An Evaluation of Naive Bayes Variants in Content-Based Learning for Spam Filtering

Alexander K. Seewald <alex@seewald.at>
Tel. +43(664) 110 68 86, Fax. +43(1) 2533033-2764
Seewald Solutions, A-1180 Vienna, Austria

Abstract

We describe an in-depth analysis of spam-filtering performance of a simple Naive Bayes learner and two extended variants. A set of seven mailboxes comprising about 65,000 mails from seven different users, as well as a representative snapshot of 25,000 mails which were received over 18 weeks by a single user, were used for evaluation. Our main motivation was to test whether two extended variants of Naive Bayes learning, *SA-Train* and *CRM114*, were superior to simple Naive Bayes learning, represented by *SpamBayes*. Surprisingly, we found that the performance of these systems was remarkably similar and that the extended systems have significant weaknesses which are not apparent for the simpler Naive Bayes learner. The simpler Naive Bayes learner, *SpamBayes*, also offers the most stable performance in that it deteriorates least over time. Overall, *SpamBayes* should be preferred over the more complex variants.

Keywords: Empirical study, Spam filtering, Machine Learning, Naive Bayes

1 Introduction

Spam has become a problem of global impact. According to a study undertaken for the European Commission, Internet subscribers worldwide wasted an estimated 10 billion Euro per year just in connection costs due to Spam already in 2001 [15]. 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 thus receive around 15-20 spam mails – more than 1,000 spams per day per user in the worst case.

Several approaches exist to deal with spam [6]. Filtering approaches based on simple message features such as the occurrence of certain words (e.g. Viagra) are widely used. In fact, most email clients already allow their users to manually build such 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. Such attacks usually work well. They are aided by the human ability to recognize ambiguous words. For example, there are 600,426,974,379,824,381,952 ways to spell viagra comprehensibly, see `http://cockeyed.com/lessons/viagra/viagra.html`.

Another option for filtering is to collect samples of spam and ham (i.e. non-spam) and 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, which operate on large word occurrence vectors and utilize Bayes' Rule. Most state-of-the-art spam filters now include learning systems, e.g. *SpamBayes* (`spambayes.org`), *CRM114* (`crm114.sourceforge.net`) and SpamAssassin (`spamassassin.org`). We will investigate all three of these approaches, the last one augmented by our own training procedure, *SA-Train*. The obvious goal would be to train a model that is harder to attack, i.e. one that deteriorates less over time than the manual filtering approaches.

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 rule creation, adaption or refinement; Naive Bayes learning, or non-disclosed techniques. In [13], we have compared the performance of various learning and ready-to-use systems, and found that the best ready-to-use system (Symantec BrightMail 6) already performs competitively to the best learning system (*SpamBayes*), which might seem to make training spamfilters seem too much effort. However, the effort in updating and running these systems is high and even only receiving updates may be prohibitive for small research institutes or small companies. For example, during the evaluation of the test version of BrightMail, around 700 megabytes of updates were received weekly. Around 7 megabytes of ham and spam email are received at our institute per user and week, so the break-even point – where the bandwidth for BrightMail equals the email bandwidth – would be achieved at around 100 users. Below 100 users a locally trained filter may be preferrable for efficiency and cost reasons.

Apart from these content-based systems there are also behaviour-based systems. These systems decide if a given mails is spam based not on its content, but on the behaviour of the sending mail server. Sending the same mail to all accounts of a mail domain within a few milliseconds, or starting the SMTP session with HELO followed by an IP address instead of a fully qualified domain, as well as more elaborate techniques (e.g. greylisting - `http://projects.puremagic.com/greylisting/whitepaper.html`) have been proposed. The approach looks promising and would probably benefit from combination with content-based approaches. Behaviour-based systems will not be discussed here as completely different data would have to be collected.

Our main motivation was to evaluate our own novel learning approach to train SpamAssassin (*SA-Train*) which has been developed and refined over a year to two other state-of-the-art learning systems, *CRM114* and *SpamBayes*. Of these systems, both *CRM114* and *SA-Train* can be seen as extensions of Naive Bayes learning[1] (represented by *SpamBayes*) in two different directions: towards more complex concept descriptions (*CRM114*: phrases with wildcards, which also necessitates a more complex way to estimate phrase probabilities via Markov Models), and by extending the NB learner with human background knowledge (SpamAssassin: 900+ handwritten rules to recognize spam). In the light of these conceptual similarities, we can rephrase our main motivation as to determine whether these two ways to improve on Naive Bayes learning have been successful. It will turn out that they have not been successful.

The choice to use existing state-of-the-art spam filtering systems rather than a set of current machine learning algorithms on reasonable feature-based representations was informed by earlier unpublished experiments where we found out that it is indeed very hard to achieve the performance of refined spam filtering systems from scratch. These systems – both learning algorithms and feature sets – have all been fine-tuned over years on large, mostly private, mail collections and we cannot hope to easily replicate their work. On the upside, an advantage of our choice is that it effectively prevents implementation bias.

Previous studies, e.g. [2], have focussed on representative mailbox collections from a single user, collected online during spamfilter training. The focus on a single user is understandable, since collecting and cleaning mails is a major

[1] with usual setting for text mining: splitting each document (mail) into words, and using each unique word as a feature in a binary word occurrence vector. The probabilities are estimated in the usual way, see Section 4.3

effort, and long periods of time are needed to obtain sufficiently large mailboxes for extensive experiments. However, this limits the generality of the results, as mailboxes are usually quite variable, and conclusions do not always agree.

Our approach compares results on seven different mailboxes, therefore the results are likely to hold more generally. However, our mailboxes are not truly representative since training a spam filter online (which would automatically collect a representative set of ham and spam mails) is very costly: several months are needed to achieve acceptable performance. During this period, all received mails – ham and spam mails both – need to be thoroughly checked. None of my colleagues was willing to spend so much time to obtain a good spamfilter. To speed this process up, we have combined a set of recent spam mails with historical ham mails from each user's mailbox, and used this mailbox for training. This set is no longer representative, as the time frame of ham and spam mails is different. Additionally, the ratio of spam to ham mails is also different and somewhat arbitrary. Even though these mailboxes are no longer truly representative, the resulting *SA-Train* spam filters show exceptionally good performance which remains stable for several months. This may be because concept drift is much smaller for ham than for spam mails, and therefore the mixing of recent spam mails with older ham mails does not affect generalization performance adversely. In light of these positive results, we have chosen to use non-representative mailboxes for the evaluation. However, the non-matching timeframes of ham and spam mails prevents using these mailboxes to determine the deterioration of performance over time due to concept drift. For this, we had to use one representative mailbox by another user, #8.

Overall, the contributions of this paper can be summarized as an in-depth analysis of three state-of-the-art content-based learning spam-filters – the learning methodology of one of them, *SA-Train*, due to our research – which incorporates the following aspects:

- Performance (FP and FN rate) with default thresholds
- Threshold-independent performance (i.e. FP/FN trade-off curves: plotting FP vs. FN rate for a large set of thresholds at double-logarithmic scales)
- (Class-)Noise-level susceptibility (default threshold only)
- Performance over time (default threshold and threshold-independent)
- Performance with simplified training procedures (default threshold and threshold-independent)
- Relation between time-independent and time-dependent estimation of error w.r.t performance and shape of FP/FN trade-off curves.
- Effect of Concept Drift on performance w.r.t. different models and training methods

The primay focus was to determine whether the extended Naive Bayes learning systems (*SA-Train* and *CRM114*) outperform their ancestor, represented by *SpamBayes*. The rest of the paper is structured as follows: Section 2 describes related research. In Section 3, we will explain the evaluation measures we used. In Section 4, we will describe the learning systems and how we trained each one of them. In Section 5, we will describe our mailbox collection and how we collected and cleaned it. In Section 6, we will shortly describe the remainder of experimental setup. Section 7 is concerned with our primary experiments and results. Section 8 is concerned with our experiments and results with respect to concept drift. Section 9 summarizes the overall conclusions.

2 Related Research

[2] presents a comprehensive study on eight months of personal mail, using eleven variants of six open-source filters. Their evaluation is concerned with sequential training efforts where we are interested in determining performance on larger corpora. Concerning evaluation measures, their ham misclassification fraction hm is equivalent to our FP_{rate}, and their spam misclassification fraction sm is equivalent to our FN_{rate}. They report a final human error rate for FPs of 0.45% and for FNs of 0.64% which is based on a sample of around 50,000 mails.

[12] propose an approximation for the string kernel, making it more feasible to run on large problems. As a sample task, they applied the modified kernel to a spam filtering task: recognizing spam and ham mails solely from the sender email address. The quoted accuracy of around 84% (based on two-fold cross-validation on 3,902 samples from our corpus) is of course much worse than for state-of-the-art spam filtering systems, but string kernels could in the medium to long term become a feasible way to build better systems.

[7] present the *SpamBayes* system. They note that a SH_{ratio} of 1 works best for training, which is what we found as well, and that incremental training takes much longer to converge than batch-style training. They also note that ham and spam collections should overlap in time as the system would otherwise be inclined to distinguish them by the year token in the Date: header which would yield very bad results on unseen data.

[17] proposes TUNE and TOE-style learning approaches, and reports results for several types of learning spamfilters (including *CRM114*) on publicly available mail collections. His results are hampered by the non-representative nature of the public collections, which may explain that we cannot replicate his finding on the superiority of TUNE plus *CRM114* here. However, he also notes that training spam filters for several users via pooled mailboxes may work quite

well, which we also found. He also proposes some behaviour-based features for filtering spam.

[10] presents an extensive empirical evaluation of a memory-based spam-filtering system. Their focus is spam filtering for mailing lists, and is thus different from our focus on filtering personal mail, but allows them to use available public corpora for evaluation. They evaluate memory-based systems differing in neighborhood size, feature set size and the size of the training corpus while we are focussing on different training methodologies, modes of performance estimation, and noise-level susceptibility. Their study is mainly empirical, but quite comprehensive, and demonstrates that memory-based learning has its niche in this research field.

