Spam Filter Improvement Through Measurement

by

Thomas Richard Lynam

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2009

Abstract

This work supports the thesis that sound quantitative evaluation for spam filters leads to substantial improvement in the classification of email. To this end, new laboratory testing methods and datasets are introduced, and evidence is presented that their adoption at Text REtrieval Conference (TREC)and elsewhere has led to an improvement in state of the art spam filtering. While many of these improvements have been discovered by others, the best-performing method known at this time – spam filter fusion – was demonstrated by the author.

This work describes four principal dimensions of spam filter evaluation methodology and spam filter improvement. An initial study investigates the application of twelve open-source filter configurations in a laboratory environment, using a stream of 50,000 messages captured from a single recipient over eight months. The study measures the impact of user feedback and on-line learning on filter performance using methodology and measures which were released to the research community as the TREC Spam Filter Evaluation Toolkit.

The toolkit was used as the basis of the TREC Spam Track, which the author co-founded with Cormack. The Spam Track, in addition to evaluating a new application (email spam), addressed the issue of testing systems on both private and public data. While streams of private messages are most realistic, they are not easy to come by and cannot be shared with the research community as archival benchmarks. Using the toolkit, participant filters were evaluated on both, and the differences found not to substantially confound evaluation; as a result, public corpora were validated as research tools. Over the course of TREC and similar evaluation efforts, a dozen or more archival benchmarks – some private and some public – have become available.

The toolkit and methodology have spawned improvements in the state of the art every year since its deployment in 2005. In 2005, 2006, and 2007, the spam track yielded new best-performing systems based on sequential compression models, orthogonal sparse bigram features, logistic regression and support vector machines. Using the TREC participant filters, we develop and demonstrate methods for on-line filter fusion that outperform all other reported on-line personal spam filters.

Acknowledgements

I would like to thank my supervisor, Gordon Cormack, for all the guidance he provided and all he has done for me. I especially enjoyed working together on the numerous topics over the years.

Thanks to Kimberly Becken for all the support she provided when I needed it.

Finally, thanks to the members of my committee, Gordon Cormak, Charlie Clarke, Vic DiCiccio, John Aycock, and Olga Vechtomova, who provided valuable feedback and interesting questions.

Dedication

To my wife, my mom, my family and all those who supported and encouraged me along the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Thesis

Sound quantitative evaluation is essential:

- to measure the extent to which email spam filters are effective for their intended purpose;
- to assess the relative effectiveness of current filtering techniques;
- to investigate the worthiness of new techniques.

This thesis describes the development and application of a novel methodology for laboratory spam filter evaluation. Specifically, the evaluation methodology[37] was designed and used to study eight on-line open source spam filters. The methodology was encapsulated and made available to the public as the Spam Filtering Evaluation Toolkit[81]. In conjunction with the toolkit, several private and public corpora were developed to evaluate filters[36]. Both the methodology and toolkit were used in a large scale evaluation of independent filters at the Text Retrieval Conference (TREC) Spam Evaluation Track [30, 33, 34]. Finally, the best known spam filtering method was created by exploiting the evaluation toolkit to combine the results of multiple filters [83].

1.2 Objectives

A spam filter is something that identifies spam email (i.e. unwanted bulk email[1, 127]) so as to prevent its delivery. Filters can mitigate the negative impact of spam, provided that they act in a reliable and predictable manner, eliminating a large portion of unwelcome email, and posing a minimal risk of eliminating welcome email. No filter is perfect; however, neither are the human email recipients who bear the burden of identifying spam, absent a filter. The cost of reading every email is greater than just time [21]. Some users only read the subject or look at the email briefly before deciding it is spam. This may lead the user to mistake a wanted email as spam and ignoring it or worse, deleting it. I, for one, have missed or deleted real emails thinking the messages were spam. This is one of the reasons I became interested in spam filtering. Filters also make mistakes but can be used in conjunction with users to minimize these errors. These considerations all beg the question: **How well do spam filters work?**

To answer this question, one has to answer a variety of questions.

What is spam?

Spam is more than just unwanted email. A spam email is a message that contains a payload the sender wishes to deliver, despite the wishes of the recipient. The payload could be any of a number of schemes from advertising to a bait for fraud. Some of the gains made by the sender of spam are money, control of computers, or extracting personal information. We must have a precise definition of spam in order to find how well filters work.

What is a spam filter?

A spam filter is a computer program that classifies email messages, determining with high probability whether or not a message is spam. Once the filter has identified a message as spam, it may be labeled or flagged, removed from the inbox to a spam folder, or simply deleted. The filter identifies a message as spam through any number of methods which include comparing it statistically to previously received email, examining who sent the message, or matching known keywords and patterns.

How does one quantify spam filter performance?

A perfect spam filter would identify every spam message as spam and every non-spam message as non-spam. A naïve measure of filter performance is accuracy, the fraction of all messages that are correctly identified by the filter. Accuracy gives little indication of the relative risk of identifying spam as non-spam or vice versa. And a vacuous filter that identifies every message as spam, although useless, may achieve a high accuracy score. Such a filter would score 95% accuracy in a typical environment where 95% of messages were spam. Accuracy measures the prevalence of spam as much as it measures filter performance.

A better approach is to measure separately the fraction of spam identified to be non-spam (false negative rate) and the fraction of non-spam identified to be spam (false positive rate). If one filter has both a lower false positive and false negative rate than another, it is clearly superior. But if one rate is higher and the other lower, comparison is more difficult. Summary measures that aptly characterize overall performance are needed.

What is the best method to evaluate realistic spam filter performance?

The ultimate objective is an evaluation methodology that aptly models and measures real filter behaviour. The evaluation should also be repeatable. These constraints are somewhat contradictory due to the real-time nature of the email delivery process. Some filters will use external resources that are nearly impossible to capture. These external resources are changed by their use; they are also updated from other sources, for example, by other filters. For filter development, repeatable evaluation has considerable importance as it is essential to compare proposed techniques and enhancements. To this end, we consider laboratory evaluation methods using archived data, as opposed to in situ measurements.

A laboratory evaluation may restrict the filter's interaction with external entities, it may capture their states, or it may simulate their behaviour. An example of restricting the filter's interaction is the prohibition of challenge-response or greylisting techniques. A primary example of captured state is a complete chronological email stream, along with timing information, and the "true" classification of each message. An example of simulation is the user's behaviour in providing feedback to the filter.

Filter deployment may be simulated in the laboratory using a framework that presents a stream of email messages to the filter, captures the filter's result, and evaluates the captured result. A standard interface allows different filters and message streams to be tested, without requiring any test-specific modifications to either.

Once we can measure filter effectiveness in a repeatable experiment, we may compare results to address the question of which filters work better than others, and, more importantly, what novel approaches improve on established ones.

1.3 Results

A laboratory study of the effectiveness of real spam filters for on-line filtering of real email was completed. The study reports the comparative evaluation in a realistic controlled environment of commonly-deployed spam filters applied to a sequence of all email delivered to an individual within a specific time interval. During the process of completing the study a novel methodology for evaluating spam filters was developed. Measures were developed to reflect a filter's effectiveness for its intended purpose; i.e. abating spam while preserving welcome email messages. In conducting the study it became clear a standard plug-in method of running filters was needed. The Spam Filter Evaluation Toolkit was created to meet this need.

The goal in creating the Spam Filter Evaluation Toolkit is to increase the availability of appropriate evaluation techniques for use by industry and academia, including the deployment of new evaluation techniques that simulate different modes of deployment. The toolkit has provided a standard evaluation of current and proposed spam filtering approaches. It also provides an architecture, common tools and methodology for an open-ended network of evaluation corpora (public and private) by establishing a standard interface for evaluating spam filter performance. The toolkit has also been widely used by researchers creating and studying spam filters. Notably, this toolkit is the foundation of evaluation for the TREC spam track.[30]

The Text Retrieval Conference, TREC, is an annual conference hosted by the U.S. National Institute of Standards and Technology (NIST) whose goal is

> "to increase the availability of appropriate evaluation techniques for use by industry and academia, including the deployment of new evaluation techniques more applicable to current systems."[1]

With Cormack, I co-founded the TREC Spam Track to explore, develop and evaluate spam filtering techniques at the conference. Groups from around the world have participated in the TREC spam track. These groups submit their filter to TREC for evaluation; the results are published at the conference where the groups describe their spam filtering techniques. The TREC spam filtering evaluations were completed using the Spam Filtering Evaluation Toolkit. The spam track ran in TREC 2005, TREC 2006 and TREC 2007. Each year, significant improvements in spam filtering techniques have been achieved.

[1]trec.nist.gov

A novel aspect of the TREC Spam Track, facilitated by the toolkit, was the submission of participant filters for evaluation on private data, in addition to the more traditional evaluation on public data. Realistic email data are very difficult to acquire, and in general cannot be distributed. The Spam Track answered affirmatively the outstanding question of whether measurements on publishable data agree with those on more realistic private data.

The standard interface enforced by the toolkit makes available a large number of plug-in filters that, in addition to being compared, can be combined into an ensemble. Using the filters submitted to TREC, an on-line fusion technique was developed that consistently improves on the performance of the best individual filter. To this date, the developed fusion technique yields the best published results on all datasets amenable for use with the toolkit.

Chapter 2

Motivation and Background

In 2004, my personal motivation was to find a solution to the spam that was filling my mailbox. My task was to find the most effective way to eliminate spam while minimizing the risk of eliminating real mail. The easiest option would have been to install one of the many available open source spam filters; however, I was interested in learning more about how filters work and believed a better filter was still possible. To achieve this, I decided to create my own filter.

Having just started my Doctorate of Philosophy when spam filtering became a topic of much interest, I decided to make it the focus of research for my thesis. I tackled this interesting problem by breaking it into three steps:

1. Learn how spam is filtered.
2. Determine the best spam methods.
3. Improve on the best methods.

This chapter will cover an overview for each step of my doctorate plan and why the last two steps became much more complicated when explored in detail.

2.1 Understanding Spam Filtering

To learn how spam is filtered, it must be defined before describing how the process is completed. Although methods vary greatly, spam filtering has a well-defined purpose.

What is Spam?

Even defining what appears to be a simple term quickly became complex. Early definitions like "unsolicited commercial email" [63] do not appear to capture the public consensus on the nature of spam. A great deal of effort may be expended in determining whether or not email is commercial, when for practical purposes this aspect has little impact on the user's perception. The looser definition "unwanted email" fails to capture the indiscriminate nature of spam, and is nearly impossible to adjudicate in an objective and repeatable manner.

I propose a formal definition in chapter 3, which was used for TREC and is used throughout the thesis. Even so, there may be uncertainty about the true class of any particular message, either because of disagreement in interpreting the definition or because a user is simply unsure. Some evaluations exclude such messages; such exclusion tends to inflate performance measurements while introducing a new uncertainty: which messages should be included and excluded from the evaluation? The philosophy employed here is that the definition should be as precise as possible, and the truth should be determined as accurately as possible relative to the definition. The effect of different definitions is itself amenable to measurement.

How is Spam Filtered?

Figure 2.1 shows the delivery of an email stream including filtering. It begins with the arrival of an email message addressed to a particular recipient. The message is given to the spam filter, which classifies the mail as spam or ham (non-spam) using a variety of methods. If the filter classification is ham, the message is delivered to a ham file (inbox) which the user reads systematically. The user may identify incorrectly classified spam delivered to the inbox and correct the spam filter via the mail reading interface. The user does not systematically read the email classified as spam but may search or browse it occasionally to find and retrieve any mislabeled email.

Spam filtering differs from classical text categorization [111] in that spam evolves. There is a never-ending battle between those who create spam and those who create spam filters. For this reason, filters must be able to be updated frequently. Filter update strategies are as varied as filter methods. Updates fall into two types: manual and learned. A manual update involves an actual person changing the filter. This could be as simple as adding the rule that all email with the term Viagra are spam. With learned update, the filter will find content of the email that indicate spam or non-spam by using known

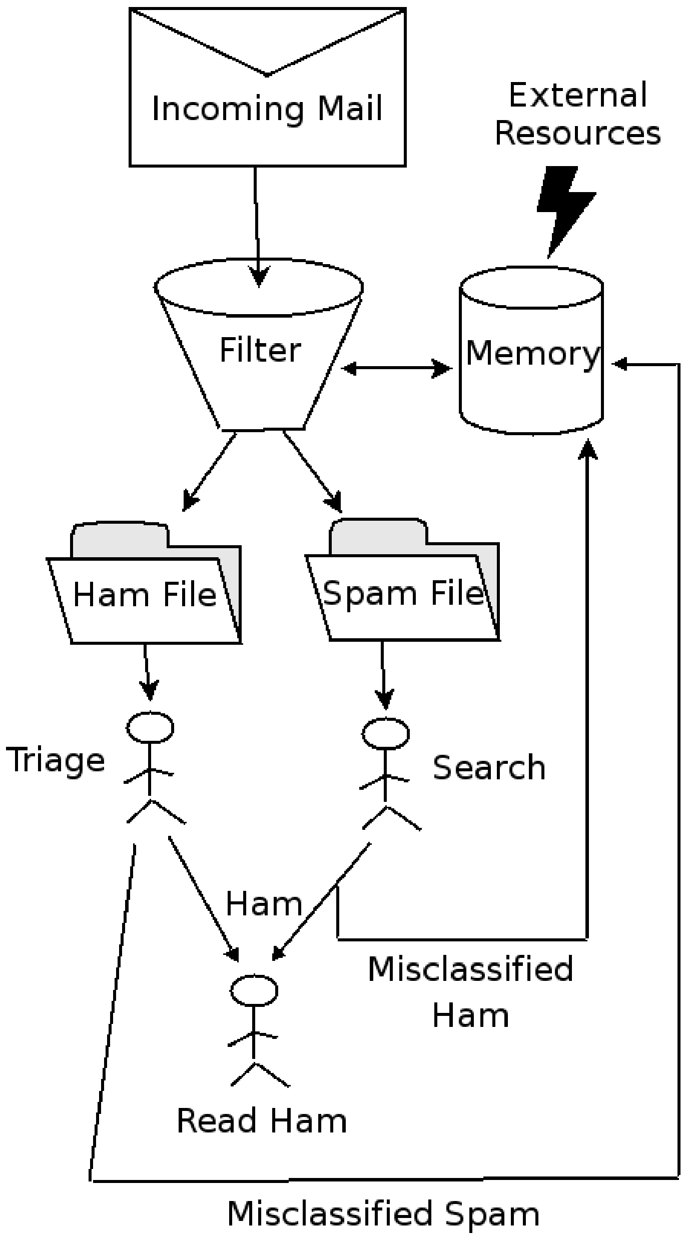


Figure 2.1: Spam Filter Usage

classified email. The time frame that filters update may only occur every few months or can happen very frequently (after every new email) or everything in between.

The process shown in figure 2.1 demonstrates an on-line update process. One message is processed and classified at a time. Before the next message is processed, the filter may adapt based on the information it receives, which is often referred to as training. The information could include user feedback, features in the email and such. External sources of information such as DNSBL[1], that many filters use, can be added to the process.

When the filter trains on more than one email at a time, the process is referred to as batch training. A very common batch filtering approach is to update the filter on a set of training email and then use the updated filter on real incoming email. This filter may be updated again after some number of email. Some learning filters take vast resources to complete training; these filters are usually updated using the batch process.

Based on this model presented in figure 2.1, the filter can learn what is spam and ham email for a particular user by using the user's feedback to update the filter's memory.

A perfect spam filter would avoid ham misclassification – incorrectly placing a ham message in the spam file – and spam misclassification – incorrectly placing a spam message in the ham file. Ham misclassification poses a serious risk; should the recipient fail to retrieve the misclassified message from the spam file, it would be lost. Spam misclassification, on the other hand, exposes the recipient to a degree of the original inconvenience, annoyance and risk associated with spam. An effective filter should mitigate the effects of spam while maintaining an acceptable risk of loss.

Typically a filter will estimate the likelihood that the classified message is spam. If the estimate (a spamminess score) is over some threshold the filter will classify the message as spam. Because the relative risks and costs associated with ham and spam misclassification may vary from one situation to another, most spam filters, in addition to classifying each email message, have a threshold parameter which may be adjusted to decrease ham misclassification at the expense of spam misclassification, or vice versa.

With the goal of removing spam from my inbox, I decided to focus this research toward on-line personal learning filters.

[1]DNS Blacklist is a list of DNS addresses linked to spamming

2.2 Spam Filtering circa 2004

In 2004 when this research was initiated, there was considerable diversity – and considerable controversy – in the methods applied to spam filtering and the means by which these methods were evaluated. It was not at all clear which methods were best, and which held the most promise for improvement.

These issues were addressed by three distinct communities: (1) the community of developers and practitioners whose motivation was to build tools for immediate deployment; (2) the community of spam filter vendors whose motivation was to sell spam filters; (3) the research community whose motivation was to discover new truth or to validate existing theories and algorithms.

The methods employed and investigated by users, practitioners, vendors and researchers, may be broadly categorized as:

- manual inspection;
- systems-oriented approaches;
- content based filters.

Content-based filters may be further categorized as:

- ad hoc rule-based filters;
- practical learning filters, dubbed “Bayesian”, derived from the work of Graham[55, 56] and Robinson[102, 101, 100];
- laboratory-oriented methods of machine-learning research.

Manual inspection

The obvious alternative to automatic spam filtering is to deliver all email to the end user, who examines each message and decides whether or not to treat it as spam. The user acting in this capacity may be regarded as a human spam filter, with associated cost and risks not unlike those for automatic filters. While the cost in terms of time is obvious, it is not necessarily easy to quantify[62]. Furthermore, the risk of error is not zero, as is sometimes assumed in comparing the risks and benefits of automatic filtering.

Users may believe that they can do a better job than filters. In one study Yerazunis[132] diligently adjudicated a collection of email on two different occasions. Yerazunis diligently examined the full email (headers and text) and found a disagreement rate of 0.16%. It is likely other diligent users should expect an error rate similar to 0.16%. In reality average users' error rates are much higher. Anecdotally, the author is aware of several instances of users overlooking important email that happened to look like spam.

Filtering by hand requires time and, as the percent of spam increases, so does the time requirement. When spam is infrequent, human filtering is a practical solution as extra work is minimal and mistakes extremely rare. As the spam increases, so does the workload, as well as mistakes.

Human filters have an advantage of full knowledge of what the user thinks is spam or not spam. The disadvantage is that humans make mistakes which could mean deleting the wrong email. Users also do not have easy access to external resources that might indicate an email is spam such as DNS Blacklists.[2]

"Systems" approaches

Systems approaches rely on information extrinsic to the message and the user in identifying spam. They are typically applied before the email is delivered to the user. Common methods employ lists of good senders (white lists), lists of bad senders (black lists), or lists of particular spam messages (fingerprint lists). These lists may be created and maintained by system administrators, or they may be maintained collaboratively, with end-users contributing list elements as they are discovered.

In contrast to the methods above, greylisting and challenge-response elicit some particular behaviour from the sender and assume that the message is spam if this behaviour is not observed. Each introduces delay, additional network traffic, and risk of message loss.

System approaches are employed by both practitioners and vendors. Due to the dynamic real-time nature of many system approaches, their effectiveness is difficult to measure, and few results appear in the literature.

A white list is a list of senders (user, domains, or IP addresses) that are safe addresses. The safe addresses are acquaintances or not known for sending spam. White lists are used to override a spam filter so that all messages from senders in the white list are classified

[2]DNS Blacklist is a list of DNS addresses linked to spamming

as ham. Though white lists can reduce the chance of a real message being classified as spam, the downfall is that all email from the senders in the list is classified as ham. If a spammer can spoof one of the addresses in the white list, spam will be misclassified as ham. It is very common for spammers to exploit this weakness to get spam to the users. It is actually relatively easy to find addresses in the white list. One simple example is email addresses being on the same website. If users addresses are on a website because they know each other or work together, there is a strong likelihood they are in each other's white lists. It is too easy for spammers to spoof the sender ID that white lists use as an indication of ham.[80]

A black list[29, 89] differs from a white list in that senders in white lists are classified as ham, where senders in a black list are classified as spam. One of the problems with black lists is that spam comes from so many different sources that a single user's black list is too incomplete to be effective. In order to be more effective, much larger black lists are compiled using many users. Two examples of this are Real-time Black lists (RBL) and DNS Black lists (DNSBL). Spam filters can both query and add to the lists. Blacklists can cause false positives because a genuine email sender may have been spoofed by a spammer. The real-time update of black lists makes repeatable evaluation very difficult. The black list's state would need to be captured at each point in time in the testing corpus.

