A Close Look at Current Approaches in Spam Filtering

Alexander K. Seewald
Austrian Research Institute for Artificial Intelligence
Freyung 6/6, A-1010 Vienna, Austria
alexsee@oefai.at, alex@seewald.at

Abstract

In this paper we present an applied machine learning approach to spam filtering, SA-Train. We compare SA-Train, which runs repeated steps of rule score learning via linear Support Vector Machine and bayesian learning and is based on SpamAssassin, to a state-of-the-art bayesian phrase learner, CRM114, as well as to a simpler bayesian learner, SpamBayes. We also compare our approach to ready-to-use systems such as SpamAssassin's default settings without bayesian model, Symantec BrightMail and Google Mail. Surprisingly, the relatively simple SpamBayes learner turns out to be the best learner in this setting, and its corresponding model degrades less over time than two other ready-to-use models, giving almost constant performance for four months without additional training.

1 Introduction

Spam[1] has become a problem of global impact. For example, according to a study undertaken for the European Commission, Internet subscribers worldwide are wasting an estimated 10 billion Euro per year just in connection costs due to Spam [9]. Economic impact is only part of the problem – waste of time, resources and the gradual erosion of trust in EMail communications should also be considered. Within the scientific community these effects are felt strongly. For example, at our institute the overall proportion of Spam now exceeds 90%. For every nonspam mail, we receive around 15-20 spam mails – more than 300 spams per day per user in the worst cases.

Several approaches exist to deal with spam [5]. Filtering approaches based on simple message features such as the occurrency of certain words (e.g. Viagra) are most widely used. In fact, most email clients already allow their users to manually build email filters. However, the manual approach is time-consuming and much expertise is needed to create useful filters from scratch. Also, such filters need to be maintained and updated as they are an obvious target for spammers to attack.[2]

Another option for filtering is to collect samples of spam and ham (i.e. nonspam) to train a learning system. This has been proposed e.g. by [4] and works surprisingly well even with simple statistical classifiers such as Naive Bayes. Most state-of-the-art spam filters now include learning systems, e.g. SpamBayes (`spambayes.org`), CRM114 (`crm114.sourceforge.net`) and SpamAssassin (`spamassassin.org`). However, the effort in collecting and cleaning samples is significant, especially when this has to be done for each user.[3]

There are also ready-to-use systems which do not need to be initialized for a specific user, but work in a user-independent way. These use a variety of techniques, e.g. Spam traps (honey pots) which collect spams sent to specifically set up EMail adresses, manual rules, bayesian filtering, or non-disclosed techniques. The performance of these systems is either not known or has not been independently verified.

[1]We define Spam as not only Unsolicited Bulk Email, but also Phishing mails (which aim to collect credit-card or personal information for fraud and identitiy theft), virus-infected mails, Mail Delivery Errors for mails which originated elsewere but sent with a fake From address, etc. So we have a quite broad Spam definition.

[2]E.g. 600,426,974,379,824,381,952 ways to spell viagra, `http://cockeyed.com/lessons/viagra/viagra.html`

[3]We are currently addressing this problem by training models based on pooled samples from seven users at our institute, and checking whether the trained model can generalize to the other users. Since pooling samples tends to improve performance, all learning models reported here use pooled samples.

Table 1: A set of classification decisions for mails belonging to either ham (+) or spam (-) category within a specific mailbox can be succintly represented by a contingency table. The columns represent the prediction of the system while the rows represents the true class. a-d represent the number of mails in the respective category.

		Predicted Class	
		spam (-)	ham (+)
True	spam (-)	a	b
Class	ham (+)	c	d

The main motivation for this paper was to compare state-of-the-art learning and ready-to-use approaches, and check which one is most appropriate for our institute. We have also been investigating a combination of repeated learning of rule scores (via linear Support Vector Machine[6]) and bayesian learning. This approach has been tested since June 2004 by seven colleagues at our institute and the author. Our system was found to work very well even without online training. From March 2005, it has been superseded by the latest system based on a unified model which has been trained on pooled data from all collected mailboxes. A secondary motivation was thus to check whether the approach to training we used for our latest system (similar to SA-Train) was competitive to state-of-the-art learning and ready-to-use systems.