[9] reports that a scheme for combining classifiers known as stacked generalization improves the performance of spam categorizers. While the use of ensembles is similar in spirit to SpamAssassin, their ensemble is a set of classifiers trained on bag-of-word representations of mails while *SA-Train*'s ensemble (human-created rules and a Naive Bayes model) is expected to be more diverse than their ensemble which differs only in the choice of learning algorithms at the base level. Diversity is one of the key elements for successful ensemble learning. However, the *SA-Train* ensemble is not able to improve on a simple Naive Bayes learner in our setting which is somewhat similar to stacked generalization.

Our results disagree with [1] who found that current spam filters are not suitable for deleting messages classified as spam. Assuming a human error rate of 0.45% FP rate and 0.64% FN rate [2], at least one of the three systems tested here performs comparably. Since minimizing FPs is far more important to most users, clearly almost all of the tested systems are acceptable with properly set treshold. They used Total Cost Ratio with a simple cost of 1000 for all false negative errors while we reported FP and FN rate separately, and also showed the trade-off curve between FP and FN rate in a different form.

[16] 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 Naive Bayes 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 a linear support vector machine – arguably the best algorithm for learning a linear discriminant model in a classification setting, although it still needs some adaptation to generate useful probability estimates.

[5] reports a comparative evaluation of several machine learning algorithms on the text of messages and also a set of 9 heuristics. The reported improvement due to the use of heuristics is modest. Our approach *SA-Train* uses 900+ heuristics from SpamAssassin but also fails to improve on a simpler Naive Bayes learning system.

[3] compare boosted decision trees, SVM, Ripper and Rocchio with several feature set variants on a small corpus of 3000 ham and spam mails. They conclude that SVM (with binary bag-of-word feature) and boosted decision trees (with term-frequency weighted bag-of-word features) are the two best candidates. They also find that using no stop-word list is preferrable. Their work is similar to ours in that they also consider corpora from multiple users, and in that they consider several learning systems in an empirical study. However, their corpus is much smaller that ours, which limits the generality of their results. It would be interesting to repeat their experiments on our corpus in the future, but we would be bound to expect that the performance falls short of the state-of-the-art systems we have tested here.

[11] is one of the first papers dealing with bayesian approaches to spam filtering. They are arguing for the importance of probabilistic classification, as well

as the use of domain knowledge, to create state-of-the-art spam filters. Domain knowledge in tuning features and learning algorithms for spam filtering systems continues to be a significant factor even today. Their work is similar to ours in that they try to evaluate their proposed system in a realistic setting, albeit on a much smaller corpus. Many of their proposed domain-specific features have in the meantime found their way into other systems such as SpamAssassin.

3 Evaluation measures

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

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

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

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

$$FP_{rate} = \frac{c}{\#Hams} \qquad \text{FP rate, Ham Recall} \qquad (5)$$

$$FN_{rate} = \frac{b}{\#Spams} \qquad \text{FN rate, Spam Recall} \qquad (6)$$

$$Err = \frac{b + c}{\#N} =$$

$$= FN_{rate}\frac{1}{1 + \frac{1}{SH_{ratio}}} + FP_{rate}\frac{1}{1 + SH_{ratio}} \qquad \text{Overall error rate} \qquad (7)$$

$$Acc = \frac{a + d}{\#N} = 1 - Err \qquad \text{Overall accuracy} \qquad (8)$$

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 (+). 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 **P**ositives) and **d** (**T**rue **N**egatives) are the number of spam resp. ham mails which are correctly predicted by the system. **c** (**F**alse

Positives) 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. Since our collected mailboxes (except for #8) are not representative concerning the ratio of spam to ham mails (SH_{ratio}), we chose two measures which do not depend on this property: FP_{rate} and FN_{rate}. FP_{rate} can be interpreted as $P(\text{misclassified}|\text{true ham})$, i.e. the estimated probability that a true ham mail is misclassified as spam. FN_{rate} can be similarily interpreted as $P(\text{misclassified}|\text{true spam})$, 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 and follows the standard nomenclature in the field of spam filtering research. We note the equivalent terms of ham recall for FP_{rate}, and spam recall for FN_{rate}, which follow the usual definition of recall from Information Retrieval literature.

We always report separately FP_{rate} and FN_{rate} for each system and mailbox – if a single error or accuracy value is needed for comparison to other studies, it can be computed from these values by formulas (7) or (8) given the additional value of SH_{ratio}.

4 Learning Systems

Here we will describe the three learning systems we investigated, along with their settings and training methods. All of these systems base their classification decision on mail content including headers. The first two systems, *SA-Train* and *CRM114* are trained in a non-default way. For these, only the errors of the current model are trained (Train-On-Error, TOE), beginning with a reasonable default model. Furthermore, this training step is repeated until convergence (Train-Until-No-Errors, TUNE). We were motivated to use this approach because of excellent results in training our initial filters [14] and since it has been reported as working very well for *CRM114* by its author in [17], on a set of publicly available mailbox collections. We have however found some weaknesses related to this training approach, and therefore chose to investigate simpler training approaches as well: The last system, *SpamBayes*, is trained in the usual way – by training all mails from the training set exactly once in a batch-style setting.

Both *SA-Train* and *CRM114* can be viewed as improvements of a simple Naive Bayes learner such as *SpamBayes*. *SA-Train* adds background knowledge in the form of manually created rules; and *CRM114* extends the description language of NaiveBayes from single words to multi-word phrases with additional modifications concerning probability estimates. The main focus of our evaluation is to see whether these extension are successful in improving on *SpamBayes*. Table 2 shows all considered learning systems at one glance.

We chose to use existing state-of-the-art spam filtering systems rather than a set of current machine learning algorithms since the former offer much better performance, having been refined for years on large mail collections. Earlier unpublished experiments confirmed that creating a spam filtering system from scratch is a complex task that should not be underestimated. Finally, using state-of-the-art systems effectively precludes implementation bias.

4.1 SA-Train

SpamAssassin (`www.spamassassin.org`, DataMation 2005 Product of the Year in the category Anti-Spam) is an open-source hybrid spam mail filter incorporating a state-of-the-art Naive Bayes learner (similar to *SpamBayes*) as well as a set of human-created heuristic rules for spam recognition. SpamAssassin thus incorporates background knowledge on spam in the form of heuristic rules as well as a Naive Bayes classification system. Contrary to a pure Naive Bayes (NB) approach, this makes adapting the system harder, because it is not clear when to adapt the scores, train the NB filter, or both. The NB filter is initially disabled and activated only when 200 mails have been manually trained. Anecdotal evidence suggests that adapting the NB filter on its own is not sufficient for reasonable performance - it is also necessary to adapt the scores to prevent misclassification for several types of ham mails.

As the performance of SpamAssassin with default rule scores is poor even for recent spam (see e.g. [13]), we have developed our own approach to training SA, *SA-Train*. This method can either be viewed as an application of machine learning techniques to the problem of optimal score assignment and NB learning within SpamAssassin from an empirical viewpoint; or as a multi-view learning approach (one view is the model of the NB learner; another is the score assignment) within SpamAssassin. An open-source implementation of our training method can be found at `http://alex.seewald.at/spam`.

We used version 2.63 of SpamAssassin. As we mentioned, SA is a hybrid classifier with a set of 900+ heuristic rules, and a NB learner. Each heuristic rule has a weight (score) attached. Rule matching is binary and based on perl regular expression matching. The sum over the scores from 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 NB 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 for each pseudo-rule. A genetic algorithm has been used by the authors of SpamAssassin to optimize the scores for all the rules and the NB pseudo-rules on a large corpus of spam and ham mails, and gives the initial score set, and is available within SpamAssassin.

The initial NB 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 NB model is not activated unless it contains at least 200 mails. Autolearn, which automatically trains some spam and ham mails, has been switched off. The auto whitelist, which averages mail scores over several mails from the same sender, was also switched off.

The training procedure was inspired by Train-Until-No-Errors (TUNE) which was found to work best in [17]. TUNE is essentially a repetition of Train-On-Error (TOE), which in each step trains the training set errors encountered for the previous model, thus converging towards a model with low training set error. Since initially no training set errors are known, we start with score learning. The meta-data 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 with value 0 for nonmatching and 1 for matching rules. For such a linear binary classification task, a linear Support Vector Machine is the most appropriate system. We used an open-source implementation of SVM based on the SMO algorithm (weka.classifiers.functions.SMO, from the WEKA data mining suite, `www.cs.waikato.ac.nz/~ml/weka`. Settings were -E 1 -N 2 -C 1) with a linear kernel (without intercept, i.e. $K(x, y) = <x.y>$), and the complexity parameter lambda was set to the default value of 1. No normalization of the input values was performed, and no parameter optimization was done. The trained system gives both the weights (score) for each rule as well as the threshold, yielding a full classification model. After training, the training set errors are counted. A training set error of zero (i.e. a perfect model) yields an early exit as no further TOE training can be done. A nondecreasing training set error after the first three cycles also yields an early exit. This was added to prevent overfitting in the presence of class noise.

The training set errors from the SVM model are afterwards trained incrementally via the NB learner (i.e. via sa-learn), and the process is repeated from SVM score learning onward. Again, we repeat until no training set errors are found (which happens in 72.8% of the runs), or until the errors do not decrease after the first three cycles. The first three cycles are intended to give the system time for initial progress. During our early experiments, we found that the error sometimes slightly increases in the second cycle, but still falls below the initial value (from the first cycle) on the third or fourth cycle. At the end, a user_prefs setting file and a NB model for SA is available that can replace the default settings and model. Note that the NB model is additive and will include all unique mails that were misclassified in any cycle. Duplicate misclassified mails are learned only once since sa-learn ignores requests to train a previously trained mail again.