Another system spam filter approach is greylisting[79, 60]; a method for filtering spam by temporarily rejecting all email from unknown senders by the mail server as it is received. A properly configured mail server will resend the message and the email will be delivered to the user. The theory is that the spammer will not resend the messages. The advantage is that because it is not well used, spammers do not usually resend messages; therefore, it reduces spam. It also requires no additional architecture at the user's end. The impact on the user is a delay in the first email from a sender, as well as possible lost messages from misconfigured mail servers. Spammers can easily outflank greylisting simply by resending the messages. Another drawback is that some sender mail servers will warn the senders about the temporary rejection leading to confusion. One common delay is when a user creates an account on a new website, the confirmation email will be delayed by greylisting. This can be frustrating to the user who is accustom to the instantaneous nature of email.

Collaborative filtering[115, 75, 74, 44, 42] is a system approach that exploits the fact that the same spam email is sent to many different users. A collaborative spam filter must capture spam and be able to recognize duplicates over a large number of systems.

Finding spam is as easy as setting up an email address. If an email address is never used for legitimate email, all email received will be spam. Due to privacy, messages need to be encrypted; this would limit the ability to find near duplicates. Any user-specific header information could not be used. Also, due to the volume of spam it would be impossible to store complete messages. Collaborative spam filtering could only be as effective as its completeness. The greater the completeness the more complicated and delayed judgements would be.

A more intrusive suggested system approach is challenge response[61, 24, 25, 118, 97, 22] filtering which adds a cost to sending an email. The cost should be easy for a legitimate sender but time consuming or costly for a spammer to send millions of spam. Examples of challenge response are asking the user to resend, clicking a link, and making a payment. An email is usually sent to the user with instructions how to respond. This causes legitimate mail to be lost by autonomous sender's web transaction confirmations and such, as there is no one to respond. This make the use of challenge response unfavorable.[58]

Ad hoc Content-Based Methods

Almost all email systems provide a mechanism to filter messages based on keywords or simple patterns. Early spam filters used these mechanisms to identify words like "Viagra" likely to occur in spam. SpamAssassin[121] generalizes this approach, using a wide variety of rules composed by authors and users to identify spam.

In 2004 SpamAssassin was the most popular open source filter that exploited rules as one of its methods. By default SpamAssassin had approximately 800 rules, each having a score. New rules and scores may be added or old ones may be customized to improve performance for a specific user. SpamAssassin also embedded white list, black list, fingerprint as "rules". SpamAssassin's popularity was also a hindrance as spammers would test spam against SpamAssassin. An example of a SpamAssassin rule named OFFER is

```
body OFFER /free offer/i
score OFFER 1.0
```

If the body contains the text "free offer" the messages score is increased by 1.0. SpamAssassin default threshold is 5.0; anything over that score is classified as spam. The default threshold can be easily changed.

"Bayesian" Filters

Graham's influential essay, "A Plan for Spam,"[55] reported very good results using a simple content-based classifier. The technique was improved by Robinson[100] and became the de-facto standard for practical content-based spam filtering. While it is not a pure naïve Bayes classifier, and a number of derivative methods bear even less similarity to naïve Bayes; the entire class of methods has come to be known as "Bayesian Spam Filtering." A large number of stand-alone filters employing the technique, including SpamBayes, Bogofilter and SpamProbe remain in common use. Other systems, notably SpamAssassin and Mozilla Mail, adopted the method.

Graham[57, 56] and Robinson's[102, 101, 100] method decomposes each message into individual tokens or words, and computes the fraction of spam and the fraction of ham messages containing each word by examining a number of messages, each known to be spam or ham. In classifying a new message, words in the message that occur more frequently in spam are taken as evidence that the message is spam, and vice versa. Once the message has been read by the user and definitively labeled as spam or ham, it may be used to update the computed fractions, in accordance with on-line deployment (figure 2.1).

As of 2004 Bayesian spam filters were heavily utilized by developers and practitioners and explored by researchers. Due to the user interaction worries, it was not a method used by vendors at the time.

Learning research methods

The research community has investigated a multitude of classification methods. Well-researched methods for text classification include naïve Bayes classifiers, perceptron[51, 50], winnow[117], support vector machines[124][124], clustering methods such as nearest neighbor[40, 41], and decision trees[23].

The perceptron algorithm[51, 50] is a linear classifier that separates a data set of items (emails for spam) into two groups by iteratively attempting to correct all errors by incrementing or decrementing weights for incorrectly classified items. The algorithm ignores correctly classified items. If the items are linearly separable, the perceptron algorithm converges in a finite number of steps. For non-linear separable items the algorithm must be stopped after some number of iterations. The perceptron is simple and incremental, making it useful for spam filtering. The winnow[117] algorithm is closely related to the perceptron, but computes weights using a multiplicative scheme instead of the perceptron's additive scheme. Similar to the perceptron algorithm, winnow trains on error but can be adapted to train on close calls.

Of all the researched text classification methods, support vector machine (SVM) have been touted the most effective[124][124]. SVMs are linear classifiers which have been shown to yield state-of-the-art performance on text classification by finding a hyperplane that separates two classes of data while maximizing the margin between them. To calculate the margin, two support vectors are constructed, one on each side of the separating hyperplane. SVMs require training time that is quadratic in the number of training examples, therefore, impractical for large scale email systems. As of 2004, some work was invested in applying SVM to spam; however, there were no known publicly available filters using SVM.

Nearest neighbor[40, 41] classifier computes the distance between an item and previous known items based on the features of the items. The item is classified the same as its closest neighbor. An extension is k-nearest neighbor classifier (kNN) for some fixed k, the k closest neighbors are found and the majority classification is assigned to the item.

Another classifier that can be used for spam is decision trees[23]. A decision tree maps observations about an item to conclusions about the item's classification. To train, decision trees must successively split train data by using one observation at a time. The trick is to choose the order in which to examine the observation.

2.3 Spam Filter Evaluation circa 2004

The communities have largely investigated different methods using different approaches, and reported results in different venues. Even within each community, there is little consensus as to which methods works best, and incomparable claims abound (see table 2.2). Practitioners and vendors claimed amazing spam filter performance. Researchers report significantly lower spam filter performance results than the other two groups.

The next section outlines the diverse claims and reporting methods existing at the outset of this research. The evaluation measures are detailed and contrasted in the following section.

2.3.1 Incomparable Claims

Paul Graham[56] contrasts his practical results with those of the research community:

> Spam filtering is a subset of text classification, which is a well established field, but the first papers about Bayesian spam filtering per se seem to have been two given at the same conference in 1998, one by Pantel and Lin [93], and another by a group from Microsoft Research [104].[3]
>
> When I heard about this work I was a bit surprised. If people had been onto Bayesian filtering four years ago, why wasn't everyone using it? When I read the papers I found out why. Pantel and Lin's filter was the more effective of the two, but it only caught 92% of spam, with 1.16% false positives.
>
> When I tried writing a Bayesian spam filter, it caught 99.5% of spam with less than .03% false positives.[4] It's always alarming when two people trying the same experiment get widely divergent results. It's especially alarming here because those two sets of numbers might yield opposite conclusions. Different users have different requirements, but I think for many people a filtering rate of 92% with 1.16% false positives means that filtering is not an acceptable solution, whereas 99.5% with less than .03% false positives means that it is.

Jonathan Zdziarski [136], developer of DSPAM, makes the following claim:

> After three months of development, the first public beta of DSPAM v3.2 has been released for testing. ... Accuracy in 3.x has reportedly peaked as high as **99.991%** (2 errors in 22,786 messages).

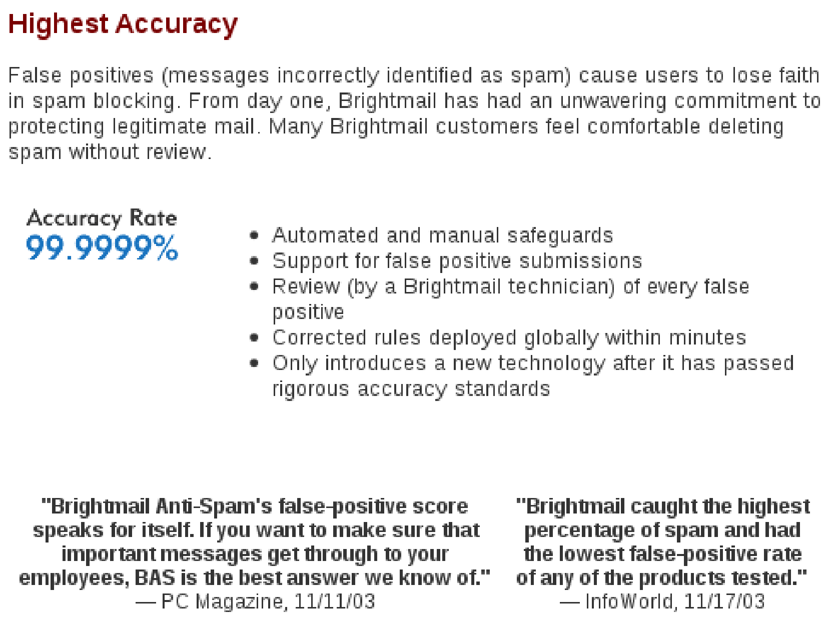
Bill Yerazunis[132], developer of CRM114 also claims astounding performance:

> CRM114 - the Controllable Regex Mutilator - As of Feb 1 through Feb 21, 2004, 6000+ messages, my total error rate was ONE. That translates to better than 99.984% accuracy, which is ten times more accurate than human accuracy (which I've measured to be around 99.84%.)

[3]Sahami et al.[104] conducted a study 1998 that indicated the utility of Bayesian classifiers for spam filtering. The best-performing system achieved ham recall of 100% and spam recall of 98.3%.

[4]At the time I had zero false positives out of about 4,000 legitimate emails. If the next legitimate email was a false positive, this would give us .03%. These false positive rates are untrustworthy, as I explain later. I quote a number here only to emphasize that whatever the false positive rate is, it is less than 1.16%.

Vendors claim even better performance than practitioners. The following is advertising from brightmail[19] in 2004.



Highest Accuracy

False positives (messages incorrectly identified as spam) cause users to lose faith in spam blocking. From day one, Brightmail has had an unwavering commitment to protecting legitimate mail. Many Brightmail customers feel comfortable deleting spam without review.

Accuracy Rate
99.9999%

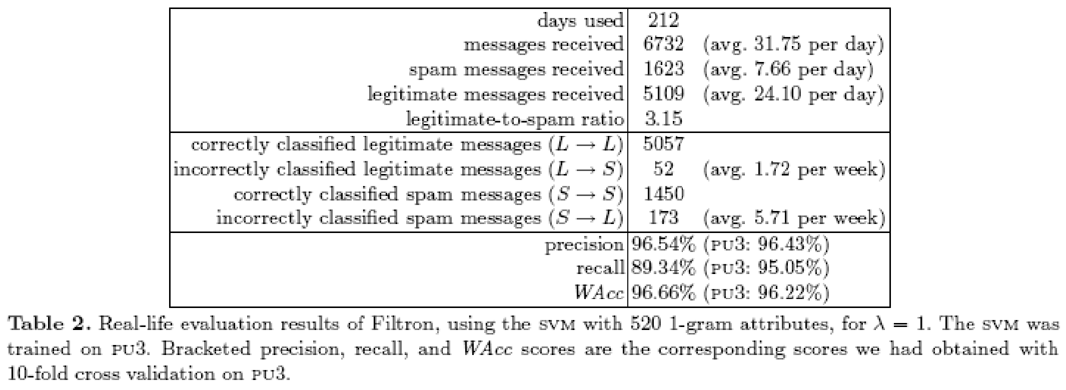
- Automated and manual safeguards
- Support for false positive submissions
- Review (by a Brightmail technician) of every false positive
- Corrected rules deployed globally within minutes
- Only introduces a new technology after it has passed rigorous accuracy standards

"Brightmail Anti-Spam's false-positive score speaks for itself. If you want to make sure that important messages get through to your employees, BAS is the best answer we know of."
— PC Magazine, 11/11/03

"Brightmail caught the highest percentage of spam and had the lowest false-positive rate of any of the products tested."
— InfoWorld, 11/17/03

Researchers have, in contrast, reported surprisingly poor performance. These reports use a large variety of measures including several that convey little insight into filter behaviour.

Michelakis et al create an SVM based spam filter. Their real-life evaluation results are detailed in the table 2.1.



days used	212	
messages received	6732	(avg. 31.75 per day)
spam messages received	1623	(avg. 7.66 per day)
legitimate messages received	5109	(avg. 24.10 per day)
legitimate-to-spam ratio	3.15	
correctly classified legitimate messages ($L \to L$)	5057	
incorrectly classified legitimate messages ($L \to S$)	52	(avg. 1.72 per week)
correctly classified spam messages ($S \to S$)	1450	
incorrectly classified spam messages ($S \to L$)	173	(avg. 5.71 per week)
precision	96.54%	(PU3: 96.43%)
recall	89.34%	(PU3: 95.05%)
WAcc	96.66%	(PU3: 96.22%)

Table 2. Real-life evaluation results of Filtron, using the SVM with 520 1-gram attributes, for $\lambda = 1$. The SVM was trained on PU3. Bracketed precision, recall, and *WAcc* scores are the corresponding scores we had obtained with 10-fold cross validation on PU3.

Table 2.1: Michelakis et al 2004 Results

An early study on spam filtering was completed by Pantel and Lin[93] in 1998. They found that a Bayesian filter did an adequate job of classifying email.

> Our experiments show that SpamCop is able to identify about 92% of spam while misclassifying only about 1.16% of the nonspam e-mails

Sahami[104] et al completed a study in 1998 investigating Bayesian filtering and found the following:

> We find that the filter is in fact quite successful at eliminating 80% of incoming junk E-mail from the user's mail stream. For completeness, we also provide the Precision/Recall curve for this task in [figure 2.2]. Based on these results we believe that such as system would be practical for usage in commercial E-mail application.

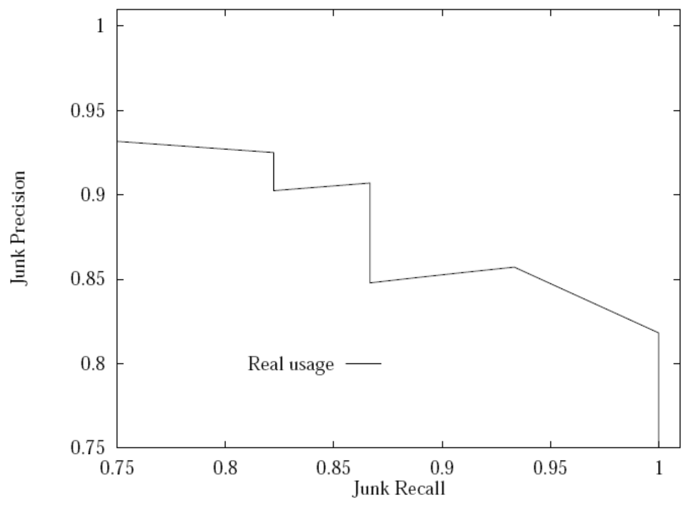


Figure 2.2: Sahami Precision/Recall curve

Kolcz and Alspector[73] completed an evaluation using SVM for spam filtering. They note that ham misclassification can be traded for spam misclassification by changing the filter threshold for identifying spam. They included receiver operating characteristic (ROC) analysis in their paper[see section 2.3.3]. Their ROC curve for the evaluated SVM filter is shown in figure 2.3.

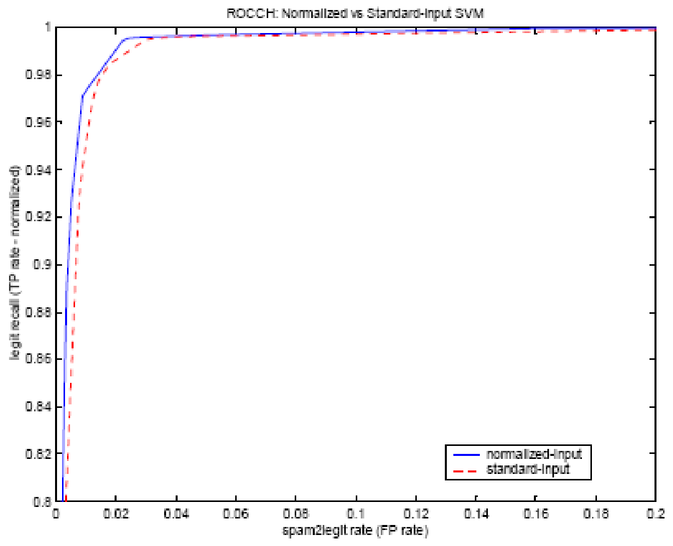


Figure 2.3: Kolcz Alspector ROC curve

Table 2.2 summarizes the major published results as of 2004. Quick analysis of the table shows more work is needed to determine what the top performing filter is. There are three problems with making comparisons. One, several different measures are used in determining filter performance. These measures are described in section 2.3.2. The second problem in comparing filters is the evaluation architecture is different for each published performance. The different test apparatus will be discussed in section 2.3.4. The final problem is each filter is evaluated against a different set of emails with the exception of the few filters tested by Holden. Test corpora are described in section 2.3.5. The Holden study is also the only one that is an independent evaluation; that is where the evaluation was performed independently from the developer of the method.

Filter (method)	Performance	Test Apparatus	Corpus test size
Sahami[104](Bayesian)	ham recall 100%, spam recall 98.3%	lab-batch	Microsoft(features) 267 messages
SpamCop[93](Bayesian)	92% spam identified,1.16% misclassified ham	lab-batch	SpamCop(raw body) 277 spam, 346 ham
Drucker[45](SVM)	0.0213 error rate	lab-batch	AT&T (features) 60 messages
Sakkis et al[105](kNN+Bayesian)	8.60 Total cost ratio$\lambda = 1$	lab-10fold	LingSpam(abstracted) ~241 message
Kolcz-Alspector[73]	1.78$[0.72, 2.27]$ Total cost ratio std	lab-batch	Kolcz-Alspector(features) 1341 spam,1508 ham
Graham[56](Bayesian)	99.5% spam identified .03% false positives	anecdotal	Graham(raw) unknown
SpamAssassin[65](Rules+Bayesian)	99.8% precision, 78.7% recall	lab-batch	Holden(raw) 573 spam, 100 ham
Dspam[136](Bayesian)	99.991% accuracy	anecdotal	Zdziarski(raw) unknown
crm114[132, 134](Markovian)	99.984% accuracy	lab-on-line	Yerazunis(raw) 1518 spam, 856 ham
Cloudmark[27](vendor)	99% accuracy	unknown	cloudmark(raw) unknown
Brightmail[19](vendor)	95% spam-catching rate,99.9999% accuracy	unknown	brightmail(raw) unknown
Bogofilter[65, 99](Bayesian)	100% precision, 81.7% recall	lab-batch	Holden(raw) 573 spam, 100 ham
SpamProbe[65, 20](Bayesian)	99.8% precision, 97.2% recall	lab-batch	Holden(raw) 573 spam, 100 ham
dbacl[65, 18](Bayesian)	99.2% precision, 89.0% recall	lab-batch	Holden(raw) 573 spam, 100 ham
SPASTIC[65, 123](Rules)	88.5% precision, 43.3% recall	lab-batch	Holden(raw) 573 spam, 100 ham
Filtron[88](SVM)	96.5% precision, 89.3% recall,96.7% WAcc	lab-10_fold	Filtron(features) ~596 messages

Table 2.2: Filter Performance 2004

2.3.2 Incomparable Measures

As table 2.2 illustrates, there are many different evaluation measures. These are outlined in this section.

Contingency Table

To measure performance, each message the filter is to be evaluated on must be correctly judged. These correct judgements are known as the gold standard. Given a set of filter classified messages and the associated gold standard, a contingency table can be constructed.

		Gold Standard	
		ham	spam
Filter	ham	a	b
	spam	c	d

Table 2.3: Contingency Table

There are four possible outcomes for given messages; a contingency table (table: 2.3) displays the number of times each outcome occurs for a set of emails.