2 Evaluation measures

$$\#N = a+b+c+d \quad \text{number of mails} \quad (1)$$

$$\#Spams = a+b \quad \text{number of true spams} \quad (2)$$

$$\#Hams = c+d \quad \text{number of true hams} \quad (3)$$

$$SH_{ratio} = \frac{\#Spams}{\#Hams} \quad \text{Spam/Ham ratio} \quad (4)$$

$$H_{err} = \frac{c}{\#Hams} \quad \text{Ham error} \quad (5)$$

$$S_{err} = \frac{b}{\#Spams} \quad \text{Spam error} \quad (6)$$

$$FP = H_{err}\frac{1}{1+SH_{ratio}} \quad \text{False Positive rate} \quad (7)$$

$$FN = S_{err}\frac{1}{1+\frac{1}{SH_{ratio}}} \quad \text{False Negative rate} \quad (8)$$

$$Err = FP + FN \quad \text{Error rate} \quad (9)$$

$$Err_c = c * FP + FN \quad \text{Cost-sensitive error} \quad (10)$$

$$Acc = 1 - Err \quad \text{Accuracy} \quad (11)$$

How to evaluate is of course at the center of such an investigation. The effectiveness of spam filtering systems is usually measured in terms of correct and wrong decisions. For simplicity, we restrict ourselves to two classes: ham (+ aka nonspam) and spam (-)[4]. For a given mailbox, the classification of a spam filtering system can then be summarized in a contingency table, see Table 1. **a** (**T**rue **N**egatives) and **d** (**T**rue **P**ositives) are the number of spam resp. ham mails which are correctly predicted by the given system. **c** (**F**alse **P**ositives) are errors where ham mails have been misclassified as spam, and **b** (**F**alse **N**egatives) are errors where spam mails have been misclassified as ham.

There are a lot of measures concerning the evaluation of spam filtering systems, but most can be computed directly from the contingency table. Some proposed measures such as *FP rate* and *FN rate* depend on the ratio of ham to spam mails and cannot be interpreted meaningfully without this information. Provided SH_{ratio} is given, comparison of different mailboxes is still possible. For the measures *Acc* and *Err* this is no longer possible as information on the relative proportion of **FPs** and **FNs** has been lost in the sum (compare formula (9)). Additionally *Err* and *Acc* give the same cost to misclassification of ham and spam mails, when for most applications the misclassification of true ham mails is far more costly.

Here, we propose two new measures, *Ham error* (H_{err}, (5)) and *Spam error* (S_{err}, (6)) which are independent of the ratio of ham to spam mails. Ham error can be interpreted as $P(\text{mail is misclassified}|\text{mail is true ham})$, i.e. the estimated probability that a true ham mail is misclassified as spam. Spam error can be similarily interpreted as $P(\text{mail is misclassified}|\text{mail is true spam})$, i.e. as the estimated probability that a true spam mail is misclassified as ham. This is a simple way to measure system performance without reference to SH_{ratio} as well as being immediately understandable. Other common measures can be computed from these two measures and the Spam/Ham ratio (SH_{ratio}) easily, see (7)-(11). (7) and (8) can also be used to compute H_{err} and S_{err} from *FP rate* and *FN rate* (plus associated SH_{ratio}) for comparison with other mailboxes.

We always report separately the spam and ham error rates for each system and mailbox – if a single accuracy value is needed, it can be approximated by formula (11) given SH_{ratio}. What is sometimes reported by ready-to-use systems is *FP* and *FN*[5] without reference to SH_{ratio}. Prediction error *Err* or accuracy *Acc* both assume equal error costs. Usually **FP** errors are far more costly than **FN** errors. We account for this by reporting cost-sensitive error Err_{1000} which weights **FP** errors a thousandfold higher than **FN** errors.

[4]For most of the ready-to-use systems, the internal score of each mail is not readily available for analysis.

[5]e.g. Symantec BrightMail reports FP=0.0001% (1:1,000,000) and FN=5% (1:20)

3 Mailboxes

The collection and especially verification of spam is time-consuming and error-prone.[6] Because of the severity of our spam problem, we were motivated to collect seven distinct mailboxes consisting of ham and spam mails from colleagues at our institute from June 2004 onwards. We have taken an effort to clean the mailboxes, inspecting every training set error of an earlier learning system to see whether the mail was assigned the wrong category. Despite all our efforts we still found some wrongly assigned ham mails during the course of our experiments, courtesy of the ready-to-use system SA-Default and BrightMail: 0.22% spams in ham mailbox #1, 0.06% spams in ham mailbox #2 and 0.03% spams in ham mailbox #7. An overview over the mailboxes, including date ranges for ham and spam mails, is in Table 2.

