr and hs: $hp = \frac{r}{1+r}$ where $r = hs \cdot \frac{hr}{sm}$. r, the ratio of ham to spam delivered to the mail file, is proportional to the incoming ham-spam ratio. Ham precision simply recasts r as a fraction as opposed to a ratio. Thus we conclude that precision and recall, taken as a pair, exhibit the same essential shortcoming as accuracy. *Average precision*, the analog of ROCAC, is similarly influenced by hs.

The medical diagnostic testing literature (cf. [Rothman and Greenland 1998]) casts the problem as one of testing a population of patients for a particular disease. The test offers a diagnosis of *diseased* or *disease-free.* To apply diagnostic testing metaphors to spam, we (arbitrarily but with some support from connotation) label spam to be diseased and ham to be disease-free. The variables a, b, c, d from table I are known as *true negatives, false negatives, false positives,* and *true positives* respectively. Ham accuracy is *specificity,* while spam accuracy is *sensitivity* [7]. The literature also discusses *negative predictive value* and *positive predictive value.* Negative predictive value is the probability that a random patient, on receiving a negative diagnosis, is really disease-free. Positive predictive value is the probability that a random patient, on receiving a positive diagnosis, is really diseased. Predictive values use Bayesian inference to combine two distinct estimates: specificity (or sensitivity), which is a property of the diagnostic test, and *prevalence,* which is a property of the population being tested. Negative predictive value is exactly ham precision as described above, while positive predictive value is spam precision.

Precision, like predictive value, is very useful in predicting the in situ performance of a filter. We believe it should, like predictive value, be computed post-hoc by combining separate measurements of filter performance and incoming ham-spam ratio, rather than used as a fundamental measure of filter performance.

Previous Studies

Sahami et al [1998] conducted an early study that indicated the utility of Bayesian classifiers for spam filtering. One experiment used a corpus of 1789 actual e-mail messages (11.8% ham; 88.2% spam), split chronologically into 1538 training messages and 251 test messages. Both ham and spam precision/recall curves were calculated. The best-performing system achieved ham recall of 100% and spam recall of 98.3%. From these values and the test sample size we may compute $hm = 0\% \, (0\% - 9.5\%)$ and $sm = 1.7\% \, (0.4\% - 4.6\%)$. A second experiment classified the spam component of a similar corpus into two genres: *pornographic* and *non-pornographic.* The genres were used in an evaluation of ternary classification, but not for a stratified evaluation of binary classification. A third experiment most closely resembles those which we conducted: an individual's email messages were captured over one year, classified manually, and used as training data. The filter was applied to further week's email received by the same individual. The resulting classification table, shown in table XII, demonstrates $hm = 1.7\% \, (0.3\% - 4.9\%)$,

[6] The information retrieval literature defines *fallout*, which in this context would be the same as sm and therefore equivalent to spam recall. Recent evaluations often report precision and recall; rarely fallout.

[7] To our knowledge, no analog of overall accuracy exists in medical diagnostic testing.

$sm = 20\% \, (9.6\% - 34.6\%)$. Sahami et al further examine the three misclassified ham messages, observing two to be newsletter messages and one to be a personal message that includes a spam message as an attachment.

Contingency table			% Ham Misc.	% Spam Misc.	% Misc.
	ham	spam			
ham	174	9	1.7% (0.3%-4.9%)	20% (9.6%-34.6%)	5.41 (2.82-9.25)
spam	3	36			

Table XII. Sahami et al

Androutsopoulos et al [2000] contribute *Ling Spam*, a publicly available corpus for evaluating spam filters. Ling Spam consists of 2412 ham messages from a mailing list, combined with 481 spam messages received by an individual. Ling Spam provides a common evaluation suite for autonomous studies, subject to the caveat that it does not include chronological or other header information that might disclose the source of the messages, and the ham messages may not resemble typical email.