- a is the number of ham messages correctly classified by the filter as ham
- b is the number of spam messages incorrectly classified by the filter as ham
- c is the number of ham messages incorrectly classified by the filter as spam
- d is the number of spam messages correctly classified by the filter as spam.

From the contingency table, a number of summary statistics may be deduced:

- Prevalence, $prev = \frac{b+d}{a+b+c+d}$, is the fraction of all email that is spam.
- Ham misclassification fraction, $hm = \frac{c}{a+c}$, is the fraction of misclassified ham. hm is identical to *false positive rate (fpr)* from signal detection theory; however, the latter term is often misused in spam filter evaluation, so we chose a more application-specific one.
- Spam misclassification fraction, $sm = \frac{b}{b+d}$, is the fraction of misclassified spam. In some literature this is referred to as false negative rate (fnr).

- Diluted false positive rate, $dfpr = \frac{c}{a+b+c+d}$, is the fraction of all messages that the filter incorrectly classifies as spam. Many practitioners and vendors incorrectly report $dfpr$ as fpr.
- Accuracy, $acc = \frac{a+d}{a+b+c+d}$, is the overall fraction of messages that are correctly classified by the filter.
- Overall misclassification fraction $m = \frac{b+c}{a+b+c+d} = 1 - acc$ is the fraction of all messages that the filter incorrectly classifies. Also known as *error.*
- Spam precision $sp = \frac{d}{c+d}$, often reported simply as *precision*, is the fraction of messages classified by the filter as spam that actually are spam. The measure derives from information retrieval, in which retrieved documents are either relevant or not. Spam precision assumes that spam messages are relevant. Spam precision is equivalent to positive predictive value in signal detection theory.
- Spam recall $sr = \frac{d}{b+d}$ is the fraction of all spam messages that are classified by the filter to be spam. Identical to $1 - sm$.
- Ham precision $hp = \frac{a}{a+b}$ and ham recall $hr = \frac{a}{a+c}$ are precision and recall, assuming ham messages to be relevant. From the perspective of evaluating the effectiveness of a spam filter for its intended purpose, hp and hr are more apt than sp and sr, but seldom reported.

2.3.3 Ham/Spam Misclassification Trade off

The consequences of ham and spam misclassification depend on the nature of the messages and the user's needs. The separate measures hm and sm characterize filter performance independent of these considerations, and may be combine with deployment-specific parameters to quantify the effectiveness of a filter in any particular situation. In particular, $prev$ varies from one deployment situation to another, as does the user's relative sensitivity to ham and spam misclassification.

The separate measures, however, make it difficult to compare filter effectiveness. One approach to comparing filters is to conflate the parameters of one specific deployment with hm and sm to yield a single measure. This approach is limited to the extent that the parameters actually capture the deployment-specific consequences, and also to the extent that the deployment is typical. acc is the simplest such measure, assuming that the misclassification of a ham or spam message is of equal consequence. Furthermore, it is impossible to deduce *hm* or *sm* from *acc*, or to predict *acc* in a deployment situation in which *prev* is different from that of the test environment, even if known.

Androutsopolous et al.[7] argue that the relative importance of ham over spam misclassification errors be quantified by a parameter λ used as input to the filter in *cost-sensitive classification* and to the evaluation measure in *cost-sensitive evaluation.* They define *cost-sensitive* measures:

- Weighted accuracy is defined as $Wacc{=}\frac{\lambda a+d}{\lambda a+b+\lambda c+d}$,
- *Total cost ratio* is defined as $TCR = \frac{b+d}{b+\lambda c}$

where a, b, c, d are taken from the contingency table.

Weighted accuracy alters the deployment-specific consequence of misclassification, but otherwise has the same shortcomings as *acc*. In particular, it is sensitive to *prev*, and a vacuous filter may achieve a high score. Hidalgo[62] discusses the problem of choosing λ:

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

Furthermore, for high values of λ, $Wacc$ is statistically unstable because the score depends almost entirely on the incorrect classification of only a handful of messages.

Androutsopolous et al. propose TCR that evaluates filters relative to the vacuous filter. A filter improves on the vacuous filter if $Wacc > 1$; however, $Wacc$ has little meaning in terms of the filter's intended purpose. TCR does not address the other shortcomings of $Wacc$.

This criticism – dependence on highly variable external factors, arbitrary filter parameters, and arbitrary evaluation weights – applies to a large class of combined evaluation measures[111].

Precision and recall similarly conflate *prev* and deployment-specific consequences with the evaluation measure. It is not possible to deduce *hm* and *sm* from precision and recall alone; however, if *prev* is also known, it is. Conversely, given *hm* and *sm* and *prev* for any deployment, precision and recall may be computed.

The problem of deployment-specific consequences may be addressed by exposing a threshold parameter to the user or system administrator. Most filters operate by computing a score s and by comparing that score to a threshold t, reporting spam if $s > t$ and ham if $s \leq t$. Increasing t reduces hm and the expense of sm, and vice versa.

A Receiver Operating Characteristic (ROC) Curve is the set of (hm, sm) pairs achievable for any setting of t [62, 48, 125]. ROC was originally developed from signal detection theory and is now standard in medical diagnostic testing. It can be said that if one filter's ROC curve is uniformly above another filter's, the above filter will always outperform the lower filter. If the curves intersect, the more appropriate filter may be determined by examining the portion of the curve corresponding to deployment-specific requirements. Kolcz and Alspector[73] displayed their results using an ROC curve which is shown in figure 2.3.

Other curves have been used to display the trade off between ham misclassification and spam misclassification. A recall-precision curve characterizes the set of all achievable (sr, sp) pairs and hence is subject to the limitations we described for precision and recall. Sahami et al[104] used a spam precision-recall curve in their evaluation; the curve is shown in figure 2.2.

Martin et al.[84] suggest the use of a DET curve to measure filters within the context of document understanding. The DET curve is a ROC curve plotted on a normal deviate scale. The TREC Spam Track uses ROC curves plotted on a $logit$[5] scale, which is similar. On both scales, the curves tend to be linear.

The area under the ROC curve is a cumulative measure of the effectiveness of the filter over all possible threshold values. ROC area ($ROCA$) has a probabilistic interpretation: the probability that a random ham will receive a lower score than a random spam. ROC area is not influenced by the deployment-specific parameter $prev$. Average precision (AP) is the analog of $ROCA$ for recall-precision curves.

As an alternative to $ROCA$, a summary measure may be computed by projecting the ROC curve to a representative hm (or sm) value. $sm@hm$ is defined to be the value of sm such that (hm, sm) is on the ROC curve for some particular hm. $sm@hm = 5\%$, $sm@hm = 1\%$ and $sm@hm = 0.1\%$ have been reported [45, 130].

[5] $logit(x) = log(\frac{x}{1-x})$; see Chapter 4.

2.3.4 Test Apparatus

The objective of evaluating a spam filter is to measure its real life effectiveness. To achieve this, one must create a test apparatus that simulates the filter's operation. It is possible to evaluate filters in real life though there are several variables that make the measurements less accurate: variance in email, variance in users, variance in users' interaction, and variance in training, making the evaluation difficult to repeat. Though laboratory testing must compromise the authenticity of the filter's operation, it can control these variances and is repeatable. Only once laboratory evaluation has shown a filter to be effective is the expense of real-world testing justified.

The evaluation of learning filters dictates a test apparatus that incorporates training. The classical test apparatus for learning includes batch training in which the classifier is first given a set of training examples, and then required to filter a set of test examples. The process involves splitting the evaluation corpus into training and test sets. A more realistic apparatus incorporates on-line training. Studies using a batch apparatus are shown in table 2.2, labeled lab-batch.

k-fold cross validation may be used to increase the effective size of a small evaluation corpus [72]. A corpus of size n is divided randomly into k subsets of size $\frac{n}{k}$ where k is often 10. k different pairs of test and training sets are created by using each subset once as the test set with the remaining $k-1$ subsets combined for training. The average of the k results has roughly the same statistical power as an experiment with training set size $n \cdot \frac{k-1}{k}$ and test set size n. The validity of this approach depends on the assumption that the order of messages is unimportant[47]; that the ratio of ham to spam and the characteristics of the ham and spam messages are invariant with time. Consider, for example, a burst of five nearly identical spam messages that arrive in a short time interval. In a real email sequence, the filter might easily be flummoxed by the first of these messages, but learn its characteristics and correctly classify the rest. With ten-fold cross-valuation, it is nearly certain that each training set will contain several of these messages, so the filter's ability to classify the first-of-a-kind is essentially untested. In general, cross-validation tests the performance of a filter only after a fixed number of training examples; spam filter users seldom have several hundred or thousand labeled examples available for training prior to deployment. Studies using a ten fold method in table 2.2 are named lab-10_fold.

Modeling user feedback is substantially more challenging than assembling training and test sets for batch evaluation. Many learning filters have an interface so the user can correct misclassified emails. Some users update every misclassification while others never correct the system. Even users that complete all updates do so at various times due to the nature of reading email. The user's behaviour in updating the filter is a necessary component in evaluating such on-line filters. No studies of the effect of user feedback on filter effectiveness were found in the literature; one of the results in table 2.2 used an on-line test apparatus, training the filter on randomly-ordered messages.

2.3.5 Test data

The test corpora available in 2004 (table 2.2) were limited in one or more of the following ways:

- they contained private data, inaccessible to researchers for comparative studies;
- the messages were too few to yield adequate statistical power;
- the messages were sampled from non-representative sources;
- ham and spam were sampled from different sources;
- lossy transformations such as tokenization, feature selection, and header stripping were performed;
- message contents were obfuscated to preserve privacy.

Table 2.4 describes the spam corpora reported as of 2004. Corpora labeled public are available to be used for future studies; labeled private are available only to their proprietors.

Corpus	# email	# ham	# spam	availability	source
Spambase	4601	2788	1813	public	personal sample
Sahami et la.	1789	1538	251	private	1 yr recovered personal
SpamCop	1259	812	437	private	personal sample
Drucker(AT&T)	3000	2150	850	private	personal sample
Ling Spam	2412	1971	481	public	mailing list & spam
PU1/PU2/PU3/PUA	5957	3508	2449	public	personal
Filtron	6732	5109	1623	private	7 mth personal
Kolcz & Alspector	11408	6043	5365	private	group & personal & spam
crm114	2374	856	1518	private	personal
SpamAssassin	6034	4149	1885	public	news groups, personal sample
Tuttle et la.	2800	1000	1800	private	several personal sample
Holden1	1273	200	1073	private	1 mth personal

Table 2.4: Spam Corpora 2004

- The first public spam test collection, Spambase[122], has been heavily used in the Machine Learning research area. Each message is abstracted as a feature vector containing 57 attributes representing word frequencies, and other email characteristics such as number of capitals in the email. The text of the message is not available.
- Sahami et al[104] use a private corpus in which handcrafted features are extracted from one year of personal email (1789 messages).
- The SpamCop private corpus [93] consists of 1259 personal email without headers.
- Drucker[45] collected 3000 private messages from one AT&T staff member. Emails consisted of subject and body from which the 1000 best features were extracted.
- The public *SpamAssassin* corpus [120] is a collection of real email from multiple users and news groups. It is widely used by practitioners but has also been used by researchers. In 2004 it was the most realistic corpus available, albeit with certain limitations. The ham and spam were drawn from different sources during different time intervals. Some headers were redacted to preserve privacy.
- Ling Spam[2, 8] is an abstraction of 2412 ham messages from a mailing list and 481 spam messages from an individual recipient. We say abstraction because the messages are stripped of headers and line breaks, converted to lower case, tokenized and stemmed.
- Androutsopoulos et al.[2, 7] defines four public corpora – PU1, PU2, PU3 and PUA – with a total of 5957 messages (3508 ham and 2449 spam); each corpus abstracts and also obfuscates email from one recipient, so as to preserve privacy. In addition, repeated spam messages and ham messages from regular correspondents – about half the spam and eighty percent of the ham – are discarded in forming the corpus.
- The Filtron private corpus[88] consists of real email received by an individual over seven months. During this interval, 5109 ham and 1623 spam messages were received and classified. The messages have been abstracted into feature vectors.
- Kolcz and Alspector[73] assembled a private corpus of 11408 messages (6043 ham; 5365 spam) which were labeled according to category; each category was assigned an estimated cost of misclassification. The corpus was split into training and test sets in a 3:1 ratio. The messages have been abstracted into feature vectors.
- The CRM114 corpus is a private collection of personal email in original formatting.

- Tuttle et al.[130] created a method to capture the email messages and judgements using novel architecture that kept the data private. It then pushes tests to the users' corpora that only return the filter statistics. Seven users with up to 800 messages each were included in this study. The messages are abstracted into feature vectors.
- The Holden1[65] private corpus contains one month of personal email. The corpus includes qualitative descriptions of the misclassified ham messages, observing a preponderance of messages like welcome advertising, news clippings, mailing lists, etc.

2.4 Creating a better Filter

The motivating for my research was not to study evaluation methodology for its own sake but to create a better filter. Experience gained from combining IR systems[82] led me to hypothesize that combining would also work for spam filtering. Combining the output from multiple tools has been reported to improve information retrieval [90, 12, 116] and classification performance [70, 14, 139, 77, 66]. In information retrieval, a primary concern has been the combination of ranked lists of documents retrieved by different systems. The combination of the results from differently structured queries has also been investigated [13]. These techniques are generally applied to a batch process in which entire ranked lists are combined. The on-line spam filtering approach resembles ranked retrieval in that the *spamminess* score reported by the filter in effect ranks messages, but the ranking is incremental as the scores must be determined one message at a time, without knowledge of future messages.

Ensemble methods [43] have been the subject of much investigation for machine learning in general and for classification in particular. Bagging and boosting combine the results of several weak classifiers, typically employing the same algorithm over perturbed training sets or configuration parameters. Stacking [131], in contrast, uses a meta-learning technique to induce the best combination of stronger classifiers that employ distinct methods. In general, these investigations have employed a batch learning configuration and have been evaluated based on their binary classification effectiveness using separate training and test sets.

Neither naïve fusion nor stacking has been shown conclusively to have substantial benefit in this application. Dzeroski and Zenko state with respect to general text classification, "Typically, much better performance is achieved by stacking as opposed to voting," and "Our empirical evaluation of several recent stacking approaches shows they perform comparably to the best of the individual classifiers selected by cross-validation, but not better."[46] Hull et al., within the context of batch filtering, state, "We have found that simple averaging of probabilities or log odds ratios generates a significantly better ranking of documents," and "We generated [meta] parameter estimates using both linear and logistic regression but failed to reach the standard set by the simple averaging strategies."[66] Sakkis et al. stack Naïve Bayes and k-nearest-neighbor (KNN) classifiers using a KNN meta-classifier over various parameter configurations and observe that the best stacking configuration outperforms the best individual classifier configurations by a small margin:

> The results presented here motivate further work in the same direction. In particular, we are interested in combining more classifiers [...] Finally, it would be interesting to compare the performance of stacked generalization to other multi-classifier methods [...] .[105]

Segal et al. [113] employ a pipeline of purpose-built filters to analyze various aspects of email messages. At the end of the pipeline, if no filter has definitively classified the message, the scores from all filters are combined using linear coefficients computed by a non-linear optimizer, the combination showing improvement over the individual filters.

2.5 The Next Steps

Chapter 3 considers on-line filter evaluation methodology and its application to real spam filters and real email. Chapter 4 describes the TREC Spam Track, corpus creation, and associated tools. Chapter 5 harnesses the toolkit to create a better filter. Chapter 6 casts previous results in terms of standard measures and the major results that have been achieved building on the tools and methods described here.

Chapter 3

A Study of On-line Supervised Spam Filtering

A comparative evaluation of several commonly deployed open source spam filters was conducted. The filters were evaluated using a complete sequence of one user's email from an eight month period. Our study advances the methodology, scale, realism and repeatability of spam filter evaluation. Results from the study are an indicator of the filters' real world performance.

This approach is novel in that it closely models real filter usage, presenting to the filter a large sequence of real email messages, one at a time in chronological order, for classification. The same sequence of messages, under exactly the same conditions, is presented to several filters for the purpose of comparative analysis. Measures are computed which reflect a filter's effectiveness. Statistical confidence intervals, which estimate the extent to which the measured results might be due to chance, are computed for all measures.

An email corpus was created by capturing from the user a sequence of raw email messages along with a ham or spam label for each message. Errors in the labels were corrected using an iterative process to form a more accurate gold standard that was used for evaluation.

The study also examines the kinds of misclassification filters make. Each misclassified messages was labeled according genre (e.g. personal, advertising), and filter effectiveness for each genre was measured. Change in filter effectiveness over time is also measured, indicating the increasing difficulty of spam classification and improved filter effectiveness with more training.

3.1 Defining Spam

For the purpose of evaluating filters, Cormack and I[30] adopted the following definition of spam:

> "unsolicited, unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient."

Every message not fitting the definition was considered ham. Some illustrative examples follow.

- Personal email from a friend. This email is ham.
- Advertisement for V I A G R A from unknown sender. This is spam.
- Chain letter from a friend. Chain letters are annoying and should be spam but this email is neither indiscriminate nor from someone having no current relationship; therefore, it is ham.
- Email from your bank advertising a new mutual fund. This may be unsolicited and unwanted but, because of your prior relationship with the bank, the email is ham.
- Email from an unknown sender with a question about a project you worked on. This email is unsolicited and you have no relationship with the sender but it has not been sent indiscriminately so it is ham.
- A bulk email from an e-card organization you once used selling you on-line prescription drugs. The email is unsolicited, unwanted and though you had a relationship with the sender it is no longer current and is definitely outside the scope of that relationship; therefore, the email is spam.

3.2 Study Design

Each filter configuration was tested in a laboratory environment simulating the usage characterized by our model. In the interest of repeatability, we made two simplifying assumptions. We assumed that no filter used time-varying external resources, so that email messages captured at one time would be classified the same way later. We idealized the recipient's behaviour by assuming that he or she accurately and immediately reported all misclassified messages to the filter.

We captured all of the email received by one individual over a period of time. These messages were presented, one at a time, in chronological order, to each filter for classification. In addition, we extracted from each filter a spamminess score, indicating the filter's estimate of the likelihood that the classified message was spam. Immediately thereafter, a gold standard classification for each message was reported to the filter. The filter's classification, the filter's spamminess score, and the gold standard classification were recorded for later analysis. Confidence intervals were computed for each measure, under the assumption that the test email sequence sampled an infinite hypothetical population of materially similar email.[1]

3.3 Evaluating Public Spam Filters

3.3.1 Evaluation Methodology

Each trial run is an idealized[2] reproduction of X's behaviour over the eight months in which the email collection was originally delivered, with a different filter in place of SpamAssassin 2.60. The subject filter is presented with each message, with original headers, in the same order as originally delivered. Each filter was encapsulated using three common interface procedures: *filterinit, filtereval,* and *filtertrain. filterinit* sets the filter's memory to a clean initial state; *filtereval* is given an email message and returns a classification and a spamminess score; *filtertrain* is given an email message and the gold standard classification. Some filters require that *filtertrain* be invoked for every message (*train on everything*) while others require that *filtertrain* be invoked only for misclassified messages (*train on error*). We used the method suggested by each filter's documentation.

3.3.2 Subject Filters

We selected the current versions of six open-source filters whose deployment had been widely reported on the internet and in the popular press. At the time of the study a large number of classification techniques potentially relevant to spam filtering had been reported in the literature, an extensive search of available practical email filters yielded filters that used only a limited number of techniques, which we characterize as hand-coded rule bases, internal or external black lists and white lists, and content-based 'statistical'

[1]The notion of *population* has been the subject of historical and current philosophical debate[78]. We adopt Fisher's view [[49]] of an infinite hypothetical population.

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

or ‘Bayesian’ filters owing their heritage to Graham’s *A Plan for Spam*[56, 55] with improvements due to Robinson[102, 101].

Of the filters we selected, SpamAssassin[121] is a hybrid system which includes hand-coded spam-detection rules and a statistical learning component. The other filters – Bogofilter[99], CRM114 [134], DSPAM[137], SpamBayes[94], and SpamProbe[20] – are all ‘pure’ statistical learning systems, with only a few tacit rules such as those for tokenization.

Five different configurations of SpamAssassin were tested, in order to evaluate the roles and interactions of its various components. These five configurations were compared with one another, and with the in situ performance of SpamAssassin, which was deployed when the email for the test corpus was collected.