Earlier experiments with this kind of system are reported in [14]. Multiple runs of V6 are most similar to our system, but we have found that spam collections by a single user are not sufficient. When evaluating the first trained models on new users (i.e. one whose mails were not part of the training set) FN rates up to 50% were reported – thus half of some user's spam was unknown to the pretrained system. This contradicts previously established notions about high similarity between spams from multiple users within the same organization.

This family of systems has been extensively tested at our institute, and user feedback has been very positive. It should also be noted that the FP rate was always reported to be zero, so the small number of FPs we observed in most of our experiments don't seem to be significant to the user. Training on pooled (*allTrain*, see Section 7.2) mailboxes from multiple users improves on this – one new test user was astonished and told us that the pooled model offered a much lower FN rate than his own locally trained SpamAssassin model.

4.2 CRM114

For *CRM114*, we similarily used Train-Until-No-Errors (TUNE, [17]) modified with an exit on nondecreasing errors after the first ten cycles or achievement of a perfect model (i.e. without training set errors). TUNE repeats Train-On-Error (TOE, see above) until a perfect model has been learned. This does not always happen, so the exit condition is essential to prevent endless looping. The exit condition is not checked in the first ten cycles to allow initial progress, similar to simulated annealing. *CRM114* uses far fewer mails for training in TOE and therefore converges more slowly than *SA-Train*, which motivates the higher number of cycles until the nondecreasing error condition is checked. The training procedure is somewhat similar to *SA-Train* and also follows Yerazunis's recommendations (except for the early exit conditions which were added by us).

CRM114 is based on Sparse Binary Polynomial Hashing with a Bayesian Markov Model. It is a generalization of NB filtering that tries to estimate propabilities for phrases instead of single words, where longer phrases are given more weight. Phrases can include wildcards which are placeholders that can stand for any word. Details can be found on *CRM114*'s homepage, `crm114.sourceforge.net`.

The mails were cut off at 1,000,000 size (i.e. only the first MB from each mail was used for training) because the system exhibited erratic runtime behaviour for larger mails. A cutoff at 10k as proposed by the author of *CRM114* was tested and found to reduce performance. Still, erratic runtime behaviour for the large noiselevel experiment forced us to cut off at 10,000 bytes for the noiselevel and concept drift experiments.

According to experiments which are not shown, this change may modify the FP rate of these experiments by factor 1.56±1.03, and the FN rate by 0.62±0.43 (averaged over mailboxes #1-#7). Note that the FP rate increases by almost the same factor as the FN error is decreased which hints that the differences can be explained as differences in the absolute values of the confidences (equivalent to a different default threshold) rather than differences in the underlying model. We would therefore still expect the FP/FN rate trade-off curves to be similar.

Additionally, three mails had to be manually removed from the training set (only for the time-dependent evaluation of CRM-simple) after training these mails took more than one hour per mail. Normally, training a mail takes a few seconds. This probably makes the tested CRM version unsuitable for batch operation unless appropriately time-limited.

4.3 SpamBayes

For *SpamBayes* (`spambayes.org`, [7]), we trained the full training set once. The default thresholds of 0.9 for spam mails and 0.2 for ham mails classified a large proportion of mails as *unclear* (i.e. neither ham nor spam, undecided). A threshold of 0.5 is a sensible a priori choice for training sets with equal number of ham and spam mails (SH_{ratio}=1), and was therefore initially adopted. *SpamBayes* is based on ideas by [4], and is a Naive Bayes learner. It is the simplest system presented here.

Naive Bayes learning in this domain roughly works as follows: Split each mail into a set of words via a tokenizer (for mail headers and body separate tokenizers are usually used), and count 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 (e.g. by frequency with Laplace correction to prevent the occurrence of zero probabilites, or with more complex formulas). 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). In *SpamBayes*, ham and spam probabilities are combined via an approach using chi-squared probabilities which leads to more robust estimates than simple renormalization.

Since the previously mentioned systems turned out to be quite similar to each other, we have adapted the threshold to each mailbox (for *byMBXTrain*), as well as one uniform threshold for the model trained by pooling data from all mailboxes (*allTrain*). *SpamBayes* performs similar to the other systems after this normalization while before it performed significantly better in FP rate and significantly worse in FN rate. Among the eleven thresholds 0.0-1.0 (in steps of 0.1), we chose the one that minimized mean squared error versus the mean of the average performance (over ten runs) from both *SA-Train* and *CRM114*. The chosen threshold was 0.1 for *allTrain*; and 0.1, 0.4, 0.3, 0.4, 0.0, 0.0 and 0.3 for mailboxes #1-#7 and *byMBXTrain*. FP and FN rate MSE were weighted equally. Using mean absolute deviation yields the same thresholds. These thresholds were used for all experiments except where otherwise noted.

5 Mailbox Collections

The collection and verification of spam is time-consuming and error-prone – e.g. error rates for manual verification of 0.45% (true ham), and 0.64% (true spam) mails were reported by [2] for a SH_{ratio} of 4 and a corpus of 50,000 mails. 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 created mailboxes #1-#7 by merging mails recently predicted as spam by SpamAssassin 2.63 (Default settings without bayesian model)[2] with spams collected and submitted by the users themselves over a short time span (2-3 weeks), separately for each user. SpamAssassin markup and headers were removed. The ham mails, on the other hand, were collected from past ham mails within the user's mail archive. We removed all mails which were sent by the user himself since these are not incoming mails and therefore not appropriate for training (except those also sent to the user via Cc or Bcc). The structure of ham mails is likely to change less over time than spam mails (i.e. less concept drift), 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 a typical user at our institute experiences in his daily life (except #8, see below). Anecdotal evidence indicates that this approach does indeed lead to well-performing models.

We have taken an effort to clean all the mailboxes, inspecting every training set error of an earlier learning system to see whether the mail was assigned the wrong category. An overview of the mailboxes, including date ranges for ham and spam mails, is in Table 3. As you can see, the mailboxes are somewhat spread out in time, and while models trained using this data work very well, we cannot investigate concept drift issues here as the collection of these mails does not accurately reflect a true second-by-second snapshot. On the other hand, second-by-second snapshots of arriving mails would have to be collected over several months by dozens of users to get as many ham mails as we have collected here, which would have been infeasible for us.

[2] 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 large spam collections.

To be able to investigate time-dependent performance, we have collected another mailbox, #8, which 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 the Medical University in Vienna, Austria. Each mail was checked at least once. The SH_{ratio} of 16.4 accurately reflects the proportion of spam by ham mails received by the author during this period.

We have taken the view of using a very broad spam and ham definition by letting each user choose which mails to classify as spam and which to classify as ham. A close inspection of the chosen mails shows that spam includes not only Unsolicited Bulk Email, but also Phishing mails (which aim to collect credit-card or personal information for fraud and identity theft), virus-infected mails, Mail Delivery Errors for mails which originated elsewere but were sent with a fake From address and so on. Our ham mails are similarily broad, including newsletters, wanted advertisements, mails with large attachments – including those without text and those with non-virus-infected[3] executable attachments, confirmation mails resultings from registering a webmail or mailing list service, status messages from online merchants such as Amazon, Mail delivery error messages concerned with previously sent legitimate mails, vacation auto-replies sent as response to legitimate mails and so on. This makes the filtering task sufficiently realistic to be of broad interest. No type of mail that users wanted to receive has been explicitly excluded.

We have decided against using publicly available corpora of spam and ham mails. Most current corpora are also not large enough to be of use. In some cases, their spam definition is very narrow, and the ham definition is similarily restricted in that no personal and professional mails are included due to confidentiality issues. In that respect our mailboxes can be considered more

[3] Virus detection due to Symantec BrightMail 6, during experiments described in [13].

representative than public collections.

6 Experimental Setup

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 more realistic challenge. We were motivated to use a training size with SH_{ratio}=1 since most learning systems are sensitive to non-uniform class distributions. A ratio of 1:1 for training is also proposed as preferred class distribution in the documentation of all investigated systems. Our approach is equivalent to undersampling the larger class, followed by one half of a two-fold cross-validation – provided the mails removed during undersampling are added to the test set.

Two training sets were used: For *byMBXTrain* each mailbox was trained separately and tested on the remaining mails. For *allTrain* the training sets from all mailboxes were combined into a single training set of SH_{ratio}=1, and all the remaining ham and spam mails were used for testing. The test results are reported for each user separately, except where otherwise noted.

Training and testing was repeated ten times with different random orderings of the mailboxes' mails, unless otherwise noted. Average and standard deviations of FP and FN rates per mailbox are reported.

All of our learning systems can be made to output confidence values for each test mail. The default threshold applied to these confidence values determines the performance of the system. A threshold-independent way of showing the performance of the system at all thresholds in one glance is a FP/FN rate trade-off curve. This is essentially a scatter plot of FP_{rate} vs. FN_{rate}, where each threshold determines a different data point. For better visualization, 2000 threshold values were uniformly sampled from the range of confidence values in the data, and neighboring points were connected, yielding a curve. The FN rate at given acceptable FP rate can thus be determined from the curve. This visualization is similar but not identical to a ROC curve, which would use *TP* instead of *FN* and no logarithmic scales. However, the use of a logarithmic scale is essential as otherwise the systems could not be distinguished. *TP* with an logarithmic scale would have given less weight to important differentiations (e.g. between 99.9% and 99.99% TP rate which corresponds to 0.1% and 0.01% FP rate) and was therefore replaced by FP_{rate}. The same argument holds for absolute TP (i.e. **a**).

7 Results

7.1 byMBXTrain

Here we report the results of the evaluation where each mailbox was trained and tested separately (*byMBXTrain*). The motivation was to see which of two factors – learning system and mailbox – have a higher influence on system performance. The hypothesis was that the mailbox has a higher influence.