We created mailboxes #1-#7 by merging mails predicted as spam by SpamAssassin 2.63[7] with spams collected and submitted by the users themselves. SpamAssassin markup and headers were removed. The ham mails, on the other hand, were collected from past ham mails from the user's mail archive. We removed all mails which were sent by the user himself, since these are another spamfilter's problem. Ham mails are likely to change less over time than spam mails, so a combination of recent spam mails and stored ham mails seems a plausible way to get to a working system fast. This is also the reason why the SH_{ratio} of these mailboxes differs from what the user experiences in his daily life.

On the other hand, Mailbox #8 is the set of all ham and spam mails received by the author between 4th of October 2004 and 13th February 2005. These are a realistic snapshot of the mails received at the institute. In this time period the author held three lectures at an Austrian University and was testing a simpler system of the kind which will be presented here. The main difference was that the mails were not pooled between mailboxes, so the author's model was trained only on his own mails. Indeed a few mails by students were misclassified. By checking each mail at least once, the ham and spam error rates of the old system could be estimated. This also makes mailbox #8 the one which has been cleaned most thoroughly. The ham error rate of 3.46% (FP=0.2%) clearly needs to be improved, but the spam error rate of 0.47% (FN=0.45%) is already quite good. Pooling the training data from multiple mailboxes tends to improve both ham and spam error rate. Detailed results on this observation will be published elsewhere.

Table 2: Mailboxes used for our evaluation. Number of hams (#H), spams (#S), and the dates of the received mails are shown, separately for ham and spam mails. Note that the date headers can be faked and are not completely reliable, especially for spam mails.

mbox	#H	#S	received during	
#1	15248	3319	12/88-07/04	01/97-07/04
#2	8982	10605	01/02-09/04	02/02-09/04
#3	3608	568	09/97-06/04	06/04-06/04
#4	2167	1123	04/96-06/04	06/04-06/04
#5	1589	3083	07/02-07/04	02/03-07/04
#6	7539	1838	09/99-07/04	05/04-07/04
#7	3278	3229	06/01-06/04	06/04-06/04
#8	1387	22795	10/04-02/05	10/04-02/05
sa-easy	3900	n/a	07/02-12/02	n/a
sa-hard	250	n/a	05/02-12/02	n/a
sa	n/a	1897	n/a	06/01-12/02

The last three mailboxes (sa-easy, sa-hard and sa) are publicly available at `http://spamassassin.apache.org/publiccorpus/` and have been added for reference and validation purposes: *sa-easy* is easy_ham plus easy_ham_2, *sa-hard* is hard_ham, and *sa* is spam plus spam_2.

4 Learning Systems

For training all our learning systems, we have randomly drawn (without replacement) a roughly same-sized set of spam and ham mails from each mailbox. The size was chosen so that in the smaller set (either ham or spam, depending on mailbox), about 50% of the data remained for testing. The total training size was thus always less than 50% (25% on average). Compared to a tenfold cross-validation where the training size is 90%, this is a far more realistic challenge. Training sets from all mailboxes were combined into a single training set of SH_{ratio} near 1, and all the remaining ham and spam mails were used for testing.

Training and testing was repeated ten times with different random orderings of the mailboxes' mails. Average and standard deviations of ham and spam error rates per mailbox are reported later. #8 became available too late to be incorporated into the evaluation, but has been included in the system-wide model which has recently been deployed at our institute.

[6]Error rates for manual verification of 0.45% (true hams), and 0.64% (true spams) were reported by [3] for a SH_{ratio} of 4 and a corpus of 50,000 mails.

[7]Default settings without bayesian model. SA has been installed at our institute since 2002, has a small FP rate but an FN rate of around 50%, so it is an obvious choice for bootstrapping spam collections.

4.1 SA-Train

SpamAssassin is an open-source hybrid spam mail filter incorporating a state-of-the-art bayesian learner as well as a set of 900+ human-created heuristic rules for spam recognition. SpamAssassin thus incorporates background knowledge on spam in the form of heuristic rules as well as a bayesian classification system. Contrary to a pure bayesian approach, this makes adapting the system harder, because it not clear when to adapt the scores, train the bayesian filter, or both. The bayesian filter is initially disabled and activated only when 200 mails have been manually trained.

Our approach to training SA, SA-Train, can either be viewed as an application of machine learning techniques to the problem of optimal score assignment and bayesian learning within SpamAssassin from a mainly empirical viewpoint; or as an investigation into approaches to multi-view learning (one view is the bayesian learner; another is the score assignment) within SpamAssassin.