SpamAssassin contains two principal components: a set of static ad hoc rules that identify patterns associated with spam, and a Bayes filter fashioned from Graham’s and Robinson’s proposals. Each ad hoc rule has a predetermined weight; the weights of features observed in a particular message are summed to yield a combined spamminess score. The Bayes filter, on the other hand, is adaptive – it uses statistics from previously-classified messages to estimate the likelihood that a particular message is spam. This likelihood estimate is converted to a (possibly negative) weight which is added to the ad hoc spamminess score. The overall score is compared to a fixed threshold; the message is classified as spam if the score exceeds the threshold.

We tested several configurations of SpamAssassin 2.63 so as to evaluate the relative contributions of the ad hoc and Bayes components, and to evaluate various training regimens for the Bayes filter.

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

SA-Supervised. SpamAssassin 2.63 (both components) with the default threshold value of 5.0.

SA-Nolearn. SpamAssassin 2.63 (ad hoc component only) with the default threshold of 5.0.

SA-Bayes. SpamAssassin 2.63 (Bayes component only) with a threshold of 0.0.

SA-Standard. SpamAssassin 2.63 (Standard configuration with no user feedback) with a threshold of 5.0. SpamAssassin is configured by default to be used in a situation, such as a mail server, where misclassification errors go unreported. To this end, it includes an internal mechanism to train the Bayes component automatically, based on the spamminess score rendered by the ad hoc component alone. *filtertrain* is never invoked.

SA-Unsupervised. SpamAssassin 2.63 (Unsupervised automated feedback.) *filtertrain* is invoked after every message, but with SpamAssassin's output classification rather than the gold standard; that is, its own judgement is fed back to itself as if it were the gold standard.

SA-Human. Real-world baseline. These are the initial classification results that we captured from the memory of X's SpamAssassin 2.60 configuration. As such, they represent the classifications rendered in situ by the spam filter, as amended in real time in response to misclassification errors reported by X. These results show the combined effectiveness of spam filter and recipient under real-world conditions.

CRM114 (version 20040328-Blame St. Patrick-auto.1). We trained the system only after misclassifications, as suggested in the documentation. We did not use the whitelist or blacklist facilities supplied with CRM114. Filter memory size was set at 10000001 *buckets* for both ham and spam.

DSPAM (version 2.8.3). DSPAM 2.8.3 self-trains on every message it classifies, and annotates the message with a signature that contains information necessary for it to reverse this self-training. We altered our test setup to supply this annotated message, rather than the original, to *filtertrain*. We did not use the *purge* facility, which reduces the size of the statistical table maintained by DSPAM.

Bogofilter (version 0.17.5). Bogofilter is a Bayes filter, like SpamAssassin's, modeled after the proposals by Graham and Robinson. Bogofilter emphasizes simplicity and speed.

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

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

SA-Bayes. SpamAssassin 2.63 (Bayes component only). From the SpamAssassin comparison group.

3.3.3 Measures

In this study, test several hypotheses are tested. For those that are amenable to statistical inference we state confidence intervals and declare significant differences based on the error probability $\alpha = 0.05$.

Receiver operating characteristic (ROC) analysis is used to evaluate the trade-off between ham and spam misclassification probabilities. Using each of the numerical scores returned by a given filter, we conduct a hypothetical run to determine the ham and spam misclassification fractions that would have resulted had that score been used as a threshold. The set of pairs *(hm, 1-sm)* resulting from the hypothetical runs define a monotone non-decreasing function that is plotted as an ROC curve. As a summary measure of the relationship between ham and spam misclassification fractions over all possible thresholds, we present $1 - roca$, where roca is the area under the ROC curve. $1 - roca$ estimates the probability that a random spam message is (incorrectly) given a lower score than a random ham message. 1-roca estimates and 95% confidence intervals were computed using SPSS 12. It should be noted that filters that do not train on all messages would have to be run using a possible threshold to produce exact receiver operating characteristic analysis. This is because changing the threshold would change the messages the filter trained on. Due to computation constraints we only complete the default run so the receiver operating characteristic analysis should be considered estimates for these types of filters.

Logistic regression[5] is used to evaluate the effect of the number n of messages processed on the probability P of ham or spam misclassification (i.e. the *learning curve).* P and n are assumed to be related by the formula $logit(P) =_{def} log(\frac{P}{1-P}) = \alpha + n\beta$ (alternatively, $\frac{P}{1-P} = e^{\alpha}e^{n\beta}$) for some α and β. Maximum likelihood estimates for α and β, 95% confidence limits, and p-values (for the null hypothesis that $\beta = 0$) were computed using SPSS 12. $\frac{P}{1-P}$ is the *odds* (as opposed to the probability) of misclassification; i.e. the ratio of incorrect to correct classifications. e^{α} is the *initial odds* when $n = 0$, and $e^{n\beta}$ is the *odds ratio*; for every n messages the odds increase (or decrease) by a factor of $e^{n\beta}$. For small P, odds and probability are nearly equal, so we may consider $e^{n\beta}$ also to be the *risk ratio;* for every n messages the probability of misclassification changes by this same constant factor.

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

3.4 Test Corpus

We captured the email received by one individual (X) over an eight month period. These 49,086 messages were initially classified in real-time by SpamAssassin 2.60 [121] and placed in X's ham and spam files. X regularly examined both files and reported misclassification errors to SpamAssassin. X has had the same userid and domain name for 20 years; variants of X's email address have appeared on the Web, and in newsgroups. X has accounts on several machines which are forwarded to a common spool file, where they are stored permanently in the order received.

X began using a spam filter when the proportion of spam in his email began to exceed 20%, causing X to overlook two important messages which arrived amongst bursts of spam. Since then, X has used SpamAssassin 2.60 in a supervised configuration to classify this incoming mail. It was necessary to modify SpamAssassin to incorporate this use, as SpamAssassin was designed to be used primarily in an unsupervised configuration. User feedback was facilitated by two macros added to X's mail client. SpamAssassin records every judgement (rendered automatically and amended to reflect user feedback) in its learning database, so it was possible to recover our preliminary gold standard judgements from this database.

The corpus created from X's email will be referred to as the Mr X corpus.

3.4.1 Gold Standard Bootstrapping

Human adjudication is a necessary component of gold standard creation. Exhaustive adjudication is tedious and error-prone; therefore we have created a novel bootstrap method to improve both efficiency and accuracy[36]. The bootstrap method begins with an initial gold standard G_0. One or more filters is run using G_0 for feedback. The evaluation component reports all messages for which the filter and G_0 disagree. Each such message is re-adjudicated by the human and, where G_0 is found to be wrong, it is corrected. The result of all corrections is a new standard G_1. This process is repeated to form G_2, and so on, until $G_n = G_{n+1}$. Often filters misclassify the same email; this email does not need to be adjudicated during every iteration by the human.

Example of Gold Standard Bootstrap

Given a corpus with Email $\{E_0, E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9\}$ we need gold standard judgements: G=$\{J_0, J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9\}$ where J_i will be ham(H)

Email	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
G_0	H	S	H	S	H	S	H	S	H	S
Truth	H	H	S	H	S	S	S	H	H	S

Table 3.1: Example G_0

or spam(S). We will test filters F_1, F_2 ,F_3 against the corpus. To create the gold standard we must first start with an initial bootstrap G_0. Table 3.1 shows G_0. For this example every other email in G_0 has been assigned as spam. The truth row in the table is the actual classification for the corpus. It should be pointed out that truth can never be known for a real corpus and the gold standard is an estimate of truth.

Tables 3.2, 3.3 and 3.4 shows the adjudication process. To start, each filter is run using G_0 for feedback. If any of the filter's classification disagree with the G_0 for a given message, shown in bold, we adjudicated that message. Adjudications are completed by a human reading the email. Disagreements, indicated in bold, are updated in the gold standard. We then run the filters using the new gold standard, G_1, for feedback. The adjudication, update and re-run process continues until $G_n = G_{n+1}$. There are four changes made from G_0 to G_1 . From G_1 to G_2 two changes are made. G_3 is equal to G_2 so the bootstrap process is stopped.

Email	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
F_1	H	**H**	H	S	**S**	**H**	H	S	H	S
F_2	H	**H**	**S**	S	**S**	**H**	H	S	H	S
F_3	H	S	**S**	S	H	S	H	**H**	H	S
G_0	H	S	H	S	H	S	H	S	H	S
adjudicated		**H**	**S**		**S**	S		**H**		
G_1	H	**H**	**S**	S	**S**	S	H	**H**	H	S
Truth	H	H	S	H	S	S	S	H	H	S

Table 3.2: G_1 Bootstrap Example

Email	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
F_1	H	H	S	S	S	**H**	H	H	**S**	S
F_2	H	H	S	S	S	S	**S**	H	H	S
F_3	H	**S**	S	**H**	S	S	H	H	H	S
G_1	H	H	S	S	S	S	H	H	H	S
adjudicated		H		**H**		S	**S**		H	
G_2	H	H	S	**H**	S	S	**S**	H	H	S
Truth	H	H	S	H	S	S	S	H	H	S

Table 3.3: G_2 Bootstrap Example

Email	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
F_1	H	H	S	**S**	S	S	**H**	H	H	S
F_2	H	H	S	**S**	S	S	S	H	H	S
F_3	H	**S**	S	H	S	S	**H**	H	H	S
G_2	H	H	S	H	S	S	S	H	H	S
adjudicated		H		H			S			
G_3	H	H	S	H	S	S	S	H	H	S
Truth	H	H	S	H	S	S	S	H	H	S

Table 3.4: G_3 Bootstrap Example

Email	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
G_3	H	H	S	H	S	S	S	H	H	S
Truth	H	H	S	H	S	S	S	H	H	S
# of adjudications	0	3	1	0	1	2	1	1	1	0

Table 3.5: Number of Adjudication

Table 3.5 details the number of adjudication for each email. E_1 was adjudicated 3 times; adjudicating more than once can avoid human errors but experience indicates additional times has little benefit. We tended to only adjudicate up to two times. In this example 7 of the 10 email are adjudicated. This is much more than on a real corpus where only 3 out of the 10 email adjudication in G_0 were correct. A better G_0 can be constructed by using a non-learning filter or by judging a small number of email and using a learning filter in a batch process. It is possible that the gold standard achieved in this process will not be the same as truth. Consider if E_9 truth was ham but all filters and all bootstrap gold standard regard E_9 as spam; during evaluation all filters would benefit by not having this misclassification count against them. In real life this is unlikely to happen as filters perform well and tend not to make the same errors. Though for accurate evaluation all filters used in the evaluation should contribute to G_n and G_{n+1}. This constraint will make the evaluation fair to all filters.

Mr X Gold Standard

For the Mr X corpus G_0 consisted of the judgements rendered by SpamAssassin, amended to correct all misclassification errors reported by X. G_1 was created using *SA-Supervised* with G_0 as feedback. G_2 was created using *CRM114* and *DSPAM*. G_3, G_4 and G_5were created using all the filters but *SA-Human* as it wouldn't make any sense to include. It is important that all the filters were used in the bootstrap construction of G_5, the final gold standard has the net effect that every message reported as a ham or spam

	$S \to H$	$H \to S$
$G_0 \to G_1$	0	278
$G_1 \to G_2$	4	83
$G_2 \to G_3$	0	56
$G_3 \to G_4$	10	15
$G_4 \to G_5$	0	0
$G_0 \to G_5$	8	421
G_5	$\|H\| = 9038$	$\|S\| = 40048$

Table 3.6: Mr X Bootstrap Gold Standard Iterations

misclassification for any filter has been adjudicated by X.

Table: 3.6 show the revisions needed during the bootstrap process. $S \to H$ is the number of messages revised from spam to ham. $H \to S$ is the opposite.

3.4.2 Genre Categorization

To facilitate further analysis, we categorized messages – both ham and spam – into *genres* which may predict risk or cost of misclassification. For example, we suggest that individually addressed messages and news digest messages, while both ham, may present different levels of challenge to the filter and also different costs to the recipient, were they to be lost. After the filter tests were complete, each misclassified ham message was examined and assigned one of seven genres that we believed might be associated with the likelihood of misclassification and the importance of the email to the recipient. We also assigned a genre to each of a random sample ($n = 352$) of all incoming ham. Similarly, we assigned one of five different genres to each spam message misclassified by one or more of the four best-performing systems, and also to a random sample of spam messages ($n = 100$) misclassified by each of the other systems. We also assigned a genre to each of a random sample ($n = 142$) of all incoming spam.

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

Table 3.7: Spam as a fraction of incoming messages

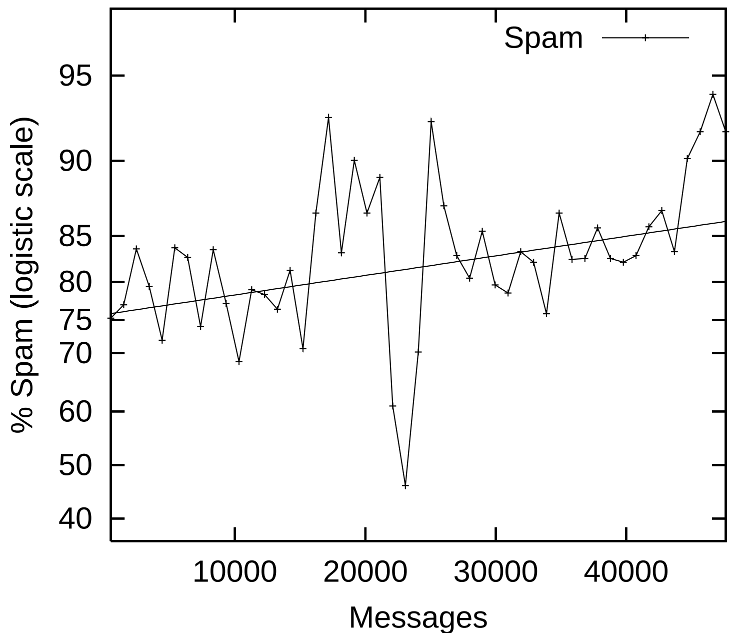


Figure 3.1: Spam Growth

3.4.3 Test Corpus Analysis

Table 3.7 summarizes the fraction of spam received by X as a function of the number of messages received. Although the overall spam fraction is 81.6%, logistic regression indicates that this fraction increased from 75.7% to 86.6% (an odds ratio of 2.0c7, $p < .001$) over the eight months during which our email stream was collected. Figure 3.1 shows a piece-wise approximation of this function juxtaposed with the regression line.

3.5 Results

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

Filter	Ham Misc.(%)	Spam Misc.(%)	Overall Misc.(%)	1-roca(%)
SA-Supervised	0.07 (0.02-0.14)	1.51 (1.39-1.63)	1.24 (1.15-1.35)	0.06 (0.04-0.07)
Bogofilter	0.08 (0.05-0.16)	6.63 (6.39-6.88)	5.43 (5.23-5.63)	0.08 (0.05-0.10)
SpamProbe	0.34 (0.23-0.49)	1.03 (0.93-1.14)	0.90 (0.82-0.99)	0.09 (0.05-0.13)
SA-Bayes	0.17 (0.09-0.27)	2.10 (1.96-2.24)	1.74 (1.63-1.86)	0.15 (0.11-0.18)
SpamBayes	0.17 (0.09-0.27)	5.86 (5.63-6.10)	4.81 (4.63-5.01)	0.16 (0.12-0.20)
SA-Nolearn	0.19 (0.11-0.30)	9.49 (9.21-9.78)	7.78 (7.54-8.02)	0.80 (0.74-0.86)
SA-Unsupervised	0.11 (0.05-0.20)	8.11 (7.84-8.38)	6.63 (6.41-6.86)	0.82 (0.76-0.88)
SA-Standard	0.07 (0.02-0.14)	7.49 (7.23-7.75)	6.12 (5.91-6.34)	1.00 (0.93-1.06)
DSPAM	1.28 (1.06-1.54)	1.98 (1.84-2.12)	1.85 (1.73-1.97)	1.03 (0.90-1.17)
CRM114	3.26 (2.91-3.65)	0.99 (0.90-1.09)	1.41 (1.31-1.52)	1.10 (0.94-1.27)

Table 3.8: Filter Misclassification

3.5.1 Filter Performance

Table 3.8 reports the performance of the filters. *SA-Supervised* misclassified 6 of 9,038 ham messages (0.07%) and 605 of 40,048 spam messages (1.51%). Overall, *SA-Supervised* misclassified 611 of 49,086 messages (1.24%). The area above the ROC curve, is 0.9994 which we report as 1-roca (%) or 0.06. The *SA-Supervised* filter is a committee of two distinct components: *SA-Nolearn,* a static rule-based filter, and *SA-Bayes*, a pure learning filter. Taken separately, each component shows inferior performance to the baseline according to all four measures. We note in particular that *SA-Supervised* shows 2.5 times fewer ham misclassifications than either *SA-Bayes* ($p < .004$) or *SA-Nolearn* ($p < .035$), two-thirds as many spam misclassifications as *SA-Bayes* ($p \approx 0.000$) and 6 times fewer spam misclassifications than SA-Nolearn ($p \approx 0.000$).

CRM114 has the best spam misclassification performance but it also has the worst ham misclassification performance. *DSPAM* and *CRM114* have significantly higher ham misclassification rates than all the other filters show.

Using the overall performance measure of 1-roca, *SA-Supervised*, *Bogofilter* and *SpamProbe* are the top performing filters. *SA-Bayes* and SpamBayes overlap confidence intervals with SpamProbe but are in the second tier. The rest of the filters have significantly lower 1-roca numbers.

SA-Standard uses SpamAssassin's default configuration: the same static and learning filter, but with the filter trained only on errors, as adjudicated by the difference in results between the learning filter and a separate (more conservative) internal invocation of the static filter. In contrast, SA-Unsupervised trains on every judgement returned by *filtereval.* Both runs are unsupervised in that they operate autonomously with no human

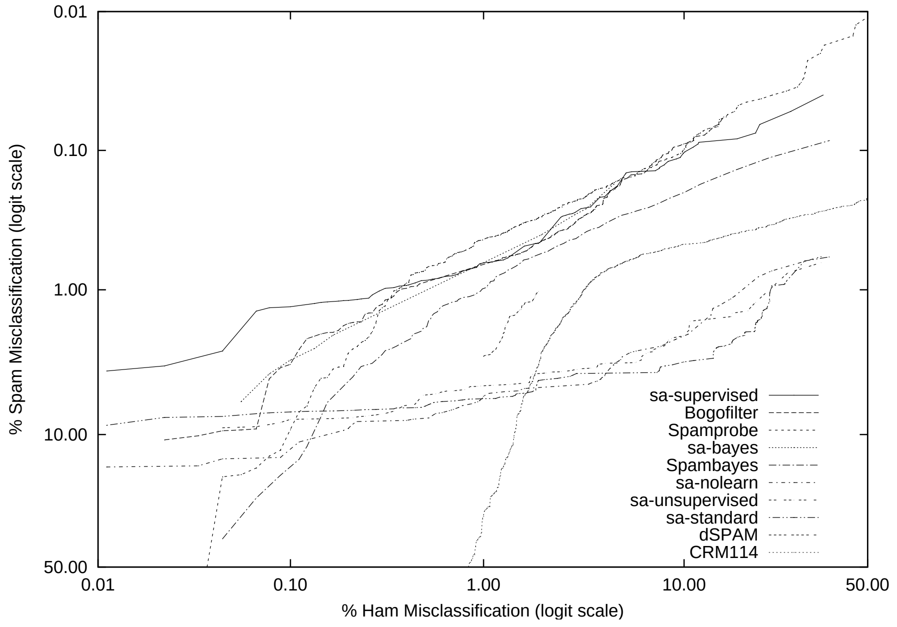


Figure 3.2: ROC Curves

intervention. As with SA-Supervised, both runs show fewer ham and spam misclassifications than either SA-Bayes or SA-Nolearn taken separately. Of the differences in ham misclassifications only the difference between SA-Standard and SA-Nolearn may be interpreted as significant ($p < .035$). All differences in spam misclassification are significant ($p \approx 0.000$).

Figure 3.2 reports the ROC curves of the filters.

The ROC curves show that SA-Supervised, Bogofiler, and Spamprobe outperform the other runs, performing better than SA-Bayes when ham misclassification is minimized and as well when spam misclassification is minimized. SA-Supervised, Bogofilter, Spamprobe, SA-Bayes and Spambayes all dominate the remaining runs. These runs, SA-Nolearn, SA-Standard, SA-Unsupervised, dSPAM, and CRM114 show ROC curves that intersect many times, indicating that their relative AUC scores are likely to be uninformative.