As we can see in Figure 1 and Table 4, all three systems perform somewhat similar. Only a few significant differences (i.e. non-overlapping errorbars) can be found for systems trained on the same mailbox. The average performance is also quite similar, except for *SpamBayes* which has only half the FP rate but double the FN rate. This is likely to be an artefact of the coarsely chosen tresholds for normalization versus the other two systems, and does not signify a superiority of *SpamBayes*. The differences between mailboxes are much larger than the differences between the three learning systems, and the standard deviation over the runs (shown as error bars in Figure 1) is also quite high, which explains why there are few significant differences between systems on the same mailbox. The latter may reflect the small amount of training data that is available for each run, especially for the smaller mailboxes.

Concluding, in this experiment the differences between the mailboxes are much larger than the differences between the learning system, so there are both easy and hard mailboxes for this task. The differences between the systems, however, is small: only in few cases is this difference significant. The initial hypothesis could be confirmed, which indicates that studies on a single mailbox may not be sufficient.

7.2 allTrain

Here, we evaluated the performance of a combined system which was trained on pooled data from all mailboxes (*allTrain*). The motivation was to see which of two factors – learning system and mailbox – have a higher influence on system performance. The hypothesis was that the mailbox has a higher influence.

Pooling data from all mailboxes to train a single model (*allTrain*, Figure 2, Table 5) reduces FP and FN rate and also reduces the variation over runs (except for mailbox #5). Although it is generally believed that models which are trained for each user separately yield the best results, these results demonstrate conclusively that combining training data is beneficial for at least a small number of users. Anecdotal evidence suggests that a pooled model also offers better performance when classifying mails for a previously unknown user (similar FP rate and better FN rate was usually observed) than a model trained for an arbitrary user. This can be attributed to the greater diversity of training inputs which improves generalization performance.

According to anecdotal evidence, *SA-Train* systems trained on pooled data seem to deteriorate slightly faster over time than per-user models, for some users. This does not seem to be the case for *SpamBayes* systems trained on pooled data (see Section 8.2).

Pooling data also enables us to determine smaller FP rates (down to 10^{-4}) than would have been possible with utilizing per-mailbox training and testing. Additionally, a larger sample of diverse mailboxes makes it more likely that our results hold generally, rather than being an artefact of a single user's mailbox.

Concluding, the mailbox is still a factor in system performance, although the mailboxes now give more similar results as for *byMBXTrain*, and the differences between learning systems have become less significant. This suggests that the systems are able to learn a more balanced model from pooled data. In neither *allTrain* nor *byMBXTrain* could an advantage for the extensions of Naive Bayes learning (*SA-Train* and *CRM114*) be found.

7.3 Class Noise Level

Noise level susceptibility is an important property of spam filtering systems as it determines the effort needed to clean the mailbox prior to training. A low noise level susceptibility is a desired property of filtering systems. Here, we have set out to determine the noise level susceptibility of our three learning systems.

The noise level experiments were conducted by permuting the training sets from each mailbox separately. Since each training set contains roughly the same number of ham and spam mails, we have chosen to invert the classification for the same number of ham and spam mails so that the total number of mails with wrong classification equals the desired noise level (half are spams with assigned ham label, and half are hams with assigned spam label). Therefore we focus on class noise where additionally the errors are equally distributed between ham and spam mails. Again, each mailbox and noise level experiment has been run ten times, and the average and standard deviation over these runs are shown.

Figure 3 and 4 show the results. In most cases, increasing the noise level also increases the FP rate, often in at least linear fashion. The same effect on FN rate is also present but less pronounced. In most cases *SpamBayes* breaks this trend and is even at first glance much less susceptible to noise. Since *SpamBayes* is also the only system that does not use TUNE-like training procedures and the pattern is otherwise similar between *CRM114* and *SA-Train* (two very different systems), this is suggestive of a link between TUNE-like training and increased susceptibility to class noise.

This is supported by the data: the number of cycles for TUNE increases monotonically for all mailboxes in relation to increasing noise level (on average by 62% for *SA-Train*, and by 16% for *CRM114*, both from 0.0% to 2.4% noise (data not shown)). This indicates that noisier data consistently needs more cycles to attain convergence, and is therefore more susceptible to worse performance due to overfitting the noisy training data. TUNE training repeats the training of mails which were misclassified by the system several times. In case the learning system works reasonably well, mails with noisy classification will tend to be misclassified disproportionately often, and thus often retrained. This biases the model and leads to worse performance. It is still surprising that the early exit criterions of *SA-Train* and *CRM114*, and the fact that misclassified mails are only trained once in *SA-Train*, did not help either *SA-Train* or *CRM114* to attain the same robustness as *SpamBayes*.

Anecdotal evidence also suggests that training set errors of at least *SA-Train* are indeed good indicators for incorrectly labeled mails, and consequently training set errors encountered while training preliminary mail collections were usually improperly labeled.

Concluding, the noise level susceptibility is high for systems with TUNE-like training (*SA-Train* and *CRM114*) and low for systems with normal training (*SpamBayes*). TUNE-like training approaches seem to be far more susceptible to class noise, which can be explained by the nature of TUNE training, as well as being confirmed by the number of training cycles which increase monotonically in lockstep with increasing noise level.

7.4 FP/FN rate Trade-Off curves

Up to now, we have compared the overall performance of each system at a single point, and found few significant differences between systems on the same mailbox. However, it is clear that FP rate can be traded off versus FN rate to some extent. This is most obvious for systems which already incorporate a threshold parameter. E.g. a threshold of 0.5 for *SpamBayes* gives 0.07% FP rate and 1.9% FN rate – roughly a tenth of FP rate and three times FN rate versus a threshold of 0.1. All the systems tested can be made to output confidence values that may be tresholded at different values rather than the default. A useful way to visualize performance at different thresholds is to plot FP rate versus FN rate. The FN rate at given acceptable FP rate can be determined from this plot. This approach allows us to observe differences between the systems which are not visible by comparing the performance at default thresholds. Our hypothesis here was that, again, there would be no differences between the learning systems.

To have sufficient data for testing, we only show results for the pooled training sets (*allTrain*), obtained by storing the confidence for the classification of each test mail (only the first run of *allTrain*). Since we already know that the mailbox is a major factor in system performance, pooling the test data essentially abstracts from this factor and enables us to investigate results that are independent of specific mailboxes. From Section 7.2 we already know that the performance of the *allTrain* model is quite similar over the mailboxes, so the combination should not increase the variance too much. We are aware that the variance of these estimates may be high because only a single run of *allTrain* is used, but there is no simple way to combine the estimates from different runs of *allTrain* as in each run different sets of mails are tested. A cross-validation is also not possible because of the 1:1 sampling approach which breaks the essential symmetry of CV. However, we believe that the large test set ensures that the differences observed here would also be apparent with a different random ordering of the input data. We will henceforth call this kind of visualization a FP/FN rate trade-off curve, or shortly trade-off curve.

Figure 5 shows the trade-off curve for the three tested systems. Each point on the curve corresponds to a FP (X axis) and FN (Y axis) rate that can be achieved by choosing a specific threshold. As can be seen, at the estimated human error rate of around 0.3-1% [2] *SA-Train*, *CRM114* and *SpamBayes* perform very similar. For smaller FP rates, *SpamBayes* and *SA-Train* perform similar while the FN rate for *CRM114* increases significantly. FP rates of smaller than around 10^{-4} are not achievable with *SpamBayes* since many mails have a score of exactly 0 which prevents further trade-off of FP against FN rate in this area.

[2] found that different internal thresholds explains most of the differences between spam filtering systems. Contrary to his findings,we already observed very little differences at default thresholds in the previous sections (after *SpamBayes* normalization), but the trade-off curve has enabled us to find a previously not noted difference: At one glance, we see that *CRM114* is only competitive to the other two systems for a small range of thresholds including the default threshold of 0 for that learner. Our initial hypothesis – that we would again

find no differences between systems – has been disproved. On the contrary, we found that one of the investigated extensions to Naive Bayes, *CRM114*, actually performs worse than its ancestor.

7.4.1 Training procedures

The *TUNE* (Train-Until-No-Errors) training procedures we used up to now for *SA-Train* and *CRM114* are rather costly. It remains to be investigated whether the same results could be obtained with less training. Also, in the light of higher noise-level susceptibility of TUNE training, we might end up with a faster *and* better model. So we investigated two ways to reduce the computational effort of TUNE as well as make it less susceptible to overfitting. Additionally, since our main motivation was to check whether the learners *SA-Train* and *CRM114* have improved on *SpamBayes*, we needed to make sure that the similar results of the three systems were not due to the difference in training methodology.

TUNE training involves repeated training of training set errors until convergence. On simplification is therefore learning the training set errors only once which was already named Train-On-Error, or *TOE*. This is equivalent to a single cycle of TUNE. The order of the mails in the training set is arbitrary but fixed. TOE is somewhat related to incremental training of spam filters (where the mails would be in chronological order) and thus to a widespread method to train spamfilters incrementally. For *SA-Train*, the SVM weights need to be learned twice: at the beginning to get the full model in order to determine the training set errors, and after NB learning to get a weight vector for the final model. TOE alone should be around an order of magnitude faster as the maximum number of cycles for TUNE was set to 15.

The other was learning *all* mails and not only the training set errors, which we called *simple*. For *SA-Train*, the full training set was learned via NaiveBayes, and afterwards SMO was applied. In that case, the initial NB model was not needed as for all mailboxes more than 200 training mails were available. It should be noted that this simpler system was actually at least two orders of magnitude slower for *CRM114*, since in that case training is much more costly than testing, and TOE reduced the number of mails to be trained by two orders of magnitude. For *SA-Train*, *simple* was only slightly slower than TOE.