We used version 2.63 of SpamAssassin[8] . SA is a hybrid classifier with a set of 900+ heuristic rules, and a bayesian learner. Each heuristic rule has a weight (score) attached. Rule matching is binary and based on perl regular expression matching. The sum over all matching rules is the full score for the mail. A user-definable threshold is used to determine if a mail is to be classified as spam or ham. The bayesian learner is integrated into the ruleset as a small set of pseudo-rules (e.g. BAYES_00 matches when bayes spam probability is between 0% and 5%, BAYES_05 when the probability is between 5% and 10% etc.), also with an attached user-definable score. A genetic algorithm has been used to optimize the scores for all the rules and the bayesian pseudo-rules on a large corpus of spam and ham mails.

The initial bayesian model for SpamAssassin was taken from a model which has been sporadically trained by the author prior to June 2004 on his own spam and ham mails. An initialization was necessary since the bayesian model is not activated unless it contains at least 200 mails. Autolearn, which automatically trains some spam and ham mails, has been switched off.

Since initially no training set errors are known from the previous step, we start with score learning. The metadata consists of the known mail classification (spam or ham, from training set) and the set of SA rules (including bayes pseudo-rules) which match the corresponding mail. Each rule is represented by a binary attribute. For such a linear binary classification task, a linear Support Vector Machine is the most appropriate system. We used an open-source implementation of SVMs based on the SMO algorithm.[9] After training, the training set errors are counted. A training set error of zero (i.e. a perfect model) yields an early exit. A nondecreasing training set error after the first three cycles also yields an early exit.

The training set errors from the SVM model are afterwards trained via the bayesian model (i.e. via sa-learn), and the process is repeated beginning with score learning. So we repeat until no training set error are found (which happens in 72.8% or runs), or until the errors do not decrease in one step. This is similar to Train-Until-No-Errors (TUNE, [11]). At the end, a user_prefs setting file and a bayesian model for SA is available that can replace the default settings.

Earlier experiments with this kind of system are reported in [8]. Multiple runs of V6 are most similar to our system, but we have found that spam collections by a single user are not sufficient.[10]

4.2 CRM114

For CRM114, we again used Train-Until-No-Errors (TUNE, [11]) modified with an early exit on nondecreasing errors or achievement of a perfect model (i.e. no training set errors). TUNE repeats Train-On-Error (see below) until a perfect model has been learned. This does not always happen, so the early exit condition is essential to prevent endless looping. The exit condition is not checked in the first three cycles to allow initial progress, similar to simulated annealing. The mails were trained in arbitrary sequence given by the initial randomization, and only those mails not already classified correctly by the system were trained (Train-On-Error (TOE), [11]). This step was repeated on the same training set until the exit condition (nondecreasing error) was satisfied, or a perfect model was achieved. The mails were cut off at 1MB size (i.e. only the first MB from each mail was used for training) because the system exhibited very erratic runtime behaviour for overly large mails. A cutoff at 10k as proposed by the author of CRM114 was tested and found to reduce performance.

4.3 SpamBayes

For SpamBayes[11], we trained on the training set once. The default thresholds of 0.9 for spam mails and 0.2

[8]At that time, version 3.0.2 was not yet available. Of course the same training procedure would also be applicable to the new version.

[9]weka.classifiers.functions.SMO, from the WEKA data mining suite, `www.cs.waikato.ac.nz/~ml/weka`.

[10]One user reported a FN of 50% – thus half of this user's spam was unknown to my trained system.

[11]`spambayes.org`

for ham mails classified a large majority of mails as unclear. A threshold of 0.5 was the sensible choice a priori for training sets with equal number of ham and spam mails (SH_{ratio}=1), and proved to reduce spam error rate significantly at little or no deterioration of ham error rate. SpamBayes is based on ideas by [4], and is mainly a bayesian learner. It is arguably the simplest system presented here.

Basically, a bayesian learner splits each mail into words via tokenizer (for mail headers and body separately), and counts how often each word appears in ham mails resp. spam mails. These counts are then used to estimate $P(w_x|ham)$ and $P(w_x|spam)$ for all words w_x, using e.g. LaPlace correction to prevent the occurrence of zero probabilites. By applying Bayes' Rule, it is possible to estimate the probability of a new mail being ham, i.e. by computing the product of all $P(ham|w_x)$ and the prior probability $P(ham)$, and also the product of all $P(spam|w_x)$ and the prior probability $P(spam)$, followed by renormalization of these two probabilities (i.e. ensuring that they sum to 1).