In their experiments, Androutsopoulos et al use *ten-fold cross validation* [Kohavi 1995], in which a corpus of size n is divided randomly into 10 subsets of size $\frac{n}{10}$. Each subset is used as the test set with the remaining nine subsets combined for training. The combined sets of results are treated as one set of n Bernoulli trials. Spam precision, spam recall, and weighted accuracy are reported for three different values of the parameter λ which was used both as a threshold value and evaluation weight. The authors suggest *total cost ratio (TCR)* as a measure to distinguish the weighted accuracy of a filter from that of a simplistic baseline approach with classifies every message as ham. Table XIII recasts these results in terms of contingency tables and misclassification probability estimates.

λ	Contingency table		% Ham Misc.	% Spam Misc.	% Overall Misc.
1	2410	83	0.08 (0.01-0.30)	17.3 (14.0-20.9)	2.94 (2.35-3.62)
	2	398			
9	2410	104	0.08 (0.01-0.30)	21.6 (18.0-25.6)	3.67 (3.01-4.41)
	2	377			
999	2412	168	0 (0-0.12)	34.9 (30.7-39.4)	5.81 (4.98-6.72)
	0	313			

Table XIII. Androustopoulos et al

Tuttle et al [2004] evaluate three common machine-learning algorithms – naive Bayesian classifiers, support vector machines, and boosted decision trees – within the context of an enterprise mail system. They deployed a novel architecture to capture email messages and judgements from several users, keeping this information private and under the control of the users to whom the messages belonged. Test runs were "pushed" to the users' corpora, and only statistics were reported back to the central system. Seven users participated in the study, and corpora consisting of up to 800 messages per user were subject to ten-fold cross-validation. Results for each of the seven corpora were computed and the mean of these results was reported. The primary experiment used individual corpora with 400 messages each, approximately 62% spam, and reported piece-wise ROC curves (see table XIV) for $hm \in \{0.0\%, 0.5\%, 1.0\%, 2.0\%, 5.0\%\}$. Other experiments fixed $hm = 1.0\%$ as a proxy for the operating range. Confidence intervals are not reported but we note the overall sample size of 2800 suggests that they would be comparable in magnitude to those for Sahami et al and Androutsopoulos et al. Tuttle et al perform a 2-factor analysis of variance and conclude that there is a significant difference in results among the seven corpora, but not among the three filters.

%Ham Misc.	% Spam Misc.		
	Naive Bayes	SVM	AdaBoost
0.0	5.9	6.2	10.5
0.5	4.1	4.4	8.1
1.0	2.8	3.5	5.6
2.0	2.0	2.2	2.6
5.0	1.1	0.5	1.3

Table XIV. Tuttle

Holden [2004] conducted five experiments to compare seven open-source spam filters, including Bogofilter 0.13.7.2, SpamProbe 0.8b, and Spamassassin 2.55 (with Bayes filter disabled). As training and test corpora, Holden used various combinations of personal email received during July and the first week of August 2003. The most comprehensive experiment uses 1270 messages (200 ham; 1070 spam). Ten-fold cross validation is used to compute spam precision and spam recall separately for each of the runs; the mean and standard deviation over the ten runs are reported for each measure. In addition, the weighted accuracy and TCR are computed, for various λ, considering all ten runs to be one set of Bernoulli trials. From these reported summary values, we reproduced the contingency tables[8] and estimated hm and sm, reported in table XV. Holden provides a qualitative description of the few misclassified ham messages, observing a preponderance of messages like welcome advertising, news clippings, mailing lists, etc.

Yerazunis [2004b] reports the results of applying various configurations of the CRM-114 filter to the Spamassassin Public Mail Corpus [spamassassin.org 2003] – 4,147 messages (2750 ham; 1397 spam) hand-selected from various email sources. Evaluation was based on a method similar to ten-fold cross validation. Ten random permutations of the corpus were generated and the filter was run in supervised configuration on each permutation. Accuracy was computed using the combined results from only the last 500 messages in each run, for a total of 5000 messages. We note that these 5000 results do not constitute Bernoulli trials, as they are not independent. In particular, we expect only $1 - (1 - \frac{1}{4743})^{5000}$ of the messages in the corpus (65% or 3083 messages) to be represented in these results. That is, 1917 of the results will be based on applying the filter to a repeated messages. We would expect the results for a repeated message to be strongly correlated and

[8] Mean recall and mean precision as a pair yield insufficient information to reproduce the contingency table. While the mean of recalls for the subsets equals overall recall, the mean of precisions does not equal overall precision. Overall accuracy, in place of mean precision, suffices.