Figure 6 shows the influence of the training method on the trade-off curve for *CRM114* and *SA-Train*. As can be seen, the influence on *CRM114* is large, with the curve almost steadily improving from simple to TOE to TUNE. Surprisingly, the effect on *SA-Train* is quite small (except for high FP rates greater than 10%), so the *simple* training method seems to be sufficient for *SA-Train*.

Concluding, our results disprove the hypothesis that *SA-Train* and *CRM114* using simpler training methods would outperform *SpamBayes*, since in all cases the performance when trained with simpler methods did not improve significantly. The results also support [17] finding that TUNE and TOE improve performance of *CRM114*. However, this is not the case for *SA-Train* so TUNE and TOE seem to be less useful as a general technique. In the light of higher noise-level susceptibility, care should be taken when applying the TUNE training procedure, and comparison to simpler training methods is essential.

8 Concept Drift

Up to now, we have mainly ignored concept drift, i.e. the fact that the true model generating ham and spam mails changes over time. We have argued that the mail collection ensures datasets with little concept drift (except mailbox #8). We also note that the error for *SpamBayes* on the representative mailbox #8 (averaged over first two weeks: FP_{rate}=0.532%, FN_{rate}=0.956%) is within the 99% confidence interval of the earlier time-independent estimate (see Table 5, Avg. ± 2*stDev). This is also the case for the FN rate of *CRM114* and

SA-Train (both FP and FN rate), but not for the FP rate of *CRM114* which is one order of magnitude lower versus mailbox #8. We may tentatively conclude that time-invariant error estimates from our mailboxes #1-#7 are usually valid on new mails – even from an unseen but related mailbox – for at least a limited time period.

Concept drift must be analyzed over longer timespans. E.g. in [13] we found that performance estimates of current ready-to-use systems are only valid with reasonably recent mails, and that the FN rate increases sharply for older mails – which reflects the major effort of the system's developers to account for current spam mails at the expense of historical spam mails. This effort can also be seen in the enormous bandwidth requirements: for Symantec BrightMail, about 700 megabytes of updates were received weekly during our three-week experiments in [13]. The usual tendency for learning systems is just the opposite: Spam classification tends to get harder in time, once the model has been trained and remains fixed.

We used mailbox #8 (see Table 3) for these experiments. While #8 on its own is not large enough for training because it contains too few ham mails (only 1387 vs. 27163 mails for the full training set used here – see below), it is sufficient to test two aspects of our trained systems: One, we can determine how well the system generalizes to an unknown mailbox as no mails from the user corresponding to #8 were used during training (except for the initial model of *SA-Train*). Two, we can determine how fast the trained systems deteriorate over time. Additionally, we can evaluate continuous learning approaches by simulating different ways to learn the incoming ham and spam mails in chronological order, and compute FP and FN rate for the continuous learning system as it evolves over time.

One disadvantage of this approach is that we have to check two different effects at once. However, as we already noted, the time ranges of spam and ham mails within each of our collected mailboxes are too different to be used for that purpose. Pooled mailboxes (*allTrain*) cannot be used, since the earliest ham mails from the pooled mailbox would come mainly from one or two users. An additional complication is the 1:1 sampling, which would force us to split ham and spam mails each into a different number of folds. This would complicate training, testing and combining the results. We note that our intention was to create a trained model that generalizes well to unknown users, and therefore the more challenging analysis of two meshed effects is in our interest.

#8 is both temporally separated as well as collected from a different user as #1-#7. Each learning systems was trained on mailboxes #1 to #7 except the ham mails from #1 which were removed to obtain a SH_{ratio}=0.87 and thus reasonably near 1.

8.1 Deterioration over time

Here we focus on the deterioration of FP and FN rate over time for each learned model. This is an important property of learned spam-filtering systems, as it determines the effort that is needed for batch or continous retraining to sustain a desired level of filtering performance indefinitely.

While we could not find any advantage for the more complex learners until now, it could be that *SA-Train* and/or *CRM114* offer more stable performance over time and thus deteriorate less even though the absolute performance and trade-off curves are remarkably similar to a large extent.

8.1.1 Weekly FP/FN rate

Deterioration can be measured by weekly FP/FN rate, but also by comparing the FP/FN rate trade-off curve from the first half and the second half of the timespan corresponding to #8. We will compute and contrast both measures as well as comparing them to the FP/FN rate trade-off curves from *allTrain* evaluation.

Figure 7 shows the FP and FN rates over time for each model. The FP rate shows no degradation, but fluctuates quite strongly. By FP rate, *SA-Train* is best, followed by *SpamBayes* and *CRM114*. As can be seen, *SpamBayes* shows the smallest FN rate degradation over time. This indicates that *SpamBayes* has learned a model of spam which degrades very slowly. *SA-Train* performs very well in FP rate and quite good in FN rate initially, but the latter degrades rapidly. Retraining at around week 14 would clearly be necessary here. *CRM114* performs badly in FP rate but does not degrade, and its FN rate starts low but degrades faster than the one by *SA-Train*. This hints that the default treshold might not be appropriate at least for *CRM114*, so we continued the analysis with trade-off curves.

8.1.2 FP/FN rate trade-off curves

We computed the curves over the first ten weeks, and over the remaining nine weeks. Additionally, we added the curves from Section 7.4 to allow a comparison to the earlier time-independent estimate. Note that we would expect the new estimates to perform worse, as no mails from the user corresponding to #8 were part of the training set (except for *SA-Train*, where a small set of earlier mails from the same user as #8 were used for the initial Naive Bayes model). This is the cost of generalizing to previously unseen mailboxes.

Figure 8 shows the trade-off curves for our three learning systems. The good agreement between the *allTrain* curve and the curves estimated from #8 for *SA-Train* is most likely due to the initial Naive Bayes model of *SA-Train* which was partially based on user #8's mails. For the other models, the agreement is less good. For all systems but *SpamBayes*, the curve over weeks 10-18 performs worse than the one from weeks 0-9 which is to be expected if the techniques used by spammers succeed in defeating spam filter systems. Most surprising is that for *SpamBayes*, additionally to showing little effect of time on the FN rate, the FP/FN rate trade-off curve actually improves over time, hinting that the learned model is exceptionally resistant to changes in spam mail distribution. Since *SpamBayes* is also the simplest system we have tested, this is strongly suggestive of overfitting in the two other systems. We note that both for *SA-Train* and *SpamBayes*, the curve estimated on the time-dependent data is shaped differently than the one from *allTrain*: when we compare the area between 0.001 and 0.1 FP_{rate}, it is convex rather than concave. For *CRM114*, the same curve appears but is shifted along the FP rate axis which corresponds to a geometric increase in FP rate that is also observed in the time-dependent estimate of classification performance (see Section 8, first paragraph).

8.1.3 Training procedures

As would be expected from the results in section 7.4.1, the training procedure has little influence on *SA-Train*, and the simplest training methodology already works reasonably well. However, the trade-off curve of training methodology *simple*, which is the only one that does not depend on the initial Naive Bayes model which contains earlier mails from the same user as #8, is noticeably worse, which confirms our earlier suspicion that *SA-Train*'s good agreement is due to this fact. The other curves are very similar over the different training methodologies. This confirms that training methodology has less influence on *SA-Train*.

For *CRM114*, the shape of the FP/FN rate trade-off curves for the three training methodologies are similar to those curves computed via *allTrain*, provided we again abstract from the worse performance (i.e. the shift along the FP rate axis). The classification performance at the default threshold for training procedures *TUNE* and *TOE* is again similar, but *simple* yields a FN rate rate in excess of 80% and a very low FP rate rate. This indicates that the default threshold may be inappropriate in that case, or that for *CRM114-simple*, the training data needs to be very near a SH_{ratio} of 1 (the SH_{ratio} for the training data in this subsection was 0.87). Data is this section is not shown.

8.1.4 Conclusion

Concluding, the results of time-independent evaluation still apply to time-dependent evaluation for at least an initial time period. The trade-off curves are shaped differently (convex vs. concave), which is not due to the expected deterioration in applying the model to an unknown user. For *SA-Train*, the initial model included mails from the user of #8, and the curve was correspondingly more similar but still convex. We speculate that the concept drift is biasing the curve towards higher convexity in time rather than shifting it, but our data is insufficient to show this clearly. *CRM114* appears to react differently: it shows a clear horizontal shift for *allTrain* to mailbox #8 (i.e. evaluation on an unseen user's mailbox), and a vertical shift for the first nine weeks to the last ten weeks. In each case, *SpamBayes* – rather than the more complex systems – deteriorates less over time. Thus, again we find no advantage in using the more complex systems.

8.2 Continuous (incremental) training

As an encore we tested several continuous training methodologies. Since both *SA-Train* and *CRM114* fail to improve on *SpamBayes*, we only tested how the *SpamBayes* learning model can be incrementally updated. This is an alternative to retraining every few months, i.e. the batch-style training which has been used throughout this paper. While batch-style training combined with our approach to mail collection is very useful to create an initial model, incremental learning is useful to make spam filtering updates more timely and less costly. Instead of batch-training a new model every few months, continuous learning trains misclassified mails incrementally as they appear. This is reminiscent of TOE training, and also a common approach to train spam filters incrementally.

We have chosen to evaluate only the best tested system, *SpamBayes*, on independent test data from mailbox #8. As can be seen from Table 3, #8 is both temporally separated as well as collected from a different user as #1-#7. *SpamBayes* was initially trained on mailboxes #1 to #7 except the ham mails from #1 which were removed to obtain a SH_{ratio}=0.87. An evaluation of the two other systems was deemed to be unnecessary: *SA-Train* is expected to perform very similar to *SpamBayes*, and *CRM114* is not expected to perform better than *SpamBayes* either and is furthermore only competitive for a small range of thresholds.

This focus on a single system allows us to investigate continuous learning in detail: Complementing the batch-style training and test which we have used up until now, we shall now compare several variants of Train-On-Error to see whether such a continuous training approach biases FP and/or FN rate. We will investigate three such variants, or settings.