5 Ready-to-use systems

Among the ready-to-use systems, we have chosen Symantec BrightMail 6.0.1[12], Google Mail[13] and SpamAssassin 3.0.2[14] (SA-Default). These were run on all mailboxes, including #8 and the sa mailboxes. For each system, we developed an interface to the respective filtering system, and simply pushed the mails including all headers and attachments from all mailboxes through it, noting whether they were classified as spam or not.

During the evaluation of the ready-to-use systems, we checked each ham error for correctness. We found 0.22% spams in the ham mailbox #1, 0.06% spams in the ham mailbox #2 and 0.03% spams in the ham mailbox #7 and have not counted these as errors for the ready-to-use systems SA-Default and BrightMail.[15] All other mailboxes were clean. Since our experiments with the learning systems had already been finished the error estimates may be slightly pessimistic, especially w.r.t mailbox #1.

5.1 SA-Default

For SpamAssassin 3.0.2. (SA-Default), we used Debian 2.4.18 with a backport of SA 3.0.2. Leaving everything at its default setting, we simply executed spamassassin on each mail separately with the -e parameter which returns the classification of the mail as exit code. The bayesian model was not initialized, auto_learn was turned off (default). The auto-whitelist was also switched off.

5.2 BrightMail

We installed a one-month test version of Symantec BrightMail 6.0.1 to a specially installed SuSE Linux 9.1 system. We pushed all mails through the MILTER interface of bm_filter via sendmail, but prevented further delivery by closing port 25 (SMTP) and deleting the queue regularily. The MAIL field was reconstructed by analyzing the 'From:' and 'From ' headers. For 99.95% of mails the fields could be reconstructed; the others were set to an arbitrary default value. Some of the reconstructed MAIL fields contained illegal domains, so the corresponding filtering was taken out of sendmail.cf rather than using the default value, to give BrightMail the maximum amount of information for filtering. The RCPT field was given by the mailbox and corresponded to the original user's address. BrightMail classified mails as virus-infected (1.2%), spams (23.4%) and hams (75.3%). The remaining 0.1% of mails were tagged with *WARNING - NOT VIRUS SCANNED* and treated as ham. These classifications are mutually exclusive, so BrightMail cannot conceive of a spammy *and* virus-infected mail. Since 99.0% of virus-infected mails appear in spam mailboxes, we have treated them as spam.

5.3 Google Mail

For Google Mail, we similarily used sendmail to send all mails to the GMail server via SMTP. The MAIL field was set to the author's email address, and the RCPT field to his Google Mail address. Thus, the original MAIL and RCPT data was lost. However, we still sent the full mail including headers and attachments. After each mailbox (ham and spam separately), we noted the number of entries in inbox and spam on GMail. The number of entries is not the number of separate mails, but threads of mails, and it was not possible to switch off this behaviour. Therefore, we have estimated ham and spam error by the proportion of these numbers which relies on the assumption that the average length of a thread is the same for mails classified as ham and mails classified as spam by GMail.

GMail is still at beta stage, so it was not surprising that 3.1% of the mails were rejected by the server. 0.13% of ham mails and 2.45% of spam mails were rejected with *5.7.0 Illegal Attachment* error. 0.5% of

[12]Recently made available as a one-month test version.

[13]Thanks to the Google Team for sending an invite.

[14]Winner of the Datamation Product of the Year Award 2005 in category Anti-Spam.

[15]For GMail, lack of the specific set of mails classified as spam / ham prevented us from correcting the error rates.

ham mails were *relayed to non-DSN-aware mailer* – the mailer was probably confused by conflicting data in headers and MAIL. 0.002% (one) ham mails was rejected with *5.6.0 Headers too large (32768 max)*. These errors should not have a significant influence on the system's evaluation.

6 Results

Table 3 shows the results; Figure 1 shows H_{err} and S_{err} just for the learning systems. A SH_{ratio} of 17.5 was chosen as average between 15 and 20, which are the ranges reported by our eight users. Spam and ham error of SA-Train cnad CRM114 are comparable to human error (0.45% ham error, 0.65% spam error, [3]). We are however nowhere near 99.9% accuracy - 99.35% is the best we can do for now. By the single measure *Err* all of the ready-to-use systems perform worse than all learning systems. However, *Err/Acc* assume uniform cost for **FP** and **FN** errors, which is unrealistic. By cost-sensitive error Err_{1000}, the ready-to-use and the learning systems perform comparably. The best system with a big lead (6.010 vs. 22.650) is a relatively simple one: SpamBayes. It also has the third-smallest *FP* (1:25,000) and the third-smallest *FN* (1.847%), and no system manages to perform better than it on both *FP* and *FN*. GMail is very good at recognizing spam mails – arguably the best among the ready-to-use systems – but its high ham error rate is rather disappointing.