It appears that CRM114 filters the most spam having the lowest spam misclassification rate but, when we look at the ROC curves, all the other filters could choose a threshold to produce this spam misclassification rate that would greatly outperform CRM114 in ham misclassification. All filters dominate DSPAM and CRM114 by substantial margins at reasonable ham misclassification (less than 1%).

Filter	Initial Misc. (%)	Final Misc. (%)	Odds Ratio	p
Bogofilter	0.19 (0.06, 0.62)	0.02 (0.00, 0.17)	0.08 (0.00, 1.98)	0.12
CRM114	4.53 (3.71, 5.52)	2.08 (1.56, 2.75)	0.45 (0.29, 0.69)	0.00
DSPAM	1.52 (1.09, 2.12)	1.03 (0.67, 1.58)	0.68 (0.35, 1.33)	0.26
SA-Bayes	0.31 (0.13, 0.72)	0.06 (0.02, 0.26)	0.21 (0.03, 1.52)	0.12
SA-Human	0.01 (0.00, 0.09)	0.45 (0.15, 1.38)	54 (2, 1222)	0.01
SA-Nolearn	0.32 (0.14, 0.71)	0.09 (0.02, 0.31)	0.27 (0.04, 1.72)	0.17
SA-Standard	0.38 (0.12, 1.19)	0.00 (0.00, 0.07)	0.00 (0.00, 0.40)	0.02
SA-Supervised	0.19 (0.06, 0.66)	0.01 (0.00, 0.15)	0.05 (0.00, 1.80)	0.10
SA-Unsupervised	0.39 (0.15, 0.98)	0.01 (0.00, 0.10)	0.02 (0.00, 0.47)	0.01
SpamBayes	0.23 (0.10, 0.58)	0.10 (0.03, 0.37)	0.44 (0.07, 2.96)	0.40
SpamProbe	0.96 (0.56, 1.65)	0.05 (0.01, 0.17)	0.05 (0.01, 0.26)	0.00

Table 3.9: Ham Learning Performance

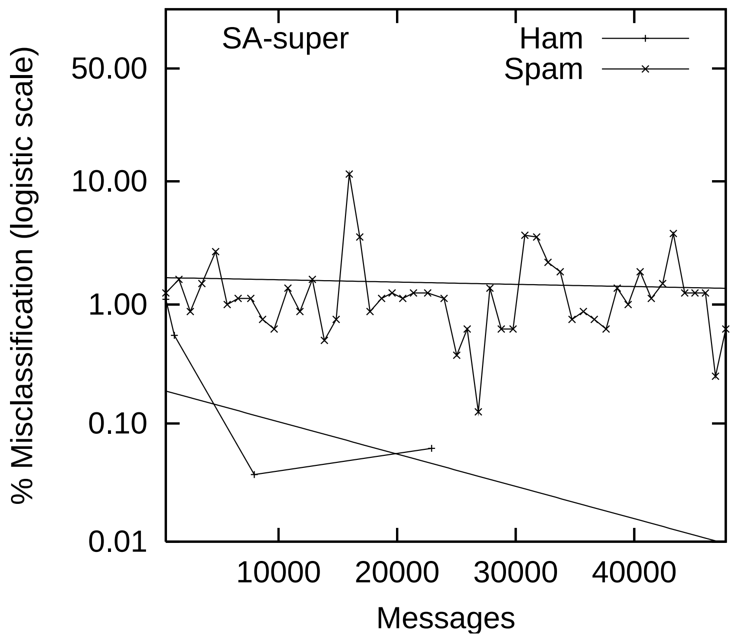
Filter	Initial Misc. (%)	Final Misc. (%)	Odds Ratio	p
Bogofilter	7.95 (7.41, 8.53)	5.50 (5.10, 5.94)	0.68 (0.59, 0.77)	0.00
CRM114	1.90 (1.61, 2.24)	0.45 (0.35, 0.57)	0.23 (0.16, 0.33)	0.00
DSPAM	7.02 (6.33, 7.77)	0.23 (0.18, 0.30)	0.03 (0.02, 0.04)	0.00
SA-Bayes	2.51 (2.21, 2.85)	1.74 (1.52, 2.00)	0.69 (0.55, 0.87)	0.00
SA-Human	1.67 (1.40, 1.98)	0.64 (0.52, 0.79)	0.38 (0.27, 0.53)	0.00
SA-Nolearn	7.73 (7.25, 8.25)	11.37 (10.76, 12.02)	1.53 (1.37, 1.72)	0.00
SA-Standard	16.07 (15.22, 16.96)	2.67 (2.43, 2.92)	0.14 (0.13, 0.16)	0.00
SA-Supervised	1.68 (1.44, 1.96)	1.36 (1.16, 1.59)	0.81 (0.61, 1.07)	0.13
SA-Unsupervised	18.03 (17.13, 18.98)	2.67 (2.44, 2.92)	0.12 (0.11, 0.14)	0.00
SpamBayes	5.91 (5.46, 6.39)	5.82 (5.38, 6.29)	0.99 (0.85, 1.14)	0.82
SpamProbe	1.29 (1.08, 1.56)	0.81 (0.67, 1.00)	0.63 (0.45, 0.88)	0.01

Table 3.10: Spam Learning Performance

3.5.2 Effects of Learning on Filter Performance

Tables 3.9 and 3.10 summarize the ham and spam misclassification fractions as functions of the number of messages processed. Each row estimates the initial misclassification proportion, the final misclassification proportion, and the odds ratio between the two. 95% confidence limits and p-values are given for each. Figures 3.3 and 3.4 provide a selection of the graphical representations of these functions; for all the learning curves refer to Cormack and Lynam[31].

The learning filters show no apparent degradation in performance over the eight-month interval. All learning filters show a reduction in both ham and spam misclassification fractions as more messages are processed, though not all reductions are large or



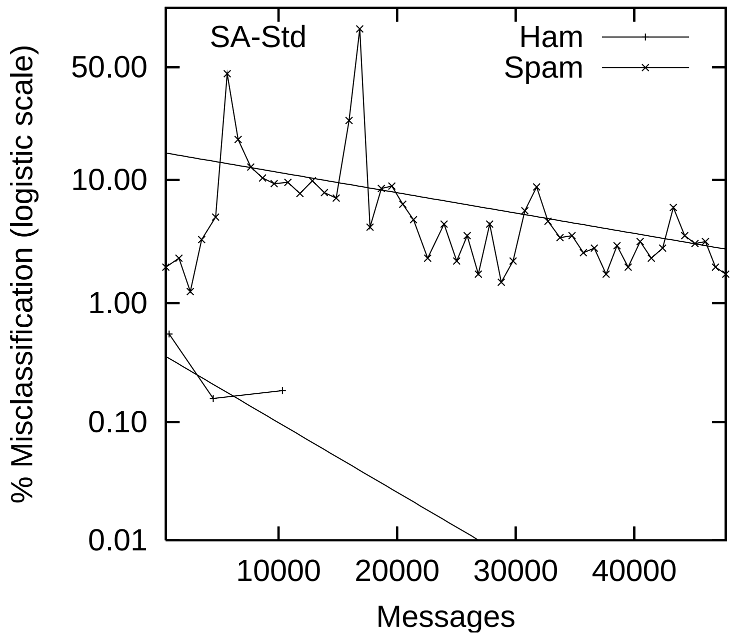
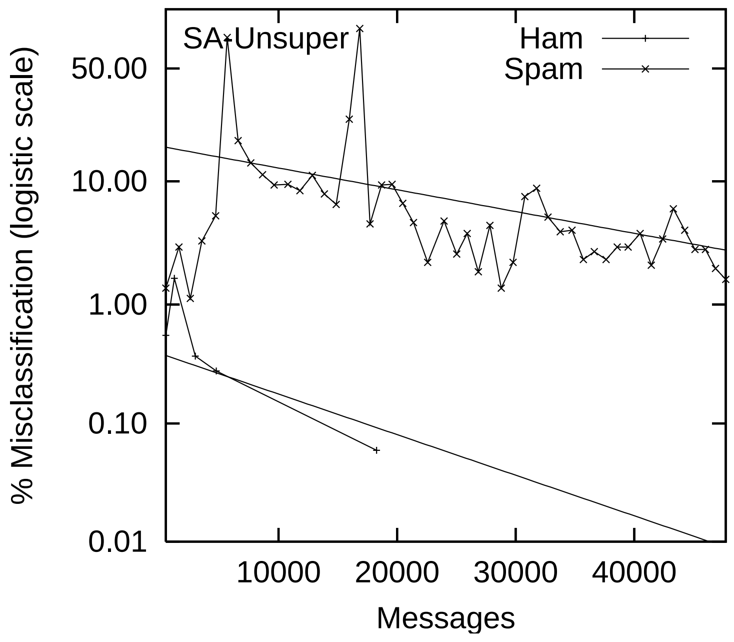


Figure 3.3: Learning Curves 1



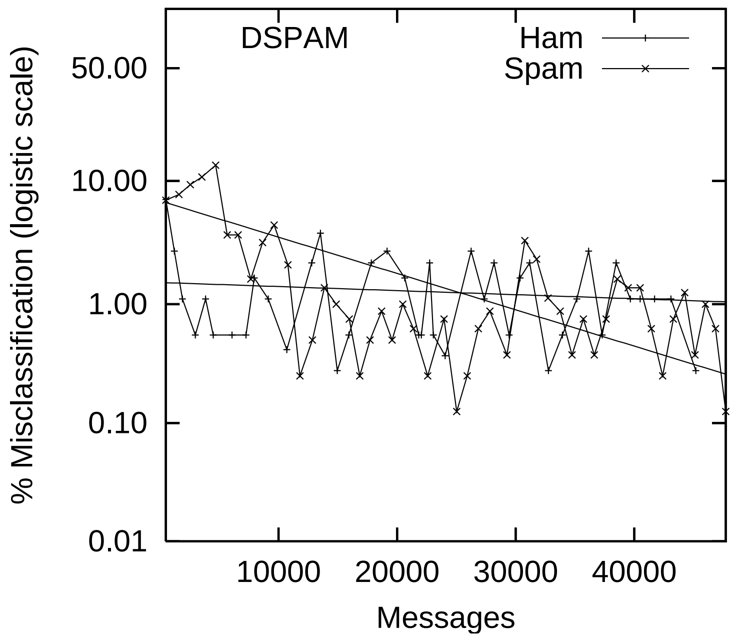


Figure 3.4: Learning Curves 2

significant. In particular, confidence intervals for the ham misclassification odds ratio are very large, due to the fact that the curve is fitted to few points - of the order of ten for the better-performing runs. Subject to this caveat, the plotted curves show a good fit between piece-wise approximation and logistic regression. Possible exceptions are DSPAM, SA-Standard, and SA-Unsupervised. DSPAM's spam misclassification curve, shown in figure 3.4, has a piece-wise approximation that appears to be more concave than the regression curve. SA-Standard (figure 3.3)and SA-Unsupervised (figure 3.4) both indicate substantially lower spam misclassification rates prior to the virus-induced anomaly at message 6,000, followed by consistent improvement notwithstanding the anomaly[3] at message 17,000. We observe that the initial misclassification fraction of a number of systems is substantially better than the final misclassification fraction of others.

3.5.3 Human with Filter Performance

SA-Human uses essentially the same configuration as SA-Supervised, but the system was supervised by X in real-time. That is, for every misclassification observed by X, the system was retrained and the human-corrected classification was recorded as the result for SA-Human. While SA-Human resulted in two more ham misclassifications than SA-Supervised (i.e. 8 vs. 6) no significant difference can be inferred. SA-Human resulted in two-thirds as many spam misclassifications ($p \approx 0.000$). Table: 3.11 details SA-Humans misclassification performance.

Filter	Ham Misc.(%)	Spam Misc.(%)	Overall Misc.(%)	1-roca(%)
SA-Human	0.09 (0.04-0.18)	1.06 (0.97-1.17)	0.88 (0.80-0.97)	-

Table 3.11: SA-Human Misclassification

We include SA-Human Learning analysis(tables: 3.9, 3.10 and figure: 3.5), as a real-world foil to our laboratory results. SA-Human's ham misclassification fraction shows a large significant increase with a huge confidence interval (odds ratio $54(2, 1222)$), indicating that this measurement is unstable, rather than that X suffered some degeneration in discriminatory ability. Further investigation reveals that the positive odds ratio may be accounted for entirely by three automated (but legitimate) messages received the same day from the same source. SA-Human's apparent decrease in spam misclassification may also be accounted for by the anomalous spike at 17,000 messages.

[3]Due to *backscatter*, as defined in section 3.5.3

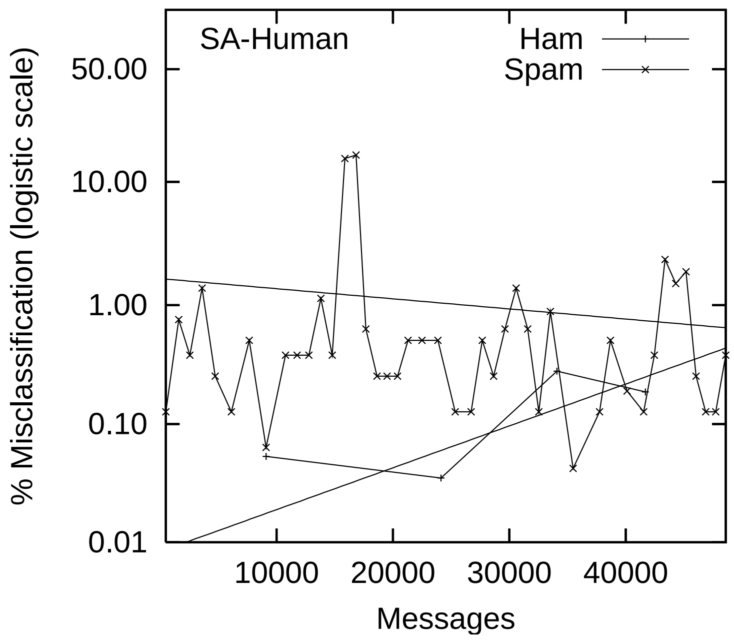


Figure 3.5: SA-Human Learning

3.6 Misclassification by Genre

In the course of examining the misclassified messages, we identified several message genres that we suspected might be associated with the filters' performance.

Ham *Misclassification Genres*

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

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

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

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

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

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

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

Table 3.12 shows the number of misclassified ham messages, by genre, for each filter. Also shown is an estimate of the proportion of all ham represented by each genre. Four of the runs have no *personal* misclassifications, a much lower fraction than would be suggested by the fact that this genre comprises 51% of all ham. At the other end of the spectrum, CRM114 misclassified 135 *personal* ham messages, or about 3% of all such messages. DSPAM also misclassified a high number of *personal* messages: 35, or about 0.75% of the total.

Filter	Advertising	Cold Call	Delivery	List	News	Personal	Transaction	Total
SA-Standard	4	2	0	0	0	0	0	6
SA-Super	1	0	0	1	1	0	3	6
Bogofilter	1	0	0	2	1	0	3	7
SA-Human	0	0	0	3	4	0	1	8
SA-Unsuper	5	0	0	1	0	1	3	10
SA-Bayes	1	0	0	4	1	1	8	15
SpamBayes	1	0	2	5	1	3	3	15
SA-Nolearn	1	0	4	0	3	9	0	17
SpamProbe	3	2	4	5	1	8	8	31
DSPAM	15	5	9	28	6	35	18	116
CRM114	7	15	13	78	10	135	37	295
Incoming Ham	0%	1%	17%	13%	14%	51%	4%	9038

Table 3.12: Ham Misclassification by Genre

Spam Misclassification Genres

1. *Advertising.* Messages sent indiscriminately to X aimed at acquiring some or all of X's wealth.
2. *Backscatter.* Delivery messages from a third-party server, rejecting a message not sent by X, but forged to appear to have been sent by X. These messages are deemed to be spam (as opposed to Delivery ham messages) because they are a direct consequence of spam.
3. *Demographic.* Advertising messages for goods and services of marginal value sent to a specific demographic group to which X belongs.
4. *Targeted.* Messages addressed to X for no reason other than X's membership in a broad identifiable group (profession, geographic location, appearance on a subject-related web-page, etc.).
5. *Virus.* Messages that contain malware.

In general, *advertising, cold call*, and *delivery* messages each represent a small proportion of overall ham and a disproportionately large number of misclassifications. *Personal*

Filter	Advertising	Backscatter	Demographic	Targeted	Virus	Total
CRM114	72%	8%	12%	4%	4%	397
SA-Human	14%	66%	10%	7%	4%	413
Spamprobe	48%	17%	17%	7%	12%	421
SA-Super	28%	36%	22%	5%	9%	605
DSPAM	58%	8%	17%	3%	14%	791
SA-Bayes	45%	19%	17%	8%	11%	840
SpamBayes	50%	16%	25%	7%	2%	2348
Bogofilter	68%	14%	10%	2%	6%	2656
SA-Standard	17%	29%	5%	0%	49%	2999
SA-Unsupervised	9%	31%	7%	1%	52%	3246
SA-Nolearn	51%	24%	5%	0%	20%	3802
Incoming Spam	92%	1%	0%	0%	8%	40048

Table 3.13: Spam Misclassification by Genre

messages represent disproportionately few misclassifications, while *transaction, list,* and *news* fall in between.

Table 3.13 shows the estimated fraction of misclassified spam messages, by genre, for each filter, as well as the fraction of all spam represented by each genre. The vast majority of spam messages are *advertising,* with *backscatter* representing a mere 1%. Yet nearly as many *backscatter* messages are misclassified. In particular, we note that SA-Human and SA-Super misclassified a fraction of backscatter messages approaching or exceeding 50%. Three-fifths of all of SA-Human's misclassifications are attributable to misclassified *backscatter.* The reason for this is that X was overwhelmed by the burst of *backscatter* occurring at 17,000, and skipped over many of these messages without recording a judgement.[4]

3.7 Spam Evolution

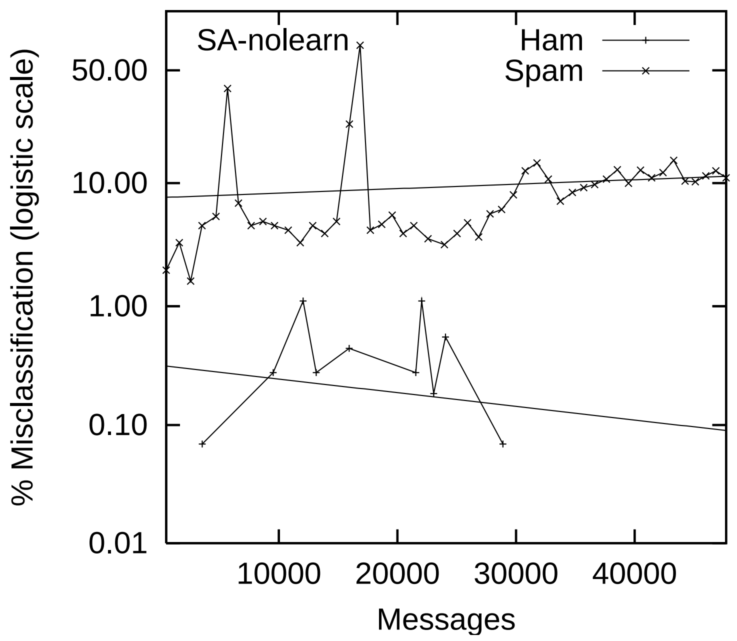


Figure 3.6: SA-Nolearn "Learning" Curve

In examining the "learning" performance of SA-Nolearn; as this system has no learning component, its performance may be used to gage any change in 'difficulty' of the

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

spam messages over the eight months. Table 3.10 shows that SA-Nolearn's spam misclassification fraction increases from 7.73% to 11.37% ($p < .001$), indicating that the nature of spam has changed so as to make it 'more difficult.' Figure 3.6 confirms this trend, but also shows anomalous spikes in misclassifications centered at about 6,000 and 17,000 messages. SA-Nolearn's ham misclassification fraction shows no significant slope over the eight-month interval. In contrast, the learning filters show no apparent degradation.