Filter	Contingency		% Ham Misc.	% Spam Misc.	% Misc.
Bayesian Mail Filter	188	12	6.00 (3.14-10.25)	1.12 (0.58-1.95)	1.89 (1.21-2.80)
	12	1058			
Bogofilter	199	60	0.50 (0.01-2.75)	5.61 (4.31-7.16)	4.80 (3.69-6.13)
	1	1010			
Quick Spam Filter	165	112	17.5 (12.5-23.5)	10.5 (8.7-12.5)	11.6 (9.9-13.5)
	35	958			
SPASTIC	139	577	30.5 (24.2-37.4)	53.9 (50.9-56.9)	50.2 (47.5-53.0)
	61	493			
SpamAssassin	199	214	0.50 (0.01-2.75)	20.0 (17.6-22.5)	16.9 (14.9-19.1)
	1	856			
SpamProbe	189	11	5.50 (2.78-9.63)	1.03 (0.51-1.83)	1.73 (1.09-2.61)
	11	1059			
dbacl	194	223	3.00 (1.11-6.42)	20.8 (18.4-23.4)	18.0 (15.9-20.3)
	6	847			

Table XV. Holden

therefore contribute little new information. Weighting them equally affords them undue influence in the overall score. This consideration introduces no bias, but does affect the effective sample size. We use a first-order approximation of 3083 for the sample size in reporting the misclassification probability estimates in table XVI.

CRM-114 Configuration	% Misc.
Train on Everything	2.98 (2.41-3.65)
Train on Error	1.39 (1.01-1.87)
Train until No Error	1.07 (0.74-1.50)
Pure Bayesian	1.85 (1.40-2.39)
Peak Window Value	1.59 (1.18-2.10)
Token Sequence Sensitive	1.56 (1.15-2.06)
Token Grab Bag	1.43 (1.04-1.91)
Sparse Binary Polynomial Hash	1.39 (1.01-1.87)
Markovian Matching	1.14 (0.79-1.58)

Table XVI. Yerazunis - Spamassassin Corpus

Yerazunis [2004a] reports on two experiments using private email corpora. Using a corpus of approximately 3000 personal messages, he estimated the accuracy of human classification by having the same individual classify the set of messages on two distinct occasions. Ten cases of disagreement were observed between these two efforts, from which Yerazunis concludes that human accuracy is $\frac{5990}{6000} = 99.84\%$. Table XVII reports our estimated misclassification rate, based on the (far-reaching) assumption that the individual's classification efforts were independent. Dependence between the two efforts would cause the misclassification rate to be underestimated by an unknown quantity.

Yerazunis also reports the results of applying CRM-114 to private email over selected time periods since November, 2002, which are recapitulated in table XVIII. It is our understanding based on correspondence with the author that these results

Filter	Corpus Size	% Misc.
Human	3000	0.17 (0.08 - 0.31)

Table XVII. Yerazunis - Human Classification of Private Email

Interval	Size (ham,spam)	Errors	% Misc.
Nov. 1-30, 2002	5849 (3914, 1931)	4	0.07 (0.02-0.18)
Sept. 1-14, 2003	> 2500	0	0.00 (0.00-0.12)
Feb. 1 - Mar 1, 2004	8738 (4498, 4240)	1	0.01 (0.00-0.06)

Table XVIII. Yerazunis - CRM-114 Performance on Private Email

were attained using the methodology akin to cross-validation detailed above, rather than from the in situ performance of the filter.