We note that BrightMail fails to achieve the marketed performance of an FP of 0.0001% (1:1,000,000) and FN of 5% by a great margin: even at an SH_{ratio} of 17.5 to 1, FP is already 0.003% (1:31,356) and FN is 38.3% (averaged over all mailboxes). This naturally lead to the question whether the ready-to-use systems are better at classifying more recent spam mails. Mailbox #8 contains all mails received between Oct. 4th 2004 and Feb 13th 2005 by a single user, and is suitable for answering this question. See Table 4 and Figure 2 for the results. These show clearly that BrightMail significantly improves over time. However, even when we assume that 5.04% is achievable in practice – which is by no means sure, since the processed spam mails were at that time already more than a week old, and there may be a time lag in updating BrightMail – Err_{1000} would be re-estimated at 8.66% and thus still almost 50% higher than the corresponding value 6.01% for the pure bayesian learner SpamBayes. Still, this indicates that BrightMail may focus on correctly classifying the most recent spam at the cost of misclassifying older spam, which should be kept in mind when evaluating it and other ready-to-use system with similar characteristics.

Table 4: This gives results of the ready-to-use systems on mailbox #8. To preserve space, all numbers are in percent (i.e. have been multiplied by 100). E.g. 0.361 stands for 0.00361, or around 1:277.

mbox	SA-Default		BrightMail		GoogleMail	
no.	H_{err}	S_{err}	H_{err}	S_{err}	H_{err}	S_{err}
#8	0.361	48.16	0.072	16.06	3.187	2.173
Err	45.572		15.192		2.228	
Err_{1000}	65.066		19.085		174.326	
sa-easy	0.000	n/a	0.154	n/a	5.090	n/a
sa-hard	0.400	n/a	0.400	n/a	42.45	n/a
sa	n/a	38.06	n/a	54.14	n/a	23.24

We think underreporting of FP errors may be a suitable explanation. For example earlier error estimates of around 0.2% lead us to expect around 350 FPs in the nine months since the spam filters were installed, distributed randomly over all seven mailboxes. In fact, one user reported 6 FPs and the others reported none at all. This can have a large influence on the estimation of performance of a running system, especially at a high SH_{ratio} and with a large number of users. BrightMail claims 100,000,000 users, so if each user gets ten mails per day and overall 1000 FPs are reported daily, this would yield an overly optimistic estimate of FP as 1:1,000,000.

Let us return to Figure 2. As can be expected from a static system, SA-Default's spam error rate deteriorates significantly over time. SpamBayes's model was also static over the testing period, and was trained on mailboxes #1-#7 except the #1 ham mailbox[16], but deteriorates to a far smaller degree. This is even more surprising since SpamBayes (5.48%) has a similar error rate to BrightMail (under above assumption: 5.04%) although the latter is updated regularily (around 700MB of updates were received weekly). SpamBayes therefore seems to be an interesting venue for further research as it generates a model which is stable over relatively long periods of time before necessitating a retraining or an update.

7 Related Research

[3] presents a comprehensive study on eight months of personal mail. Their evaluation is concerned with sequential training efforts where we are interested in determining performance on larger corpora. Using ROC curves to compare spam filtering systems is something we will look into. Concerning evaluation measures, their ham misclassification fraction *hm* is equivalent to our H_{err}, and their spam misclassification fraction

[16]The largest ham mailbox was removed to return the SH_{ratio} to near one as in training.