3.8 Analysis

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

A supervised filter contributes significantly to the effectiveness of SpamAssassin's static component. The supervised filter alone performed better than the static rules alone, but not as well as the combination of the two. This would indicate that by combining filters, even better performance could be achieved. Chapter 5 will explore combining filters in greater detail.

The choice of threshold parameters dominates the observed differences in performance among the four filters (Bogofilter, SA-Bayes, SpamProbe, SpamBayes) implementing methods derived from Graham's and Robinson's proposals. Each shows a different trade off between ham accuracy and spam accuracy. ROC analysis shows that the differences not accountable to threshold setting, if any, are small and observable only when the ham misclassification probability is low (i.e. $hm < 0.1\%$). The other filters (DSPAM, CRM114) show lower performance over all threshold settings.

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

In completing this study filterinit, filtereval and filtertrain had to be implemented for each filter. This was both tedious and time consuming. A public evaluation tool with a standard and well defined interface was needed. This insight was fundamental in the development of the Spam Filtering Evaluation Tool Kit.

The study also showed that though the corpus had 9038 ham messages, several systems misclassified less than 10 of these messages. Computing comparative statistical analysis on numbers less than ten is very difficult. This indicated to us that a large evaluation was needed and it should be completed over a variety of corpora both large and small. This leads to the introduction of the spam track at TREC.

Chapter 4

Standardized Evaluation

To improve spam filtering one must be able to measure the improvement; standardized evaluation can support this effort. Existing evaluation methodology was not sufficient to allow comparisons. We have created a standardized evaluation, specifically the Spam Filtering Evaluation Toolkit. The toolkit was created to provide a universal method for running and evaluating arbitrary spam filters. It can be used with Windows or Linux. For a filter to be evaluated by the toolkit, it must implement five command-line operations (initialize, classify, train ham, train spam, finalize).

The toolkit requires a corpus to evaluate filters against. We have created several corpora. The corpora are divided into two types: private and public. A private corpus can only be used by the owner of the email. This means to use a private corpus the owner must complete the evaluation for that corpus. A public corpus contains email that anyone can read. Finding personal email that can be made public is a difficult task but, due to a court injunction against the Enron corporation, a large amount of email was made available. These email were used to construct the TREC 2005 Public Spam Corpus.[1] Creating a gold standard classification for these Enron email involved the previously described iterative process as well as an extra step related to classifying another person's email. We show how quality of the corpus is improved with each iteration. We also exploited the folder information in determining genre for the TREC public spam corpora.

Both the toolkit and created corpora were used at the 2005 TREC spam filtering track. Twelve groups and 53 filters participated in the spam track. Each of the filters was submitted to TREC for evaluation. Groups both attended the TREC conference

[1] `http://trec.nist.gov/data/spam.html`

and wrote papers detailing their filters' methods. The results from TREC show learning filters do an adequate job of filtering spam. It also shows they get better over time. The TREC spam track was continued in 2006 and 2007 as well. Its methodology has been used in other studies. The TREC spam track has provided an environment for creating new and improved filtering methods.

4.1 TREC Spam Filtering Evaluation Toolkit

The TREC Spam Filtering Evaluation Toolkit is a standard testing framework that is designed to model a spam filter's usage as closely as possible, to measure quantities that reflect the filter's effectiveness for its intended purpose, and to yield repeatable (i.e. controlled and statistically valid) results. The *TREC Spam Filter Evaluation Toolkit* is free software that, given a corpus and a filter, automatically runs the filter on each message in the corpus, compares the result to the *gold standard* for the corpus, and reports effectiveness measures with 95% confidence limits.

4.1.1 The Need for Standardized Evaluation Methodology

Before this work there was a wide disparity of methods, claims, and measures employed. These disparities made the determination of the performance of filter methods difficult. As a result, it was very hard to make improvements and know, with any certainty, that the improvements were actually better.

Much of the problem with disparities is related to two variants: firstly, the use of different corpora; secondly, the diverse evaluation methodologies. We created a publicly available toolkit so that research could not only use the same methodology but the same corpora.

4.1.2 Toolkits required filter's commands

The toolkit was created to provide a universal method for running and evaluating arbitrary spam filters. It can be used with Windows or Linux. Each filter must implement the five command-line operations required by the toolkit:

- **initialize** – creates any files or servers necessary for the operation of the filter

- **classify** *message* – returns ham/spam classification and spamminess score for *message*
- **train ham** *message* – informs filter of correct (ham) classification for previously classified *message*
- **train spam** *message* – informs filter of correct (spam) classification for previously classified *message*
- **finalize** – removes any files or servers created by the filter.

4.1.3 Using the Toolkit

The toolkit takes as input a test corpus consisting of a set of email messages, one per file, and an index file indicating the chronological sequence and gold standard judgements for the messages. It calls on the filter to classify each message in turn, records the result, and communicates the gold standard judgement to the filter before proceeding to the next message.

The recorded results are post-processed by an evaluation component supplied with the toolkit. This component computes statistics, confidence intervals, and graphs summarizing the filter's performance.

4.1.4 Toolkit Corpus Format

The toolkit requires corpora to be in a set format. Each email in the corpus is in a separate file. The toolkit also requires an index file. Each line of the index file contains a gold standard judgement followed by the location of corresponding email file. The order of the index file is the order in which the emails will be fed to the toolkit.

4.1.5 Toolkit's Test Corpus

We have reformatted the Spamassassin corpus to work with the toolkit to use as a test corpus. The February 28, 2003 Spamassassin data set was used.[2]

[2]The files used in the toolkit's test corpus are:
http://spamassassin.apache.org/publiccorpus/20030228_easy_ham.tar.bz2
http://spamassassin.apache.org/publiccorpus/20030228_easy_ham_2.tar.bz2
http://spamassassin.apache.org/publiccorpus/20030228_hard_ham.tar.bz2
http://spamassassin.apache.org/publiccorpus/20030228_spam.tar.bz2
http://spamassassin.apache.org/publiccorpus/20030228_spam_2.tar.bz2

The index file is a list of these email with judgements in arrival date order. The date order was created using the Received: timestamp and, if this was not present, the Date: field.

An example of a corpus index file is as follows:

```
spam  spam_2/00026.c62c9f08db4ee1b99626dbae575008fe
spam  spam_2/00021.07d9ab534bbfba9020145659008a3a14
...
ham   hard_ham/00009.ddea79a02a9978cb3dafef3c05ff37a6
...
spam  spam_2/01391.d84700fa88ef00525b05a2d9c64fa654
```

4.1.6 Computed Graphs and Measures

The toolkit calculates the following previously defined measures:

- Contingency Table (true positives, false positives, false negatives, true negatives)
- *hm%,* is the filter's default threshold ham misclassification rate
- *sm%,* is the filter's default threshold spam misclassification rate
- *misc*%, is the filter's default threshold overall misclassification rate
- $(1 - ROCA)(\%)$, is the area above the ROC curve as a percent
- *sm*%@*hm*% for various values of *hm*%
- *hm*%@*sm*% for various values of *sm*%
- logistic average misclassification rate, *lam*, a new summary measure
- Learning curve, initial and final values.

An example result file, bogofilter on spamassassin corpus, produced by the toolkit can be seen in figure 4.1 with the ROC curve shown in figure 4.2 and the learning curve in figure 4.3.

```
<h1> Results for bogo </h1>
<pre>
         Gold Standard
       +--------------------+
       |       ham    spam |
       |                   |
Filter | ham  4141     451 |
Result |spam     8    1434 |
       +--------------------+
Total          4149   1885

ham%:  0.19 (0.08-0.38)
spam%:  23.93 (22.02-25.92)
misc%:  7.61 (6.95-8.30)

1-ROCA%:  0.209005 (0.139822 - 0.312312)

<a href=graphs/bogo.roc.pdf>Graph of ROC Curve</a>
<a href=graphs/roc.pdf>Combined Graph of ROC Curves</a>

Logistic Averages
HAM%:  0.19 (0.10 - 0.39)
SPAM%:  23.93 (22.05 - 25.90)
LAM%:  2.41 (1.71 - 3.38)

LEARNING CURVE
HAM% Initial:  46.53 (10.62 - 86.44) Final:  3.6e-10 (0.00 - 0.00)
SPAM% Initial:  21.02 (18.43 - 23.87) Final:  31 (25.53 - 37.91)
LAM% Initial:  32.49 (15.05 - 56.65) Final:  0.00013 (0.00 - 0.06)

<a href=graphs/bogo.pdf>Graph of Learning Curve</a>
<a href=graphs/lam.pdf>Combined Graph of Learning Curves</a>
```

Figure 4.1: Example Toolkit Result File

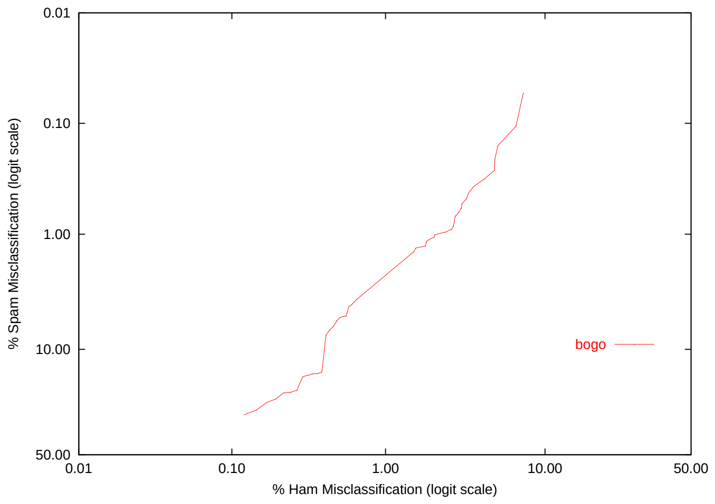


Figure 4.2: Example Toolkit ROC Curve

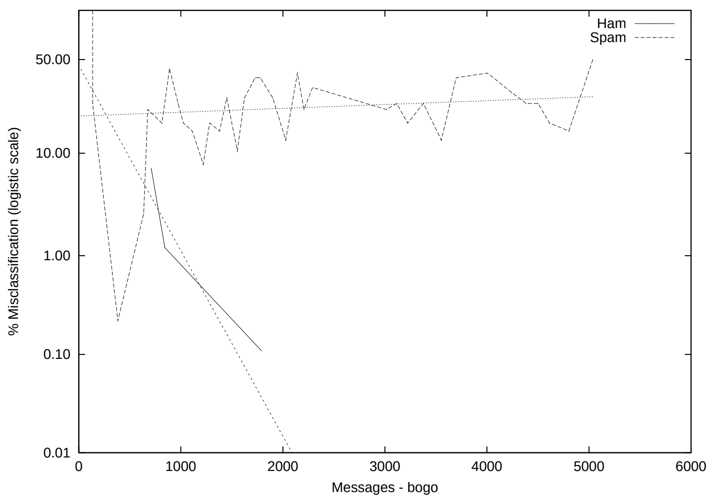


Figure 4.3: Example Toolkit Learning Curve

While $(1 - ROCA)(\%)$ provides an amenable measure for filters that return a score, some filters return only a categorical ham or spam result. For such classifiers, the single pair (fpr, fnr) does not yield a meaningful ROC curve, so an alternate summary measure is needed. It has been observed [52] that the diagnostic odds ratio, $dor = \frac{(1-hm)\cdot(1-sm)}{hm\cdot sm}$ is, for many diagnostic tests, effectively invariant over a large number of threshold settings. In terms of the logit-scaled ROC curve, constant *dor* appears as a linear curve with slope 1. The TREC 2005 results generally show this behaviour (e.g. figure 4.7)[3], particularly for the good filters. Intuitively, a change in threshold setting that increases the odds of misclassifying ham by some multiplicative factor tends to decrease the odds of misclassifying spam by the same factor. Therefore *dor* is a useful summary measure largely uninfluenced by threshold setting. In any event, there is no direct dependence on deployment-specific parameters like prevalence and the consequences of spam and ham misclassification. Instead of *dor* we use *logistic average misclassification* rate, which is equivalent under a monotone transform: $LAM = logit^{-1}(\frac{logit(fpr)+logit(fnr)}{2}) = logit^{-1}(log(dor^{-0.5}))$. Note that the value LAM is necessarily between hm and sm; when t is set to equalize error rates, we have $sm = hm = LAM$.

Under the assumption that *dor* is invariant, it is possible to estimate (hm', sm') from (hm, sm) by solving the equation

$$\frac{(1-hm)\cdot(1-sm)}{hm\cdot sm} \approx \frac{(1-hm')\cdot(1-sm')}{hm'\cdot sm'}\,.$$

It is further possible to estimate $ROCA \approx \int_0^1 (sm)\,d\,(hm)$.

4.2 Public Corpus Creation

One of the most difficult aspects of evaluating spam filters is finding or creating a suitable corpus. Because email is private in nature, no one wants their email available to the public as a test corpus. Even if someone wanted to make their email public the privacy of senders has to be considered. An alternative is to allow the corpus to remain private but the filters must be all tested on the same private corpus. Several large private corpora were created for TREC to measure the impact of corpus variation. As expected, the large corpora provided higher levels of accuracy in determining filter performance.

Prior to this research, the SpamAssassin corpus[120] was the only publicly available

[3]The one filter that demonstrates a non-linear logit-ROC curve was found to have a serious bug. When corrected its logit-ROC curve was also linear.

collection of email in nearly raw form. Though the corpus was very useful and definitely better than nothing it has some drawbacks. First, it is fairly small, making evaluation statistics unreliable. Second, it contains email from news groups and is from multiple users so it does not represent a natural email stream. Third, much of the header data has been removed. Also, it has no real chronological order. Simply changing the order the email are fed to the filter can greatly effect evaluated performance. Each of these shortcomings compromises evaluation. A large public corpus was needed for three reasons: to measure incremental improvements, provide usable evaluation statistics, and to vet public evaluation results against private ones.

4.2.1 Email Collection

It is a simple matter to capture all the email delivered to a recipient or a set of recipients. Using this captured email in a public corpus is not so simple. Few individuals are willing to publish their email, because doing so would compromise their privacy and the privacy of their correspondents. So we are left with the choice between using an artificial public collection of messages and using a more realistic collection that must be kept private.

Artificial collections [120, 88, 6] may be created by using mailing list messages as opposed to personal email, by selecting non-sensitive messages from a real email collection, by mixing messages from diverse sources, or by obfuscating genuine messages.[4] All of these approaches conflict with our design criteria – that real filter usage be modeled as closely as possible – and may compromise the very information that filters use to discriminate ham from spam, either by removing pertinent details or by introducing extraneous information that may aid or hinder the filter.

It is very hard to find real user email that is publicly available due to the privacy concerns of capturing a stream of email and making it public. Fortunately, real email was made public as part of the Federal Energy Regulatory Commission's (FERC) investigation against the Enron Corporation. Over one million messages and files from 150 Enron employees were released to the public.

The CMU Enron dataset [28, 71] contains around 500,000 of these emails with over half being duplicates. The messages have been altered substantially, replacing the original headers by synthetic ones and removing attachments. For these reasons, the CMU Enron dataset was rejected as a source of messages.

[4]The majority of filters we have evaluated exhibit pathologies on the PU obfuscated corpora.

Instead, we downloaded all public available Enron data directly from the FERC servers. Though there were over a million messages, many of the messages were calendar appointments and other non-email associated with email applications. There were approximately 250,000 emails after removing duplicates. Of these some 100,000 have headers; but only 43,000 have "Received:" line meaning the headers were mostly complete. From these 43,000 messages, the TREC public spam corpus was created. The FERC separated attachments from the email so extra steps had to be taken to re-attach the files. The publicized Enron email were from 2001 and 2002. During this time the percentage of spam was much lower. Of the 43,000 email, around 2,400 were spam.

The 43,000 Enron messages were augmented by approximately 50,000 spam messages collected by Guenter in 2005.[5] The headers of these messages were altered so as to appear that they were delivered to the Enron mail server during the same time frame (summer 2001 through summer 2002). "To:" and "From:" headers, as well as the message bodies, were altered to substitute the names and email addresses of Enron employees for those of the original recipients. Spamassassin and Bogofilter were run on the corpora, and their dictionaries examined, to identify artifacts that might identify these messages. A handful were detected and removed; for example, incorrect use of daylight saving time, and incorrect versions of server software in header information.

4.2.2 Public Corpus Gold Standard

In creating the TREC public corpus gold standard we employed the bootstrap method described in the previous chapter.

G_0 was constructed using SpamAssassin 2.63 with user feedback disabled. Subsequent iterations used a number of filters – SpamAssassin, Bogofilter, Spamprobe and CRM-114, interleaved with human assessments for all cases in which the filter disagreed with the current gold standard. This process identified about 5% of the messages as spam.

It was problematic to adjudicate many messages because it was difficult to glean the relationship between the sender and the receiver. In particular, the collection has a preponderance of sports betting pool announcements, stock market tips, and religious bulk mail that was initially adjudicated as spam but later re-adjudicated as ham. Advertising from vendors whose relationship with the recipient was tenuous presented an adjudication challenge.

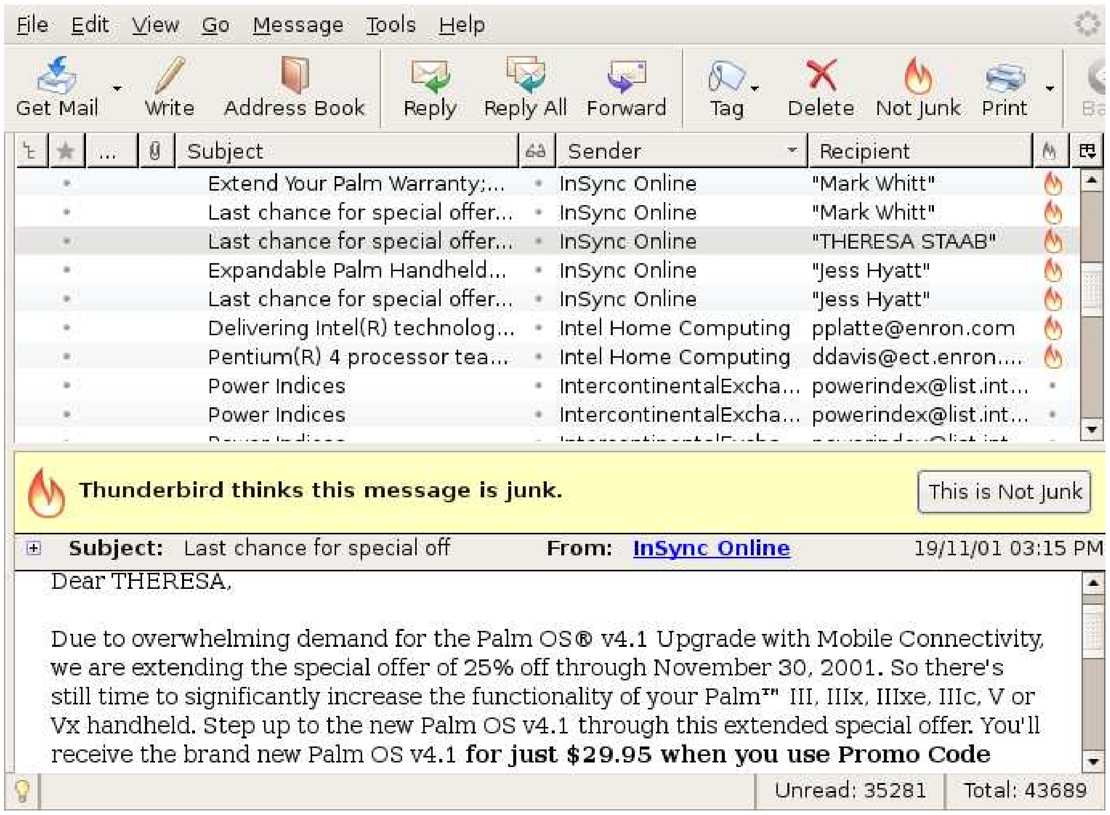
[5] `www.em.ca/~bruceg`

During this process, the need arose to view the messages by sender; for example, once the adjudicator decides that a particular sports pool is indeed by subscription, it is more efficient and probably more accurate to adjudicate all messages from the same sender at one time. Similarly, in determining whether or not a particular "newsletter" is spam, it is desirable to identify all of its recipients. This observation occasioned the design and use of a new tool for adjudication – one that allows the adjudicator to use full-text retrieval to look for evidence and to ensure consistent judgements.