Zdziarski [2004] reports 99.95% to 99.991% accuracy for DSPAM based on an unspecified methodology. Spamassassin, Bogofilter, and SpamBayes have been the subject of unpublished internal evaluations, the raw results of which may be found at their respective project sites. Burton [2002b] reports 99.7% accuracy for SpamProbe, but cautions:

> Finally a word of advice. An individual user evaluating different spam filters should not give too much weight to the accuracy figures quoted by the authors. Only a side by side comparison on a single corpus can be useful as a means of comparison. Different authors compute their numbers in slightly different ways on their own (unique) email corpus. A user looking to install a filter should pick one that they are comfortable installing and be happy with the results. Any of the filters based on Paul [Graham]'s technique should be extremely accurate for most people.

Conclusions

Supervised spam filters are effective tools for attenuating spam. The best-performing filters reduced the volume of incoming spam from about 150 messages per day to about 2 messages per day. The corresponding risk of mail loss, while minimal, is difficult to quantify. The best-performing filters misclassified a handful of spam messages early in the test suite; none within the second half (25,000 messages). A larger study will be necessary to distinguish the asymptotic probability of ham misclassification from zero.

Most misclassified ham messages are advertising, news digests, mailing list messages, or the results of electronic transactions. From this observation, and the fact that such messages represent a small fraction of incoming mail, we may conclude that the filters find them more difficult to classify. On the other hand, the small number of misclassifications suggests that the filter rapidly learns the characteristics of each advertiser, news service, mailing list, or on-line service from which the recipient wishes to receive messages. We might also conjecture that these misclassifications are more likely to occur soon after subscribing to the particular service (or soon after starting to use the filter), a time at which the user would be more likely to notice, should the message go astray, and retrieve it from the spam file. In contrast, the best filters misclassified no personal messages, and no delivery error messages, which comprise the largest and most critical fraction of ham.

A supervised filter contributes significantly to the effectiveness of Spamassassin's static component, as measured by both ham and spam misclassification probabilities. Two unsupervised configurations also improved the static component, but by a smaller margin. The supervised filter alone performed better than than the static rules alone, but not as well as the combination of the two.

The choice of threshold parameters dominates the observed differences in performance among the four filters implementing methods derived from Graham's and Robinson's proposals. Each shows a different tradeoff between ham accuracy and spam accuracy. ROC analysis shows that the differences not accountable to threshold setting, if any, are small and observable only when the ham misclassification probability is low (i.e. $hm < 0.1\%$).

CRM-114 and DSPAM exhibit substantially inferior performance to the other filters, regardless of threshold setting. Both exhibit sustained learning throughout the email stream, leading us to conjecture that their performance might asymptotically approach that of the other filters. From a practical standpoint, this learning rate would be too slow for personal email filtering as it would take several years at the observed rate to achieve the same misclassification rates as the other systems. CRM-114 was designed to be used in a *train on error* configuration, and does not self-train. This configuration could account for a slow learning rate as CRM-114 avails itself of the information in only about 1,000 of the 50,000 test messages. In an effort to ensure that we had not misinterpreted the installation instructions, we ran CRM-114 in a *train-on-everything* configuration and, as predicted by the author, the result was degraded.

Spam filter designers should incorporate interfaces making them amenable for testing and deployment in the supervised configuration (figure 4). We propose the three interface functions used in algorithm 1 – *filterinit, filtereval,* and *filtertrain* – as a standardized interface. Systems that self-train should provide an option to self-train on everything (subject to correction via $filtertrain$) as in algorithm 2.

Ham and spam misclassification proportions should be reported separately. Accuracy, weighted accuracy, and precision should be avoided as primary evaluation measures as they are excessively influenced by threshold parameter setting and the ham-spam ratio of incoming mail. ROC curves provide valuable insight into the tradeoff between ham and spam accuracy. Area under the ROC curve provides a meaningful overall effectiveness measure, but does not replace separate ham and spam misclassification estimates. Each case of ham misclassification should be examined to ascertain its cause and potential impact.

Caution should be exercised in treating ham misclassification as a simple proportion. Extremely large samples would be needed to estimate it with any degree of statistical confidence, and even so, it is not clear what effect differences in proportion would have on the overall probability of catastrophic loss. The use of a filter may mitigate rather than exacerbate this risk, owing to the reduction in classification effort required of the user. We advance the proposition that, at the misclassification rates demonstrated here, the end-to-end risk of loss is dominated by human factors and exceptional events, and is comparable to that of other communication media.