Table 3: This gives the full results of the ready-to-use systems (on the left) and the learning systems (on the right). To emphasize differences all numbers are in percent (i.e. have been multiplied by 100). E.g. 0.178 stands for 0.00178, or around 1:561. *FP*, *FN*, *Err* and Err_{1000} are computed with $SH_{ratio} = 17.5$.

mbox	SA-Default		BrightMail		Google Mail		SA-Train		CRM114		SpamBayes	
no.	H_{err}	S_{err}	H_{err}	S_{err}	H_{err}	S_{err}	H_{err}	S_{err}	H_{err}	S_{err}	H_{err}	S_{err}
#1	0.178	28.924	0.079	32.781	2.221	3.246	0.701	0.627	0.507	0.663	0.343	2.779
#2	0.045	9.298	0.000	21.480	0.647	3.697	0.154	0.275	0.198	0.602	0.029	1.511
#3	0.000	26.232	0.000	52.465	0.364	2.305	0.278	1.023	0.185	0.282	0.000	1.165
#4	0.000	21.995	0.000	51.826	0.277	1.904	0.308	0.892	0.129	0.393	0.000	1.302
#5	0.000	15.926	0.252	38.209	8.042	2.398	1.374	1.116	1.827	0.987	0.088	2.380
#6	0.199	12.622	0.080	47.769	1.168	1.376	0.964	1.002	0.648	0.566	0.077	1.938
#7	0.092	23.413	0.000	38.712	0.569	3.697	0.269	0.731	0.378	1.122	0.000	2.597
Avg.	0.073	19.773	0.059	40.463	1.898	2.660	0.578	0.810	0.553	0.659	0.077	1.953
±StD	0.086	7.295	0.093	11.210	2.790	0.904	0.454	0.291	0.593	0.302	0.123	0.648
FP/*FN*	0.004	18.704	0.003	38.276	0.103	2.516	0.031	0.766	0.030	0.623	0.004	1.847
Err	18.708		38.279		2.619		0.797		0.653		1.851	
Err_{1000}	22.650		41.465		105.111		32.009		30.52		6.010	

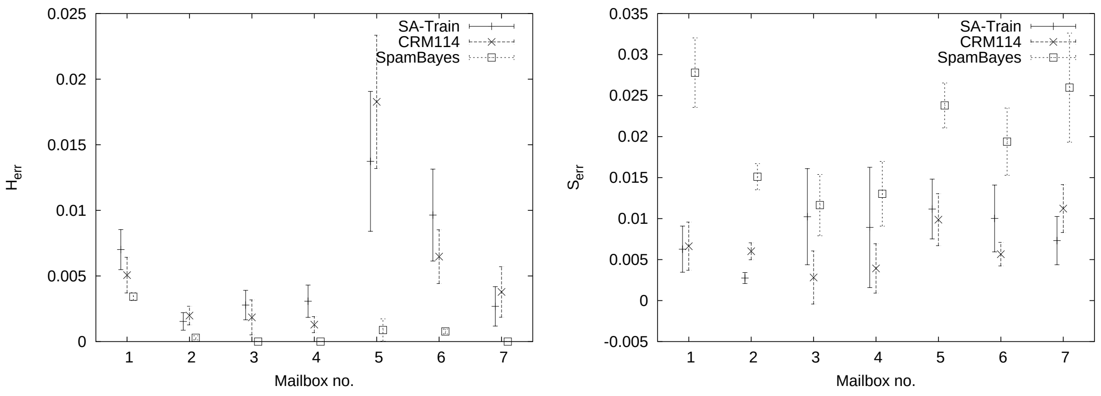

Figure 1: This figure show H_{Err} (on the left) and S_{Err} (on the right) for mailboxes #1-#7, and the learning systems. Error bars are the standard deviation of test set error over ten training runs. 0.025 = 2.5%

sm is equivalent to our S_{err}. He reports a final human error rate for ham error of 0.45% and for spam error of 0.64% which may be more realistic than the one quoted by [11] since it is based on a larger sample of around 50,000 mails.

Our results disagree with [1] who found that current spam filters are not suitable for deleting messaged classified as spam. Assuming a human error rate of 0.45% ham error and 0.64% ([3]), two of the six systems tested perform comparably, and three others have a ham error rate which is an order of magnitude less. Since minimizing ham errors (**FPs**) is more important to most users, clearly almost all of the tested systems are acceptable. They used Total Cost Ratio with a simple cost of 1000 for all false negative errors, similar to our Err_{1000}.

[10] has described an approach to use Genetic Algorithm techniques to optimize rule scores within SpamAssassin. Their approach differs from ours in that they ignore the bayesian model. For SpamAssassin 3.0.2., the developers have switched to a perceptron for determining useful default settings for rule scores, so they are only a small step away from the linear support vector machine we use – arguably the best algorithm for learning a linear discriminant model in a classification setting.

8 Conclusion

We evaluated our own learning spamfilter along with five other state-of-the-art filtering systems on our mail collections of around 100,000 mails. A relatively simple bayesian filter, SpamBayes, was found to outperform the other filtering systems by cost-sensitive error, and was competitive concerning *FP* and *FN* rate. Our systems SA-Train and CRM114 both performed comparably to human error, but misclassified more true ham mails than SpamBayes, BrightMail or SA-Default.