The adjudication tool exploited Mozilla Thunderbird message filters and GUI interface. Adjudicating someone else's email involves more detail than merely reading the disputed email and determining if it is spam or not. To construct the TREC public gold standard we start by creating the G_0. As mentioned G_0, was constructed using SpamAssassin 2.63 with no user feedback. Next, SpamAssassin, Bogofilter, Spamprobe and CRM-114 filters were run using G_0 as feedback. To adjudicate G_0 the corpus was concatenated into an email stream with the following headers added to each email:

- G_i
- file location
- adjudicated
- combined classification
- combined score
- origin
- folder
- each filter's classification
- each filter's score

The G_i header is the current gold standard bootstrap; the adjudicated header indicates if the email has been adjudicated or not. The file location header is as detailed in the index file. The combined classification is a combination of filters used; if any of the filters considered the email spam, the combined classification did. The combined score was the number of filters that classified the email as spam. The origin and folder headers come from the data downloaded from FERC. The origin header indicates to the user where the



Subject	Sender	Recipient
Extend Your Palm Warranty;...	InSync Online	"Mark Whitt"
Last chance for special offer...	InSync Online	"Mark Whitt"
Last chance for special offer...	InSync Online	"THERESA STAAB"
Expandable Palm Handheld...	InSync Online	"Jess Hyatt"
Last chance for special offer...	InSync Online	"Jess Hyatt"
Delivering Intel(R) technolog...	Intel Home Computing	pplatte@enron.com
Pentium(R) 4 processor tea...	Intel Home Computing	ddavis@ect.enron....
Power Indices	IntercontinentalExcha...	powerindex@list.int...
Power Indices	IntercontinentalExcha...	powerindex@list.int...

Thunderbird thinks this message is junk.

Subject: Last chance for special off **From:** InSync Online 19/11/01 03:15 PM

Dear THERESA,

Due to overwhelming demand for the Palm OS® v4.1 Upgrade with Mobile Connectivity, we are extending the special offer of 25% off through November 30, 2001. So there's still time to significantly increase the functionality of your Palm™ III, IIIx, IIIxe, IIIc, V or Vx handheld. Step up to the new Palm OS v4.1 through this extended special offer. You'll receive the brand new Palm OS v4.1 **for just $29.95 when you use Promo Code**

Unread: 35281 Total: 43689

Figure 4.4: Screen Capture during Adjudication

email was found. The folder header was the name of the folder the email was found in. In addition to these, every filter's classification and score was added.

The email stream is then imported to Mozilla Thunderbird using popa3d[96]. Using Thunderbird's message filters, every adjudicated message is marked as read as well as its spam status. All G_i and filter disagreements are flagged. In Thunderbird the columns recipient, tag, and flag are added. To adjudicate an email the adjudicator marked it as spam, or read and not spam for ham. Thunderbird is surprising quick at sorting email. Sorting email allows for much greater consistency. For example, by sorting by sender it became clear that some senders only sent spam. Also back when this corpus was collected, spammers would send the same message to everyone so sorting by subject identified these messages quickly and would find senders that could be sorted. Sorting on combined score could identify email highly likely to be spam. This process would continue until all the flagged email were adjudicated. As this point the collection would be sorted by spam and read. The spam would be exported to a spam file. The "read but not spam" messages would be exported to a ham file. The file location could then be easily extracted using grep and the bootstrap gold standard could be updated. A separate adjudication was also updated. This adjudication process was complete for each bootstrap iteration. A screen capture of the G_1 adjudication process can be seen in figure 4.4. During the screen capture the email was sorted by sender showing the sender "InSync On-line" sent multiple advertising messages to multiple users.

	$S \rightarrow H$	$H \rightarrow S$
$G_0 \rightarrow G_1$	89	10
$G_1 \rightarrow G_2$	134	546
$G_2 \rightarrow G_3$	12	89
$G_3 \rightarrow G_4$	0	59
$G_4 \rightarrow G_5$	0	0
$G_0 \rightarrow G_4$	132	601
G_5	$\|H\| = 41207$	$\|S\| = 2482$

Table 4.1: TREC Public Corpus Bootstrap Gold Standard Iterations

Table: 4.1 shows the revisions needed during the bootstrap process. $S \rightarrow H$ is the number of messages revised from spam to ham. $H \rightarrow S$ is the opposite.

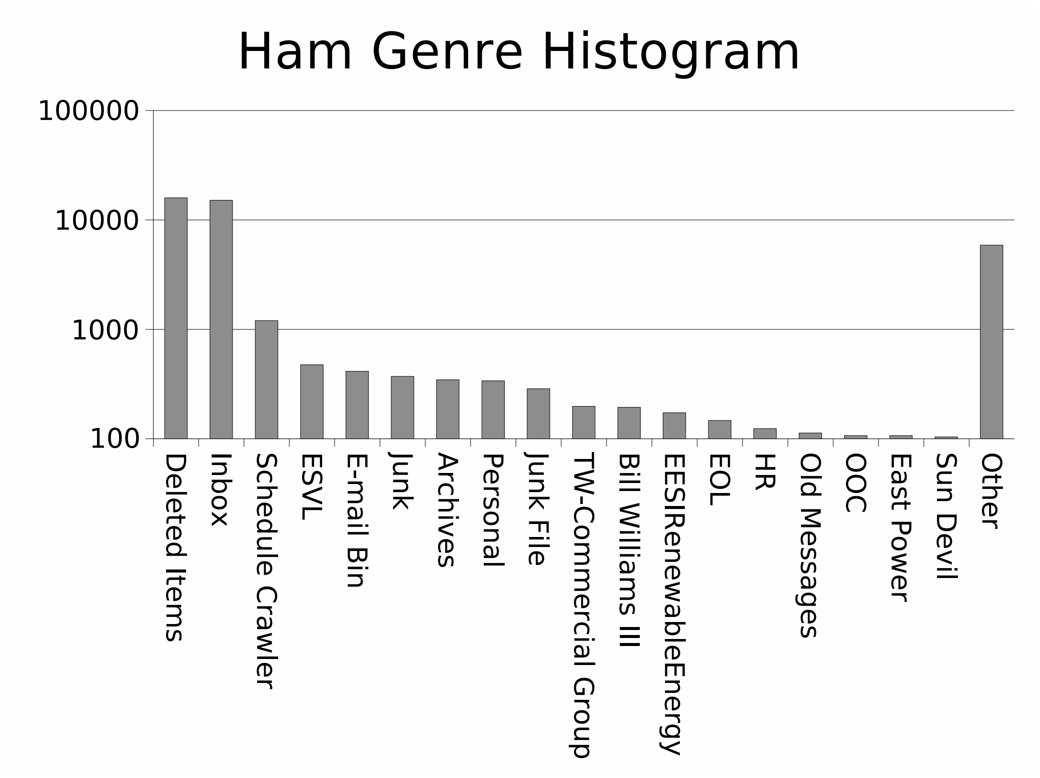


Figure 4.5: Ham Genre Histogram

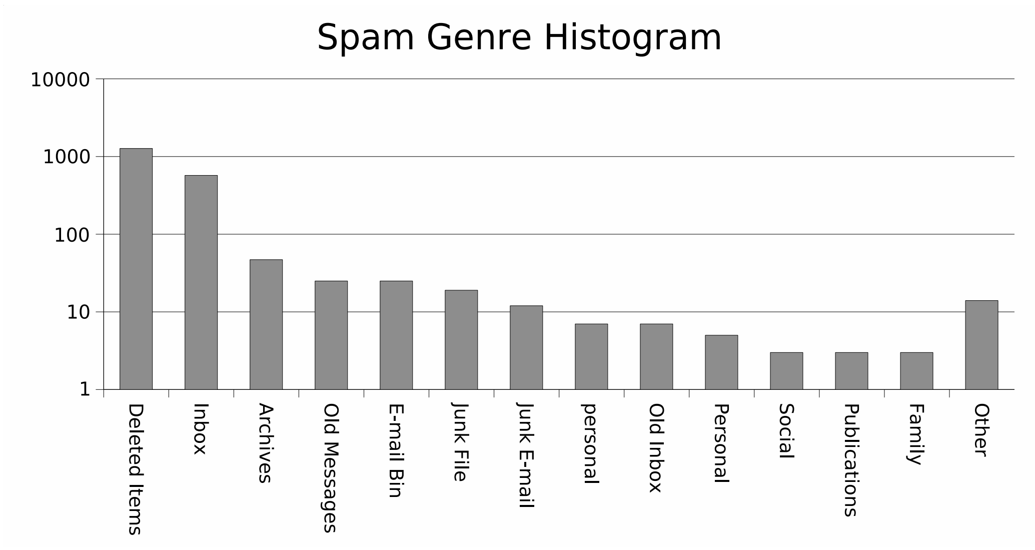


Figure 4.6: Spam Genre Histogram

Figure 4.5 shows a histogram of the folder labels for the ham emails. Figure 4.6 shows a histogram of the folder labels for the spam emails. The folder labels can be used as genre information much like the on-line study discussed in chapter 3. Folder labels were used as extra information in adjudicating the messages. The personal label was quite useful. The junk was not as useful a one might think. Often the junk folder contains joke and non-useful email that would not fit our definition of spam.

4.3 TREC Spam Filtering Track

In 2005, Cormack and I founded and coordinated the TREC Spam Track. Researchers were required to submit filters conforming to the toolkit interface for evaluation on private corpora, as well as to run their filters on the public corpus and submit the results. 12 research groups participated in the 2005 Spam Track, testing a total of 53 filters. Details of groups and filters are located in table 4.2. For comparison, revised versions of the open-source filters supplied with the toolkit were run on the spam track corpora. The authors of three – CRM-114, dbacl, and SpamBayes – were spam track participants. The authors of the remaining five – Bogofilter, DSPAM, Popfile, SpamAssassin, and Spamprobe were approached to suggest revisions or variants of their filters. These versions were tested in the same manner as the participant runs. Table 4.3 illustrates each non-participant filter.

Group	Filter Prefixes
Beijing University of Posts and Telecommunications	kidSPAM1, kidSPAM2, kidSPAM3, kidSPAM4
Chinese Academy of Sciences (ICT)	ICTSPAM1, ICTSPAM2, ICTSPAM3, ICTSPAM4
Dalhousie University	dalSPAM1, dalSPAM2, dalSPAM3, dalSPAM4
IBM Research (Segal)	621SPAM1, 621SPAM2, 621SPAM3
Indiana University	indSPAM1, indSPAM2, indSPAM3, indSPAM4
Jozef Stefan Institute	ijsSPAM1, ijsSPAM2, ijsSPAM3, ijsSPAM4
Laird Breyer	lbSPAM1, lbSPAM2, lbSPAM3, lbSPAM4
Massey University	tamSPAM1, tamSPAM2, tamSPAM3, tamSPAM4
Mitsubishi Electric Research Labs (CRM-114)	crmSPAM1, crmSPAM2, crmSPAM3, crmSPAM4
Pontificia Universidade Catolica Do Rio Grande Do Sul	pucSPAM1, pucSPAM2, pucSPAM3
Universite Paris-Sud	azeSPAM1, azeSPAM2
York University	yorSPAM1, yorSPAM2, yorSPAM3, yorSPAM4

Table 4.2: Participant filters

Filter	Filter Prefix
Bogofilter	Bogofilter
DSPAM	dspam-tum,dspam-toe,dspam-teft
Popfile	Popfile
SpamAssassin	spamasasb, spamasasv, spamasasx
Spamprobe	spamprobe

Table 4.3: Non-participant filters

Four primary corpora – three private and one public, detailed in table 4.4 – were used to evaluate the filters:

	Ham	Spam	Total
Mr X	9038	40048	49086
S B	6231	775	7006
T M	150685	19516	170201
trec05p-1/full	39399	52790	92189
Total	205353	113129	318482

Table 4.4: Corpus Statistics

- **Private Corpus – Mr. X.** The Mr. X corpus was created and defined in the previous chapter. The iterative gold standard process of the corpus is sufficiently accurate; systematic inspection of the 2004 results and of the 2005 spam track results reveals no gold standard errors – any that may persist do not contribute materially to the results.
- **Private Corpus – S. B.** The S. B. corpus consists of 7,006 messages (89% ham, 11% spam) received by an individual in 2005. This individual created the corpus and ran the filters; track organizers had no access to the data. The majority of all ham messages stems from 4 mailing lists (23%, 10%, 9%, and 6% of all ham messages) and private messages received from 3 frequent correspondents (7%, 3%, and 2%, respectively), while the vast majority of the spam messages (80%) are traditional spam: viruses, phishing, pornography, and Viagra ads. Starting from a manual preclassification of all emails, performed when each message arrived in the mailbox, the gold standard was created by running at least one spam filter from each participating group and manually reclassifying all messages for which at least one of the filters disagreed with the preclassification. During this process, 95% of all spam messages and 15% of all ham messages were manually re-adjudicated, and reclassified as necessary. Genre classification was done using a mixture of email header

pattern matching (for mailing lists and newsletters) and manual classification.

- **Private Corpus – T. M.** The T. M. corpus includes personal email, from all accounts owned by an individual, including all mail received (except for spam filtered out by gmail to the gmail address). The individual created the corpus and ran the filters; track organizers had no access to the data. There are 170,201 messages in total. Messages were manually classified as they arrived, and the classifications were verified by running timcv.py over the corpus and manually examining all false positives, false negatives and unsures until there were no more errors. Further verification was effected by running Bogofilter, SpamProbe, SpamBayes and CRM114 (in the TREC setup) over the corpus, manually examining all false positives and false negatives. The corpus ranges from Tue, 30 Apr 2002 to Wed, 6 Apr 2005.
- **Public Corpus – trec05p-1** In addition to the full public corpus, four subsets were defined. These subsets use the same email collection and gold standard judgements, but include only a subset of the index entries so as to reflect different proportions of ham and spam. *trec05p-1/spam50* contains all of the ham and 50% of the spam from the full corpus; *trec05p-1/spam25* contains all of the ham and 25% of the spam. Similarly *trec05p-1/ham50* contains all of the spam and 50% of the ham, while *trec05p-1/ham25* contains all of the spam and 25% of the ham. All subsets were chosen at random. The numbers of ham and spam in each corpus are reported in table: 4.5.

	Ham	Spam	Total
trec05p-1/full	39399	52790	92189
trec05p-1/ham25	9751	52790	62541
trec05p-1/ham50	19586	52790	72376
trec05p-1/spam25	39399	13179	52578
trec05p-1/spam50	39399	26283	65682

Table 4.5: TREC05p-1 Public corpora

The subject filters were run separately on the various corpora. That is, each filter was subject to (up to) nine test runs. The four full corpora – trec05p-1/full, mrx, sb, and tm – provide the primary results for comparison. For each filter, an *aggregate run* was created combining its results on the four corpora as if they were one. The evaluation component of the toolkit was run on the aggregate results, consisting of 318,482 messages in total – 113,129 spam and 205,253 ham. The summary results on the aggregate runs provide a composite view of the performance on all corpora.

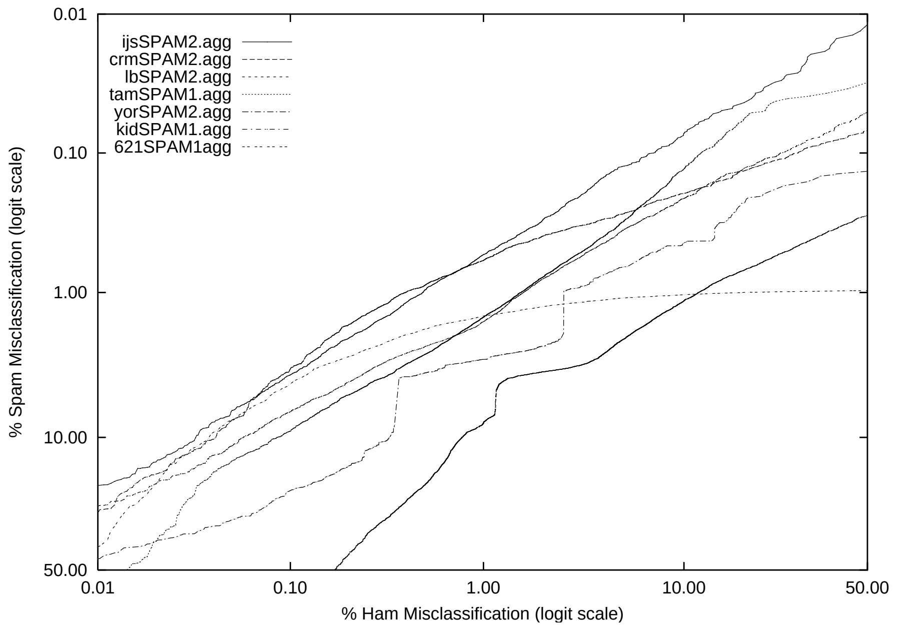


Figure 4.7: TREC ROC Curves(Aggregate)

4.3.1 TREC Spam Filtering Results

Track guidelines prohibited filters from using network resources, and constrained temporary disk storage (1 GB), RAM (1 GB), and run-time (2 sec/message, amortized). These constraints were not rigidly enforced and, in the case of run-time, exceeded by orders of magnitude by some filters. Track guidelines indicated that the largest email sequence would not exceed 100,000 messages. This limit was exceeded as well – the largest consisted of 172,000 messages – but all filters appeared to be able to handle this size, given sufficient time. All but two participant filters – tamSPAM3 and tamSPAM4, which took 22 days and 12 days respectively to process the 49,000-message Mr. X corpus – were not run on all corpora.