- *AlwaysTOE* trains each FP and FN error instantly after the misclassification appears. This is optimal from point-of-view of the learner, but is rather costly since all mails classified as spam have to be checked instantly.

- *NoHamTOE* trains only the FN errors. This reflects the behaviour of users which do not look through their spam folder at all and so miss all false positives – a common behaviour for users of spam filtering systems, and also a desirable behaviour from an application point of view since it does not involve continuously checking the spam filter.

- *None* does not train any errors at all. This is even more desirable than *NoHamTOE*, but may work only for a limited period.

In all three cases, the mails from #8 are presented in the order in which they were initially received. The FP and FN rates are averaged over full weeks to prevent artefacts due to weekday-dependencies. We will focus on the changes in FP and FN rate due to these different continuous learning settings to see what effect they have in practice.

Figure 9 shows the results. As would be expected, *AlwaysTOE* has the lowest FN and FP rate overall. However, even *None* manages to keep a FN rate of smaller than 2% over the full 18 weeks while the FP rate fluctuates wildly around the average of 2%. Note that *NoHamTOE* has only a small influence on the FP rate which is over most stretches exactly the same as for *None* while it improves the FN rate almost as much as *AlwaysTOE*, so it seems that *NoHamTOE* combines the best of both approaches. The high FP rate is rather unsatisfactory, and so we repeated the experiment with a threshold of 0.5 for *SpamBayes* – a reasonable setting for a training file with approximately equal numbers of ham and spam mails – which strongly reduces FP rate at the cost of FN rate.

Figure 10 shows the results. Now, all three continuous training settings have exactly the same FP rate. Note that only two ham mails were misclassified in week 9 by all systems, so over the time period there seems to be no deterioration of ham error – with or without training, and even when training only spam errors, of which there are a few hundred. This is an excellent result.

The FN rate, however, deteriorates fast for the *None* model, but stays about constant at around 2-3% for *AlwaysTOE. NoHamTOE* almost exactly matches *AlwaysTOE* and thus overlaps in the figure. In short, *NoHamTOE* seems to be the best continuous training setting provdided the learner achieves a sufficiently small FP rate. A high FP rate (as in the previous example with threshold=0.1) will deteriorate much faster and necessitates looking at each predicted spam mail. We propose utilizing other approaches to estimate FP rate regularily to prevent an deterioration over longer periods.

8.3 Other learning methods

[2] is one of the most comprehensive empirical studies on spam-filtering systems to date. They cite six spam-filtering systems: SpamAssassin, *CRM114*, *SpamBayes*, BogoFilter, DSPAM and SpamProbe. Of the three systems we have not tested here, BogoFilter and SpamProbe are Bayesian filters inspired by [4], which has also inspired *SpamBayes*. DSPAM and *CRM114* are found to perform substantially inferior to the other filters (when training and classifying mails in sequence). The latter is compatible with our findings. They conclude that *The potential contribution of more sophisticated machine learning techniques to real spam filtering is as-of-yet unresolved*, which is also our opinion: Overall, there are some promising papers (e.g. [10],[3],[12]) but none of them has yet been translated into a state-of-the-art spam filtering system of non-Bayesian origin. Even commercial systems fail to beat Naive Bayesian learners such as *SpamBayes*, see [13].

9 Conclusion

We have evaluated two extensions of Naive Bayes learning (*SA-Train*, *CRM114*) as well as a simple Naive Bayes learner (*SpamBayes*), on a set of around 65,000 ham and spam mails collected from seven users, plus a set of ham and spam mails from an eighth user. We draw the following conclusions from our experiments. Conclusions 1-3 are of a practical nature. They indicate what we have learned concerning how to train well-performing spam filters with minimal effort. Conclusions 4-6 are related to the TUNE training procedure (due to [17]). Conclusion 7 states specific weaknesses of both *SA-Train* and *CRM114*, and Conclusion 8 states specific weaknesses of *CRM114* which we have found during our investigations.

1. Our approach to mailbox collection – combining a set of recent spam mails with historical ham mails – allows fast creation of large mailboxes for spam filter training. These seem to contain very little concept drift and are very useful for training any learning spamfilter among those investigated with small differences in performance (at default thresholds) and the trained models are stable over several months. By Ockham's Razor, the simplest system, *SpamBayes*, should be preferred.

2. For high ratios of spam to ham mails, representative mailboxes will have a very biased class distribution. Our approach of 1:1 sampling has proven to yield very good spam filters even when the true class distribution is very biased (e.g. 1:16.4 was observed by the author for #8). Our approach to mail collection (see above) allows control over training set SH_{ratio} and a value of 1 should therefore be aimed for.

3. Pooling multiple mailboxes for training reduces the variance of estimates and does not reduce performance significantly. In some cases, the performance of the combined system is even better than that of a model trained for a specific user. The generalization to unknown users may also be improved through higher diversity of the training data.

4. The TUNE training approach works well on *CRM114* for which it was developed and improves classification performance as well as the shape of the FP/FN curve.

5. For *SA-Train*, TUNE training was found to have only marginal influence

on classification performance and the shape of the FP/FN curve while strongly increasing runtime. Thus, TUNE should not be expected to generally improve performance.

6. TUNE training for both *CRM114* and *SA-Train* increases the susceptibility to class noise in training data, which is to be expected as TUNE trains each erroneously classified mail repeatedly. *SpamBayes*, which does not use TUNE training, shows no such degradation. The better performance of *CRM114*, and the similar performance of *SA-Train*, is thus offset by the higher error rates in the presence of noise. This pattern is apparent on almost all mailboxes.

7. Both *SA-Train* and *CRM114* show a marked deterioration in spam error rate over time. *SpamBayes* does not. In this case, the TUNE training procedure is not responsible for the worse performance of *SA-Train* and *CRM114*. We suspect that the higher number of parameters available to both systems leads to overfitting and thus to a less stable model which is compromised faster. The deterioration of *SpamBayes* may only be clearly apparent over a longer period than the 18 weeks which we have studied.

8. When comparing classification performance, all three tested systems perform equally well (after normalization of *SpamBayes* – before normalization *SpamBayes* performs uniformly better in FP rate and uniformly worse in FN rate). However, a look at the performance dependence on the threshold level shows that *CRM114* is only competitive to *SA-Train* for a small range of thresholds which include the default one. The *SpamBayes* curve is similar to the *SA-Train* curve but is cut off at a FP rate of 10^{-4}. All these observations are also valid when *SA-Train* is trained in a more simple way similar to *SpamBayes* (i.e. without TUNE or TOE).

Overall, the addition of background knowledge to a Naive Bayes learner, *SA-Train*, as well as the extended description language of *CRM114*, has failed to improve Naive Bayes learning significantly, even with refined training methods such as TUNE and TOE. Rather, the perceived similarity to the Naive Bayes learner *SpamBayes* is highly suggestive and indicates that the NB learner is responsible for almost all of the classification performance of *SA-Train*. For *CRM114*, the extended description language (phrases instead of words) has had negative impact on performance for a large range of thresholds. The much higher deterioration over time of both systems vs. *SpamBayes* may be due to overfitting, since a much higher number of parameters need to be fitted from the same data (both the NB model's probabilities and the rule scores; resp. probabilities for all phrases up to a certain length rather than words).

The similarity between the three tested systems, taken together with the findings of [2] who found very similar performance on a larger set of spam filtering systems, indicate that content-based learning approaches may well have reached a ceiling. More complex approaches fare worse than simple approaches in noise-level susceptibility and deterioration over time while their performance in a time-independent evaluation is remarkably similar. Even commercial systems that process several million EMails per month such as Symantec Bright-Mail do not offer a better performance ([13]).

Additional improvement may be expected from other sources of information apart from content, e.g. behaviour-based filtering approaches which take the behaviour of the sending mail server into account. Combining these and other information sources into a single learning system might improve the filtering performance beyond the current ceiling. However, extended corpora need to be created for this purpose which is likely to be another major effort.

10 Acknowledgements

I would like to thank my previous affiliation, the Austrian Research Institute for Artificial Intelligence, for giving me the opportunity to work on this and other personal research projects. The Austrian Research Institute is supported by the Austrian Federal Ministry of Education, Science and Culture and by the Austrian Federal Ministry for Transport, Innovation and Technology. I want to thank my anonymous colleagues for contributing the mail collections without which this work would not have been possible.

References

[1] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras and C.D. Spyropoulos, 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. G. Potamias, V. Moustakis, M.n. van Someren (eds.), Barcelona, Spain, 2000.

[2] G. Cormack and T. Lynam, A Study of Supervised Spam Detection Applied to Eight Months of Personal Email, `http://plg.uwaterloo.ca/~gvcormac/spamcormack.html`, July 2004. To be published in a revised and extended version in ACM Transactions on Information Systems, 2006.

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

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

[5] J.M. Gómez Hidalgo, M. Maña López and E. Puertas Sanz, Combining Text and Heuristics for Cost-Sensitive Spam Filtering, in: Proceedings of the

Fourth Computational Natural Language Learning Workshop, CoNLL-2000, Association for Computational Linguistics, Lisbon, Portugal, 2000.

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

[7] T.M. Meyer and B. Whateley, SpamBayes: Effective open-source, Bayesian based, email classification system, in: Proceedings of the First Conference on EMail and Anti-Spam (CEAS), Mountain View, California, United States, 2004.

[8] J. Platt, 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.

[9] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C.D. Spyropoulos and P. Stamatopoulos, Stacking Classifiers for Anti-Spam Filtering of E-Mail, in: Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Morristown, Pittsburgh, US, 2001.

[10] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos and P. Stamatopoulos, A memory-based approach to anti-spam filtering for mailing lists, Information Retrieval Journal, 6(1), 2003.