When using spam mails to evaluate ready-to-use spam

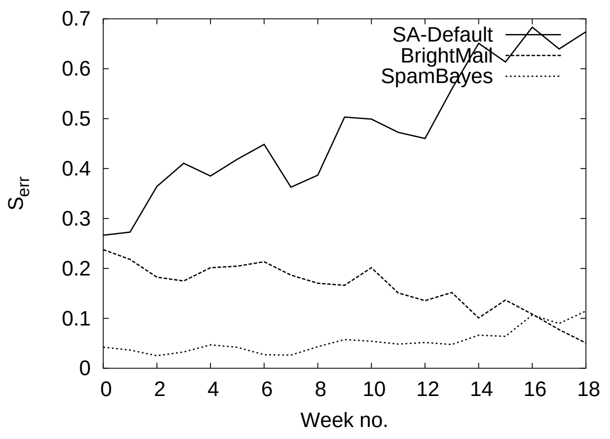

Figure 2: This figure shows S_{err} on mailbox #8 for the ready-to-use SA-Default and BrightMail, and for SpamBayes. SpamBayes has been trained on mailboxes #1-#7 except ham mails of #1. Each S_{err} value is averaged over a full week. 0.7 = 70%. Absolute H_{err} (**c**) is 5 for SA-Default, 1 for BrightMail and 2 for SpamBayes, out of 1387 ham mails.

filtering systems, the age of the spam mail becomes significant: We have noted up to tenfold decreases or increases in spam error rate for recent spam mails. This indicates that ready-to-user spam filtering systems either focus on classifying old spam correctly, but fail to generalize to new spam (e.g. SpamAssassin 3.0.2 default settings); or focus on classifying new spam correctly at the cost of misclassifying old spam (e.g. Symantec BrightMail).

The first result is expected since no static system can expect to keep up with spam forever. Spammers are always improving their systems to keep up with static models. The second result is unexpected, and immediately suggests a complementary vulnerability of the system: by reusing old spam templates and systems, spammers could easily attack the system.

We have shown that combining training mails from multiple mailboxes (#1-#7) and removing mails to ensure a SH_{ratio} near 1 – at least for SpamBayes – reduces this time-dependency significantly, reduces the amount of retraining needed, and makes the filtering process less vulnerable to both kinds of attack over an extended period of time.

9 Acknowledgements

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture and by the Austrian Federal Ministry for Transport, Innovation and Technology. We want to thank our anonymous colleagues for contributing the mail collections.

References

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. and Spyropoulos, C.D. An Evaluation of Naive Bayesian Anti-Spam Filtering. In Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, p. 9–17. Potamias, G., Moustakis, V., van Someren, M.n (eds.), Barcelona, Spain, 2000.

[2] le Cessie, S. and van Houwelingen, J.C. Ridge Estimators in Logistic Regression. Applied Statistics, Vol. 41, No. 1, pp. 191-201, 1997.

[3] Cormack, G., and Lynam, T. A Study of Supervised Spam Detection Applied to Eight Months of Personal Email. `http://plg.uwaterloo.ca/~gvcormac/spamcormack.html`, July 2004.

[4] Graham, Paul. A Plan For Spam. `www.paulgraham.com/spam.html`, 2003.

[5] Hoffman, Paul and Crocker, Dave. Unsolicited bulk email: Mechanisms for control. Technical Report UBE-SOL, IMCR-008, Internet Mail Consortium, 1998.

[6] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf, C. Burges, and A. Smola, eds., MIT Press, 1998.

[7] Quinlan, R. C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[8] Seewald, A.K. Combining Bayesian and Rule Score Learning: Automated Tuning for SpamAssassin. Technical Report, Austrian Research Institut for Artificial Intelligence, Vienna, TR-2004-11, 2004.

[9] Serge Gauthronet and Etienne Drouard. Unsolicited commercial communications and data protection. Technical report, Commission of the European Communities, 2001.

[10] Sergeant, Matt. Internet Level Spam Detection and SpamAssassin 2.50. 2003 MIT Spam Conference, Cambridge, Massachusetts, U.S.A, 2003. `spamconference.org/proceedings2003.html`

[11] Yerazunis, William S. The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It. 2004 MIT Spam Conference, MIT, Cambridge, Massachusetts.