REFERENCES

AGRESTI, A. 1996. *An Introduction to Categorical Data Analysis*. Wiley, New York.

ANDROUTSOPOULOS, I., KOUTSIAS, J., CHANDRINOS, K., PALIOURAS, G., AND SPYROPOULOS, C. 2000. An evaluation of naive bayesian anti-spam filtering.

BURTON, B. 2002a. Spamprobe - a fast bayesian spam filter. http://spamprobe.sourceforge.net.

BURTON, B. 2002b. Spamprobe - bayesian spam filtering tweaks. http://spamprobe.sourceforge.net/paper.html.

GRAHAM, P. 2002. A plan for spam. http://www.paulgraham.com/spam.html.

GRAHAM, P. 2004. Better bayesian filtering. http://www.paulgraham.com/better.html.

GRAHAM-CUMMING, J. 2004. Popfile. http://popfile.sourceforge.net/.

HIDALGO, J. M. G. 2002. Evaluating cost-sensitive unsolicited bulk email categorization. In *Proceedings of SAC-02, 17th ACM Symposium on Applied Computing*. Madrid, ES, 615–620.

HOLDEN, S. 2004. Spam filters. http://http://freshmeat.net/articles/view/964/.

HOLM, S. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics 6*, 65–70.

KOHAVI, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*. 1137–1145.

LEWIS, D. D. 1995. Evaluating and optimizing autonomous text classification systems. In *SIGIR'95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle, Washington, USA, July 9-13, 1995 (Special Issue of the SIGIR Forum)*, E. A. Fox, P. Ingwersen, and R. Fidel, Eds. ACM Press, 246–254.

LOUIS, G. 2004. Bogofilter and repetitive training. http://www.bgl.nu/bogofilter/reptrain.html.

MOZILLA.ORG. 2004. Mozilla. http://www.mozilla.org.

PARK, S. H., GOO, J. M., AND JO, C.-H. 2004. Receiver Operating Characteristic (ROC) curve: Practical review for radiologists. *Korean Journal of Radiology 5,* 1, 11–18.

PETERS, T. 2004. Spambayes: Bayesian anti-spam classifier in python. http://spambayes.sourceforge.net/.

RAYMOND, E. S. 2004. Bogo filter. http://bogofilter.sourceforge.net/.

ROBINSON, G. 2003. A statistical approach to the spam problem. *Linux Journal* 107.

ROBINSON, G. 2004. Gary robinson's spam rants. http://radio.weblogs.com/0101454/categories/spam/.

ROTHMAN, K. J. AND GREENLAND, S. 1998. *Modern Epidemiology*. Lippinscott Williams and Wilkins.

SAHAMI, M., DUMAIS, S., HECKERMAN, D., AND HORVITZ, E. 1998. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, Madison, Wisconsin.

SEBASTIANI, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys 34,* 1, 1–47.

SPAMASSASSIN.ORG. 2003. The spamassassin public mail corpus. http://www.spamassassin.org/publiccorpus.

SPAMASSASSIN.ORG. 2004. Welcome to spamassassin. http://www.spamassassin.org.

STREINER, N. 1986. *PDQ Statistics*. B.C. Decker Inc.

TUTTLE, A., MILIOS, E., AND KALYANIWALLA, N. 2004. An evaluation of machine learning techniques for enterprise spam filters. Tech. Rep. CS-2004-03, Dalhousie University, Halifax, NS.

YERAZUNIS, W. S. 2004a. CRM114 - the controllable regex mutilator. http://crm114.sourceforge.net/.

YERAZUNIS, W. S. 2004b. The spam-filtering accuracy plateau at 99.9% accuracy and how to get past it. In *2004 MIT Spam Conference*.

ZDZIARSKI, J. A. 2004. The DSPAM project. http://www.nuclearelephant.com/projects/dspam/.