[11] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, A bayesian approach to filtering junk e-mail, in: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[12] A.K. Seewald, F. Kleedorfer, Lambda Pruning - An Approximation of the String Subsequence Kernel. Technical Report, Austrian Research Institute for Artificial Intelligence, Vienna, TR-2005-13, 2005.

[13] A.K. Seewald, A Close Look at Current Approaches in Spam Filtering, Technical Report, Austrian Research Institute for Artificial Intelligence, Vienna, TR-2005-04, 2005.

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

[15] S. Gauthronet and E. Drouard, Unsolicited commercial communications and data protection, Technical report, Commission of the European Communities, 2001.

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

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

11 Tables

Table 1: Summarization of classification decisions as contingency table.

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

Table 2: Learning systems by learning algorithm and feature set.

System	Algorithm	Feature set
SA-Train	SMO, Naive Bayes	bag-of-words (NB), rule outputs (SMO)
CRM114	SPBPH w/ Bayesian Markov Model	implicit bag-of-phrases (with wildcards)
SpamBayes	Naive Bayes	bag-of-words

Table 3: Mailbox corpora used for evaluation.

mbox	number of mails		SH_{ratio}	received during	
no.	Ham	Spam		Ham	Spam
#1	15248	3319	0.21	12/88-07/04	01/97-07/04
#2	8982	10605	1.18	01/02-09/04	02/02-09/04
#3	3608	568	0.15	09/97-06/04	06/04-06/04
#4	2167	1123	0.51	04/96-06/04	06/04-06/04
#5	1589	3083	1.94	07/02-07/04	02/03-07/04
#6	7539	1838	0.24	09/99-07/04	05/04-07/04
#7	3278	3229	0.98	06/01-06/04	06/04-06/04
#8	1387	22795	16.43	10/04-02/05	10/04-02/05

[4]

[4] SH_{ratio}: Ratio of spam to ham mails. Dates were reconstructed from Received: Headers and may not be accurate for spam mails – Spam mails were collected over short time spans (at most 2-3 weeks), and the date ranges do not always reflect this correctly.

Table 4: Full results of learning systems, training method *byMBXTrain*.

mbox	SA-Train		CRM114		SpamBayes	
no.	FP_{rate}	FN_{rate}	FP_{rate}	FN_{rate}	FP_{rate}	FN_{rate}
#1	0.720	0.796	0.768	0.766	0.538	3.551
#2	0.187	0.317	0.528	0.251	0.343	0.185
#3	1.854	2.047	0.850	0.706	0.012	1.270
#4	0.690	1.195	0.536	0.785	0.240	0.553
#5	1.538	1.791	1.563	2.047	0.643	6.235
#6	0.746	1.916	0.840	1.219	0.817	1.111
#7	0.330	0.799	0.586	0.899	0.146	0.793
Avg.	0.866	1.266	0.810	0.953	0.391	1.957
StDev.	0.611	0.665	0.360	0.561	0.287	2.179

[5]

[5] 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 $\frac{1}{561}$.

Table 5: Full results of learning systems, training method *allTrain*.

mbox	SA-Train		CRM114		SpamBayes	
no.	FP_{rate}	FN_{rate}	FP_{rate}	FN_{rate}	FP_{rate}	FN_{rate}
#1	0.701	0.627	0.507	0.663	0.950	0.531
#2	0.154	0.275	0.198	0.602	0.163	0.238
#3	0.278	1.023	0.185	0.282	0.000	0.635
#4	0.308	0.892	0.129	0.393	0.111	0.464
#5	1.374	1.116	1.827	0.987	1.386	0.965
#6	0.964	1.002	0.648	0.566	0.764	0.730
#7	0.269	0.731	0.378	1.122	0.110	0.781
Avg.	0.578	0.810	0.553	0.659	0.498	0.621
StDev.	0.454	0.291	0.593	0.302	0.536	0.236

[6]

[6] 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 $\frac{1}{561}$.

List of Figures

12 Figures

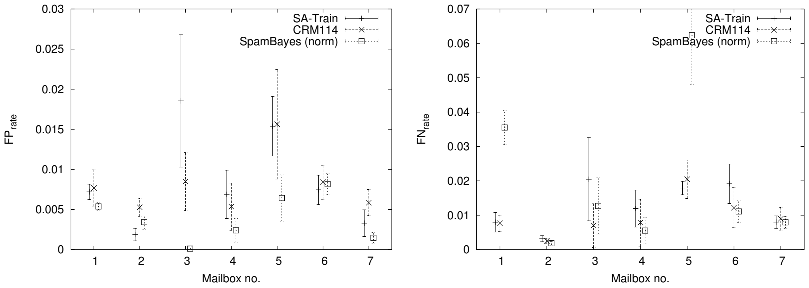

Figure 1: This figure shows FP_{rate} (on the left) and FN_{rate} (on the right) for mailboxes #1-#7, and the learning systems. Each model was trained and tested separately on each mailbox. Error bars are the standard deviation of test set error over ten runs. $0.025 = 2.5\%$.

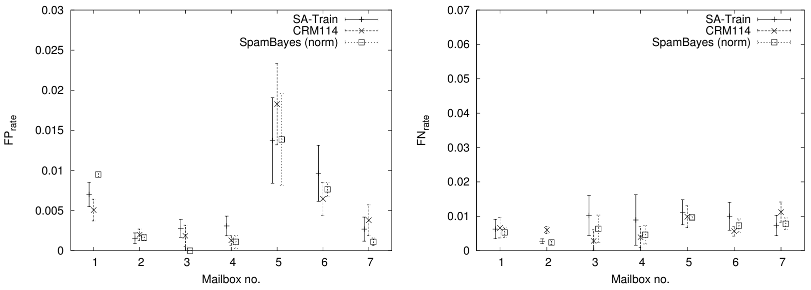

Figure 2: This figure shows FP_{rate} (on the left) and FN_{rate} (on the right) for mailboxes #1-#7, and the learning systems. Each model was trained on pooled data from all mailboxes. Testing was done on remaining mails from each mailbox that were not used for training. Error bars are the standard deviation of test set error over ten runs. $0.025 = 2.5\%$.

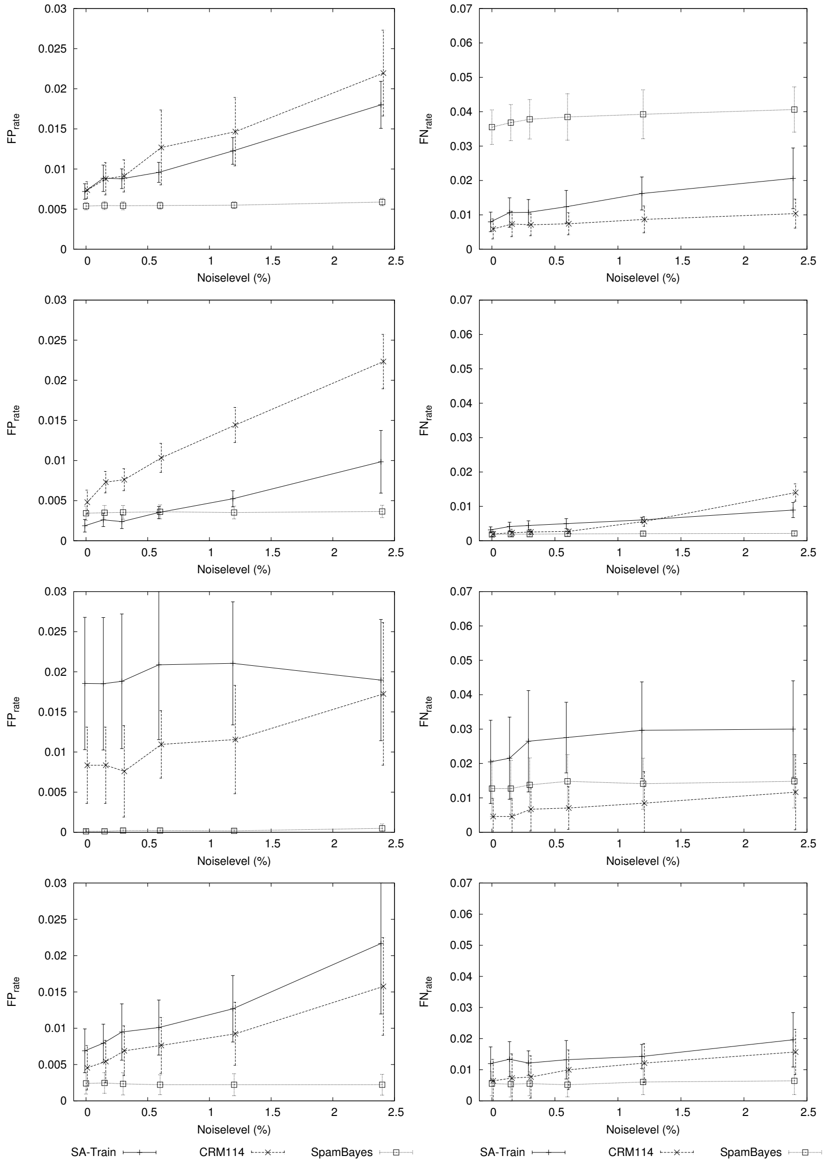

Figure 3: This figure shows FP_{rate} (left) and FN_{rate} (right) at different noiselevels, for mailboxes #1 to #4 (top-to-bottom).

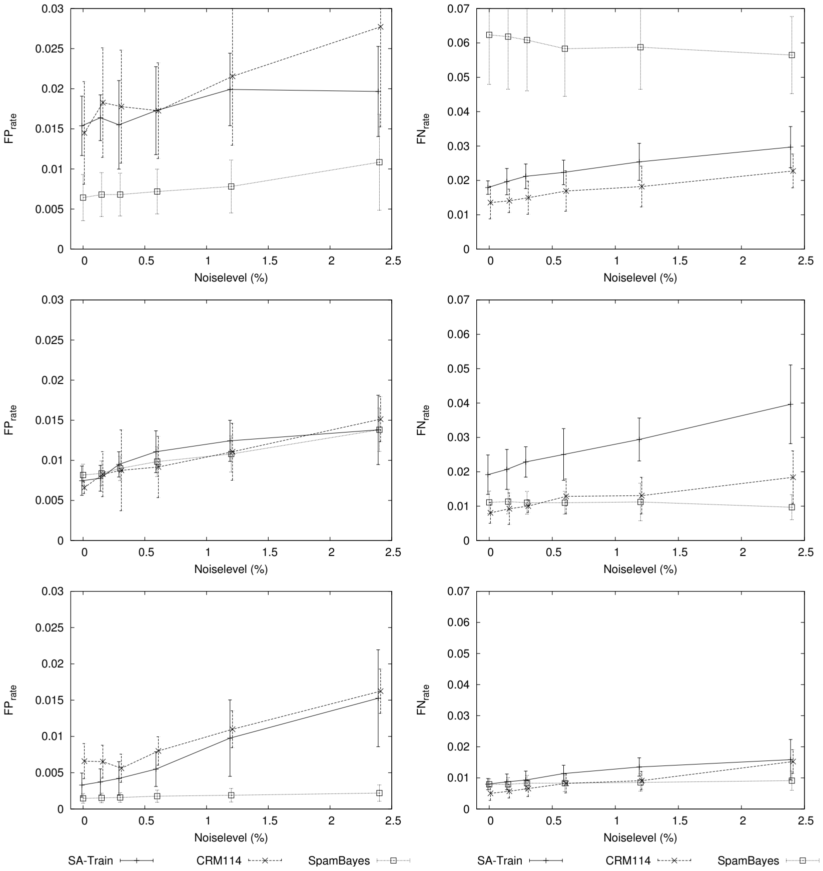

Figure 4: This figure shows FP_{rate} (left) and FN_{rate} (right) at different noiselevels, for mailboxes #5 to #7 (top-to-bottom).

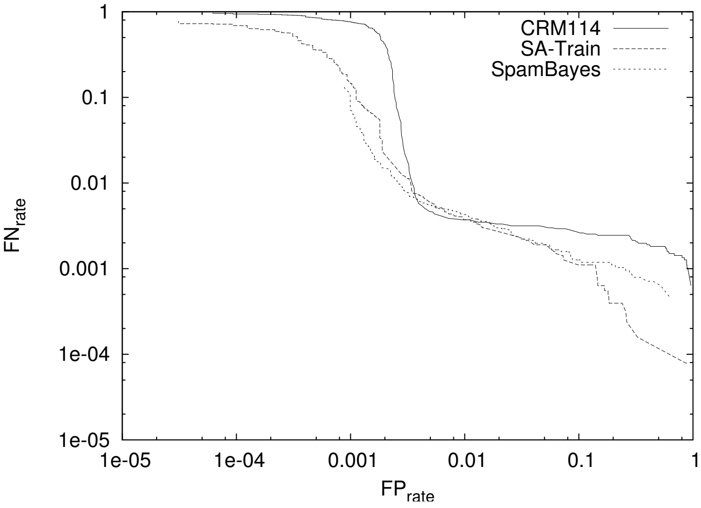

Figure 5: This figure shows a FP/FN rate trade-off curve for the three tested systems. SpamBayes returns many mails with confidence values of 1.0 and 0.0, and is therefore cut off on the left and right.

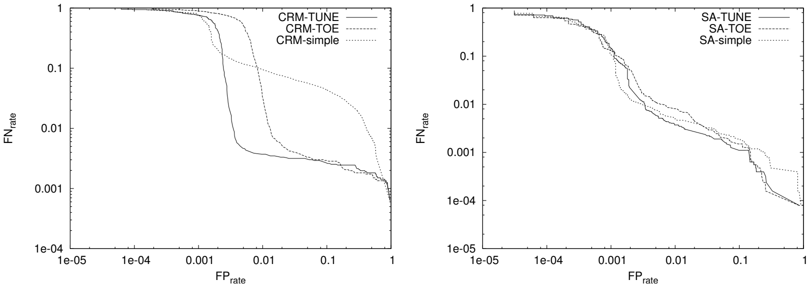

Figure 6: This figure shows a FP/FN rate trade-off curve for CRM114 (left) and SA-Train (right), for the three training methods TUNE, TOE and simple.

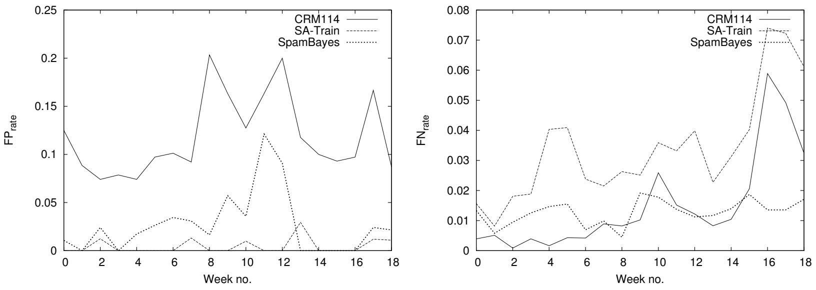

Figure 7: This figure shows FP and FN rates (left: FP_{rate}, right: FN_{rate}) on mailbox #8. Week numbers are measured from the start of #8, only full weeks are shown (i.e. the last partial week was removed)

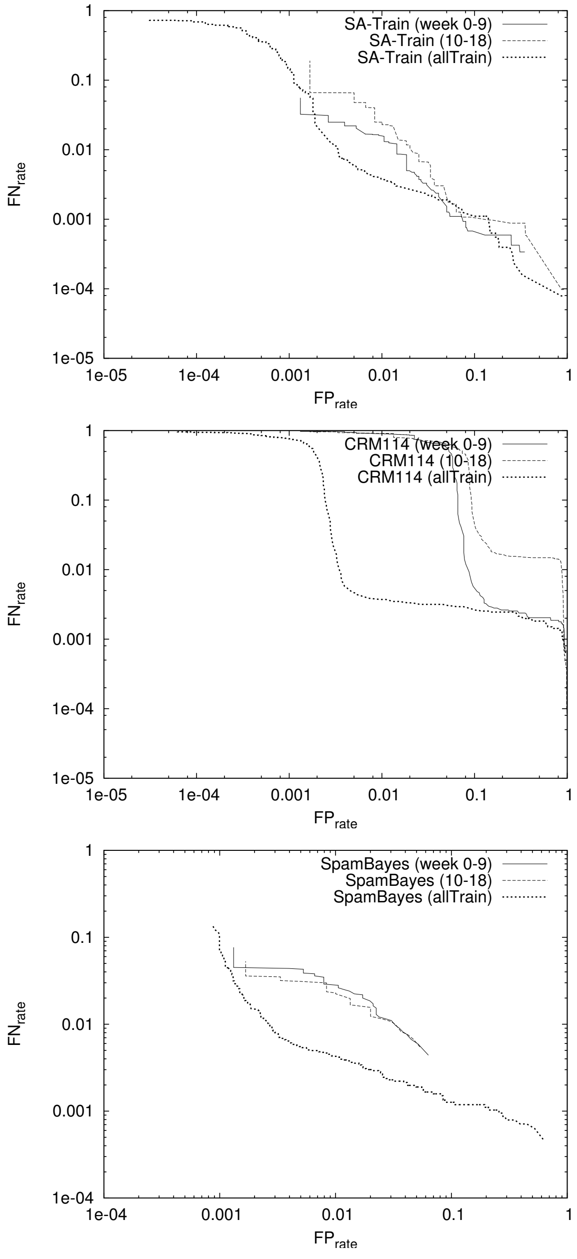

Figure 8: This figure shows FP/FN rate trade-off curves for SA-Train, CRM114 and SpamBayes. The *allTrain* curves are from section 7.4; the two others curves are taken from weeks 0-9 resp. 10-18 of #8.

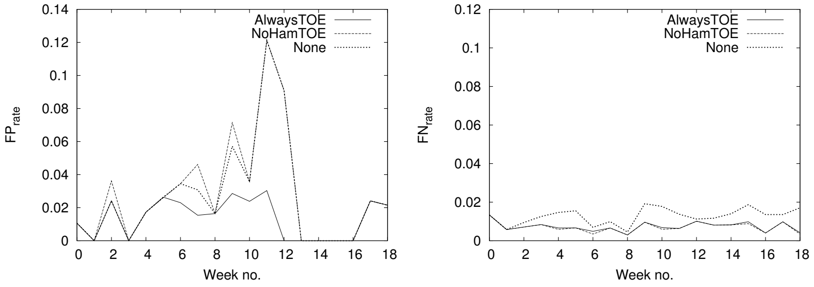

Figure 9: This figure shows FP rate (left) and FN rate (right) for mailbox #8, averaged over each week, for the three continuous training settings.

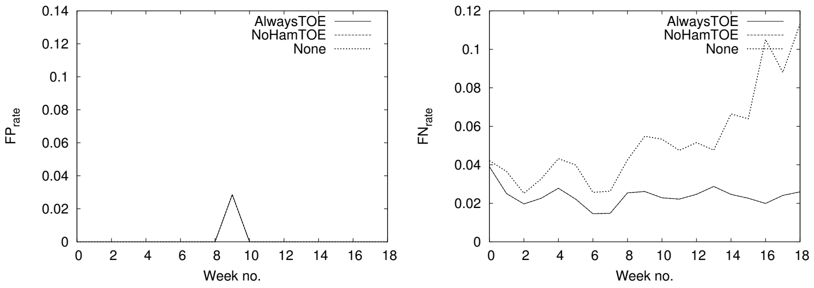

Figure 10: This figure shows FP rate (left) and FN rate (right) for mailbox #8, averaged over each week, for the three continuous training settings, and a threshold of 0.5 for SpamBayes.