Figure: 4.7 shows the ROC curves for the best eight participant runs ranked by (1-ROCA)%, and restricted to one run (the best) per participant. ijsSPAM2 dominates the other curves over most regions. However, if one considers the intercept with the 0.10% ham misclassification line, crmSPAM2 is slightly (but not significantly) higher. This difference is reflected in the different rankings shown in table 4.6. It may be argued that this intercept accurately reflects the usefulness of the filter for its intended purpose. On the other hand, a broad ROC curve may be argued to reflect good filtering performance. Indeed, the crm group indicated that the falloff of the curve was due to a bug they

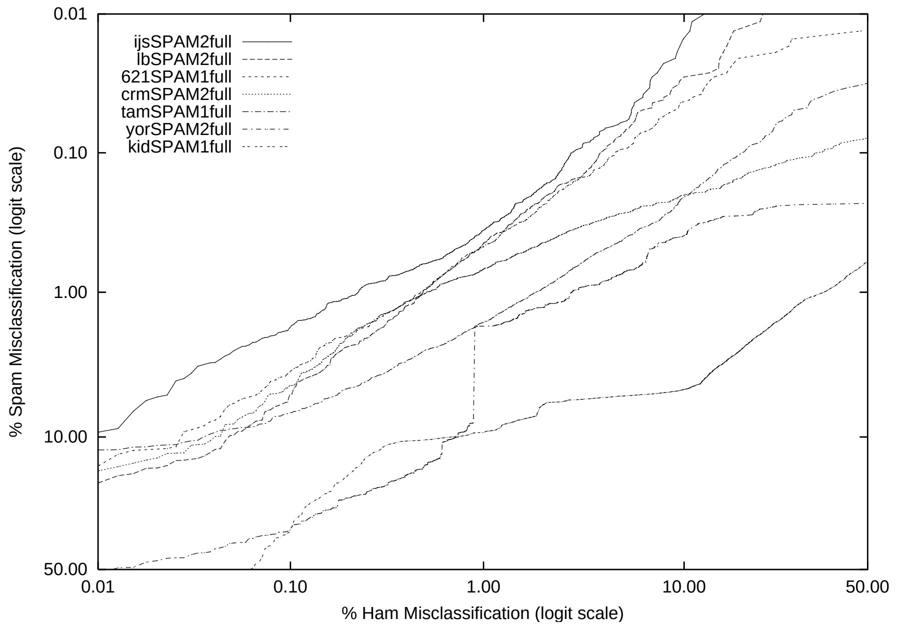


Figure 4.8: TREC ROC Curves(trec05p-1/full)

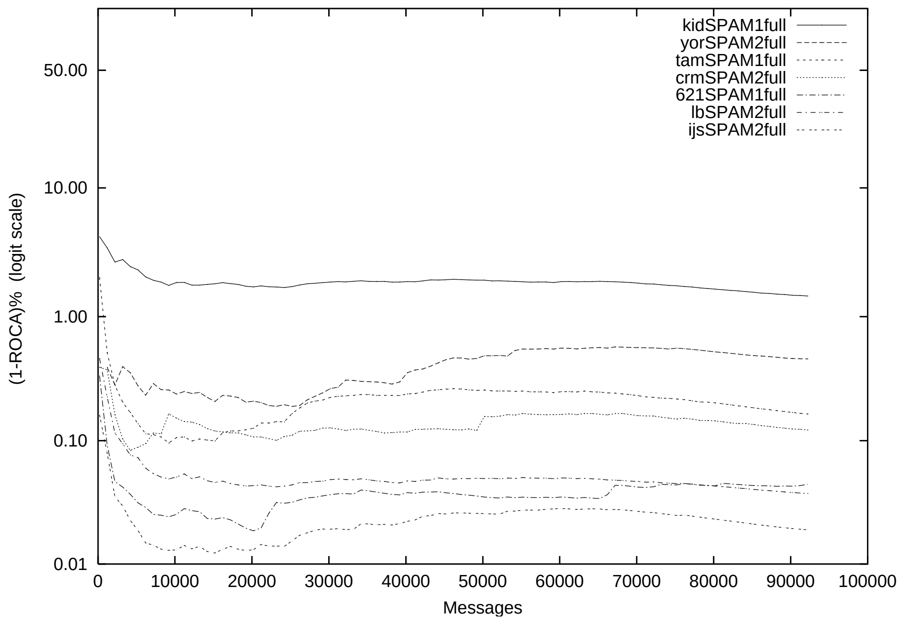


Figure 4.9: TREC Learning Curves(trec05p-1/full)

	ROCA					ROCA Rank				
Filters	Agg	full	Mr.X	S.B.	T.M.	Agg	full	Mr.X	S.B.	T.M.
ijsSPAM2	0.051	0.019	0.069	0.285	0.135	1	1	7	2	1
ijsSPAM1	0.054	0.021	0.069	0.372	0.155	2	2	7	3	2
ijsSPAM4	0.058	0.025	0.063	0.422	0.167	3	4	5	5	5
ijsSPAM3	0.064	0.022	0.050	0.475	0.181	4	3	2	6	6
crmSPAM2	0.115	0.122	0.051	1.888	0.166	5	14	3	16	4
crmSPAM3	0.116	0.042	0.177	0.231	0.195	6	7	14	1	7
crmSPAM4	0.128	0.049	0.218	0.393	0.272	7	10	15	4	8
lbSPAM2	0.132	0.037	0.083	0.835	0.411	8	5	9	9	11
lbSPAM1	0.136	0.039	0.103	0.778	0.443	9	6	12	8	13
tamSPAM1	0.172	0.164	0.138	1.892	0.294	10	16	13	17	9
spamprobe	0.173	0.059	0.097	2.039	0.445	11	11	10	19	14
tamSPAM2	0.209	0.178	0.349	1.127	0.416	12	18	17	11	12
bogofilter	0.210	0.048	0.045	1.426	0.792	13	9	1	13	21
spamasas-b	0.220	0.059	0.097	1.620	0.736	14	11	10	15	19
lbSPAM3	0.262	0.122	0.875	2.727	0.456	15	14	20	23	15
crmSPAM1	0.263	0.169	0.311	2.393	0.790	16	17	16	21	20
lbSPAM4	0.302	0.238	0.492	1.988	0.588	17	19	18	18	17
yorSPAM2	0.316	0.457	0.051	0.983	0.619	18	21	3	10	18
spamasas-x	0.380	0.345	0.065	0.558	1.123	19	20	6	7	23
kidSPAM1	0.768	1.463	1.274	3.553	0.530	20	28	25	29	16
dspam-toe	0.987	0.773	1.109	14.149	2.626	21	23	24	42	25
621SPAM1	1.008	0.044	2.616	2.389	0.161	22	8	35	20	3
621SPAM3	1.090	0.060	2.692	2.604	0.332	23	13	37	22	10
yorSPAM4	1.122	0.688	1.407	58.165	1.081	24	22	26	46	22
dspam-tum	1.274	0.827	0.997	19.384	3.700	25	24	23	43	35
dspam-teft	1.383	0.827	0.942	21.428	4.263	26	24	21	44	36
yorSPAM3	1.491	0.861	1.993	8.234	4.366	27	26	30	37	37
dalSPAM3	1.873	1.491	1.613	2.845	3.090	28	29	27	24	33
yorSPAM1	1.917	2.032	2.632	7.237	4.400	29	32	36	35	38
dalSPAM1	2.097	2.348	2.240	4.614	3.085	30	35	31	30	32
dalSPAM2	2.100	1.674	1.824	3.293	2.898	31	30	28	28	30
kidSPAM4	2.606	3.990	2.326	8.042	2.473	32	38	33	36	24
kidSPAM3	2.741	4.167	2.822	6.360	2.653	33	39	39	33	27
kidSPAM2	3.003	4.544	2.738	7.020	2.749	34	40	38	34	29
ICTSPAM2	3.048	2.643	0.943	3.110	8.298	35	36	22	26	41
dalSPAM4	3.115	1.370	4.282	9.002	6.294	36	27	43	38	40
indSPAM3	3.168	2.822	2.321	12.454	5.843	37	37	32	40	39
pucSPAM0	4.030	2.083	1.910	1.408	2.925	38	33	29	12	31
indSPAM1	4.302	5.346	2.471	13.507	8.382	39	42	34	41	42
pucSPAM1	5.746	2.185	3.081	1.585	2.712	40	34	40	14	28
621SPAM2	6.064	11.362	6.814	3.169	2.647	41	44	45	27	26
pucSPAM2	6.107	1.967	3.454	5.437	3.688	42	31	41	31	34
ICTSPAM1	15.115	4.659	0.748	3.023	34.208	43	41	19	25	43
ICTSPAM3	17.637	20.485	5.328	9.985	36.233	44	45	44	39	44
ICTSPAM4	33.879	10.952	4.114	6.112	42.893	45	43	42	32	46
azeSPAM1	34.079	28.887	34.048	44.502	39.082	46	46	46	45	45

Table 4.6: Summary Results

	trec05p-1/full					trec05-1/full Rank		
Filters	hm%	sm%	ROCA	h=.1	lam	ROCA	h=.1	lam
ijsSPAM2	0.230	0.950	0.019	1.780	0.470	1	1	2
ijsSPAM1	0.250	0.930	0.021	1.840	0.480	2	2	4
ijsSPAM3	0.260	0.970	0.022	1.840	0.510	3	2	5
ijsSPAM4	0.370	0.910	0.025	2.220	0.580	4	5	8
lbSPAM2	0.510	0.930	0.037	5.190	0.690	5	13	11
lbSPAM1	0.410	0.900	0.039	4.560	0.610	6	12	9
crmSPAM3	2.560	0.150	0.042	2.630	0.630	7	7	10
621SPAM1	2.380	0.200	0.044	3.630	0.690	8	10	11
bogofilter	0.010	10.470	0.048	3.410	0.300	9	9	1
crmSPAM4	0.910	0.250	0.049	1.960	0.470	10	4	2
spamasas-b	0.250	1.290	0.059	2.560	0.570	11	6	6
spamprobe	0.150	2.110	0.059	2.770	0.570	11	8	6
621SPAM3	3.140	0.170	0.060	7.020	0.730	13	15	16
crmSPAM2	0.620	0.870	0.122	4.520	0.730	14	11	16
lbSPAM3	0.830	1.050	0.122	22.380	0.940	14	20	18
tamSPAM1	0.260	4.100	0.164	6.920	1.050	16	14	22
crmSPAM1	1.840	1.650	0.169	10.530	1.740	17	18	26
tamSPAM2	0.850	1.450	0.178	27.380	1.110	18	22	23
tamSPAM3	0.220	4.460	0.183	7.640	1.010	19	17	20
lbSPAM4	0.910	3.870	0.238	22.940	1.890	20	21	28
popfile	0.920	1.260	0.325	7.350	0.940	21	16	18
spamasas-x	0.150	3.160	0.345	16.590	0.700	22	19	15
yorSPAM2	0.920	1.740	0.457	34.210	1.270	23	25	25
spamasas-v	0.060	39.510	0.516	31.310	1.870	24	24	27
yorSPAM4	2.990	1.360	0.688	84.920	2.020	25	38	29
dspam-toe	1.040	0.990	0.773	88.760	1.010	26	40	20
dspam-teft	0.260	1.790	0.827	47.090	0.690	27	29	11
dspam-tum	0.260	1.790	0.827	47.090	0.690	27	29	11
yorSPAM3	1.290	1.200	0.861	62.130	1.250	29	34	24
dalSPAM4	2.690	4.500	1.370	76.580	3.490	30	36	35
kidSPAM1	0.910	9.400	1.463	34.930	2.990	31	26	32
dalSPAM3	6.800	6.230	1.491	41.000	6.510	32	27	42
dalSPAM2	5.340	7.520	1.674	41.920	6.340	33	28	41
pucSPAM2	3.350	5.000	1.967	51.280	4.100	34	31	37
yorSPAM1	2.440	2.430	2.032	87.240	2.440	35	39	30
pucSPAM0	3.410	5.100	2.083	59.710	4.180	36	33	38
pucSPAM1	3.570	5.330	2.185	52.580	4.360	37	32	39
dalSPAM1	1.170	21.070	2.348	99.750	5.330	38	47	40
ICTSPAM2	8.330	8.030	2.643	79.510	8.180	39	37	44
indSPAM3	1.090	7.660	2.822	97.350	2.930	40	45	31
kidSPAM4	9.740	6.570	3.990	93.740	8.010	41	44	43
kidSPAM3	0.820	12.490	4.167	90.620	3.330	42	41	34
kidSPAM2	0.870	10.530	4.544	91.650	3.110	43	42	33
ICTSPAM1	5.690	20.850	4.659	72.260	11.190	44	35	46
indSPAM1	0.820	15.160	5.346	93.190	3.700	45	43	36
ICTSPAM4	8.180	24.890	10.952	98.440	14.660	46	46	47
621SPAM2	55.060	1.070	11.362	28.850	10.320	47	23	45
ICTSPAM3	14.100	28.220	20.485	99.390	20.260	48	48	48
azeSPAM1	64.84	4.57	28.887	99.50	22.92	49	49	49

Table 4.7: Public Corpus Summary Results

discovered during the course of their TREC participation.

Figures 4.8 shows the ROC curve (logit scale) and figure 4.9 shows corresponding learning curves for the top filters on the trec05p-1/full corpus. A system is considered superior if its ROC curve is higher than another system at all points of the curve. The curves appear to indicate that the filters have reached steady-state performance. Table: 4.6 shows the filters (1-ROCA)% on the aggregate, trec05-p1/full, Mr.X, S.B., and T.M. corpora. The table also includes the rankings for these corpora. Values of (1-ROCA)% vary a fair amount from one corpus to the next. However the rankings from one corpus to the next don't vary much with only a few exceptions in large changes in rankings.

Table: 4.7 shows the different summary measures to the trec-p1/full corpus. It also includes corpus rankings based on the measures. It is interesting to see that the rankings don't change much for the different measures.

4.4 Discussion

TREC Spam Track evaluation presented here indicates that content-based spam filters can be quite effective, but not a panacea. Misclassification rates are easily observable, even with the smallest corpus of about 8,000 messages. The results call into question a number of public claims both as to the effectiveness and ineffectiveness of "Bayesian" and "statistical" spam filters. The filters did not, in general, appear to be seriously disadvantaged by the lack of an explicit training set. Their error rates converged quickly, and the overall misclassification percentages were not dominated by early errors. In any event, the use of a training set would have been inconsistent with the track objective of modeling real usage as closely as possible. TREC 2005 did not afford the filters on-line access to external resources, such as black lists, name servers, and the like. Participants could have included, but did not, archived versions of these resources with their submissions. No aspect of the toolkit or evaluation measures precludes the use of on-line resources; privacy and repeatability considerations excluded them at TREC. The efficacy of these resources remains an open question, notwithstanding public claims in this regard. The TREC public corpus has been made generally available, subject to a standard TREC usage agreement that proscribes disclosure of information that would compromise its utility as a test corpus.

Chapter 5

Filter Fusion

Fusion methods, under a variety of names[54], have been found to achieve varying degrees of benefit for classification and ranked information retrieval applications. The toolkit's standardized filter interface allows for the easy examination of spam filter fusion methods. It is revealed that a set of independently developed spam filters may be combined in simple ways to provide substantially better filtering than any of the individual filters.

Pilot tests were conducted using the TREC Spam Filter Evaluation Tool Kit on the eight open-source filters used in the on-line supervised spam filtering study. These tests supported the primary hypothesis – that naïve fusion improves on the best base filter. The pilot tests also indicated by exhaustive enumeration that subset selection or different score-combining methods might provide further benefit.

After TREC 2005, tests were conducted using the output from 53 spam filters run on four corpora within the context of the TREC 2005 Spam Evaluation Track. The 53 filters were developed by 17 independent organizations; the four corpora, totaling 318,482 messages, were derived from independent sources. The principal objective was to test the primary hypothesis; a secondary objective was to examine the effectiveness of new fusion and subset selection methods.

The combined results were evaluated, yielding more than a factor of two improvement over the best filter. The simplest method – averaging the binary classifications returned by the individual filters – yields a remarkably good result. A new method – averaging log-odds estimates based on the scores returned by the individual filters – yields a somewhat better result, and provides input to SVM- and logistic-regression-based stacking methods. The stacking methods appear to provide further improvement, but only for very large corpora. Of the stacking methods, logistic regression yields the better result.

Finally, this research demonstrates that it is possible to select prior small subsets of the filters that, when combined, still outperform the best individual filter by a substantial margin.

5.1 Overview Fusion

The variety among spam filter techniques is immense; from rule based and statistical to distributed signature systems and many more. Even techniques with the same basic idea differ greatly. One only has to look at the great differences in the performance among the implementation of Graham's Bayesian filter to see the variety. Some filters combine multiple techniques to improve performance. The process of fusing these techniques together is a technique in itself and there are many ways of achieving this. Another option, the one I chose to take, is fusing multiple filters' results together.

Using multiple filters creates some extra steps over using just one filter. First, the incoming mail is sent to all the filters instead of just one. Next, an extra component is needed to combine the different classifications from multiple filters into one overall classification. There are several methods to combine the classification such as voting, linear regression, SVM. Once the overall classification is complete, the filter process can continue normally as if the overall classification is from a single filter. During training all filters are trained.

5.2 Pilot Fusion Experiment

A pilot study was constructed to determine the viability of fusing multiple spam filter systems. The Spam Filter Evaluation toolkit was utilized to quickly test eight common filters. The baseline would be the best performing filter. Two different fusion methods were employed. The first was a simple voting of the filters. The second made use of the toolkit's requirement that each email be given a spamminess score. Overall classification was based on a naïve normalization of each filter's scores on already seen email. Each email was given an overall spamminess score, the sum of the normalized scores

5.2.1 Pilot Results

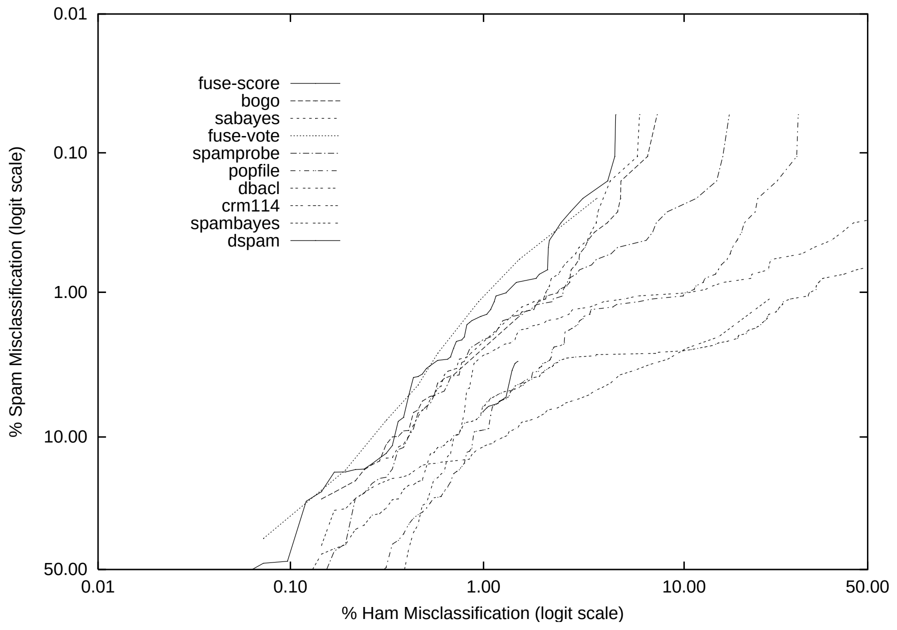


Figure 5.1: Pilot Fusion Filters vs. Base Filters(SpamAssassin Corpus)

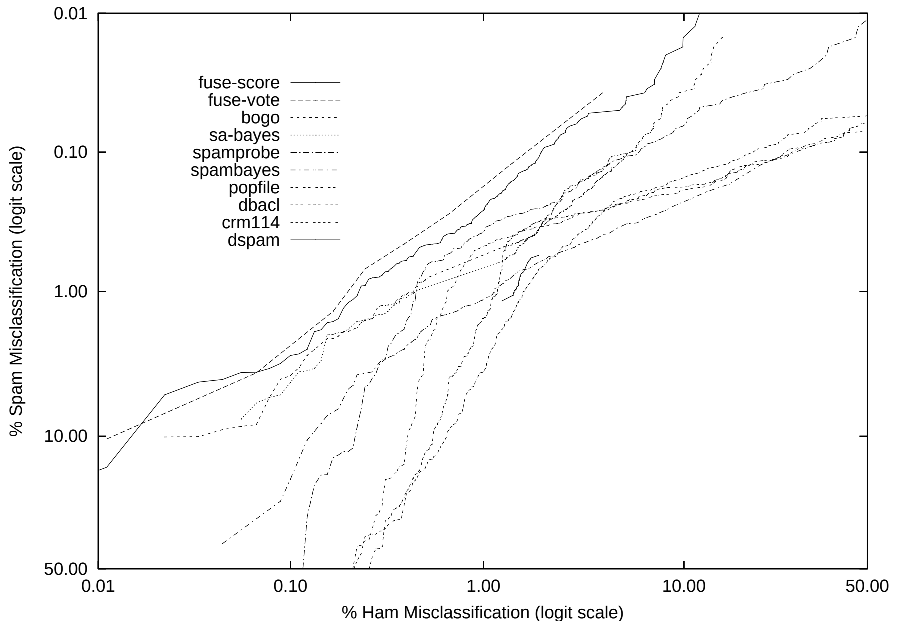


Figure 5.2: Pilot Fusion Filters vs. Base Filters(MrX Corpus)

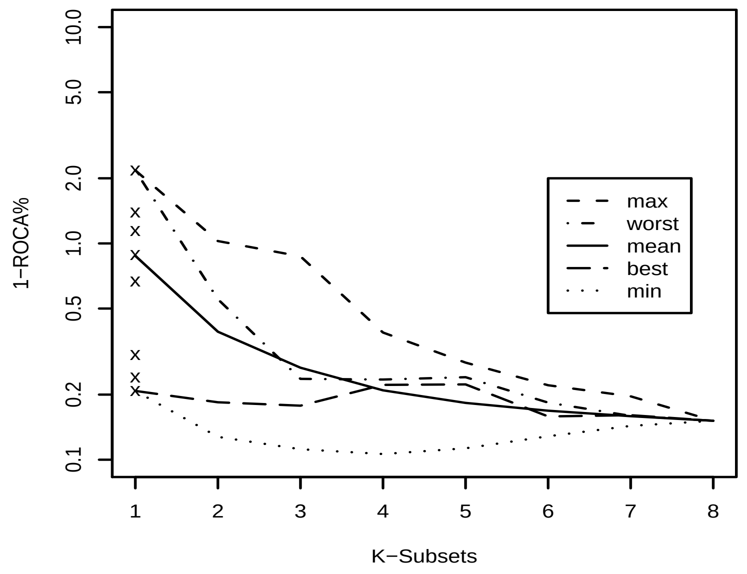


Figure 5.3: Subset Selection(SpamAssassin Corpus)

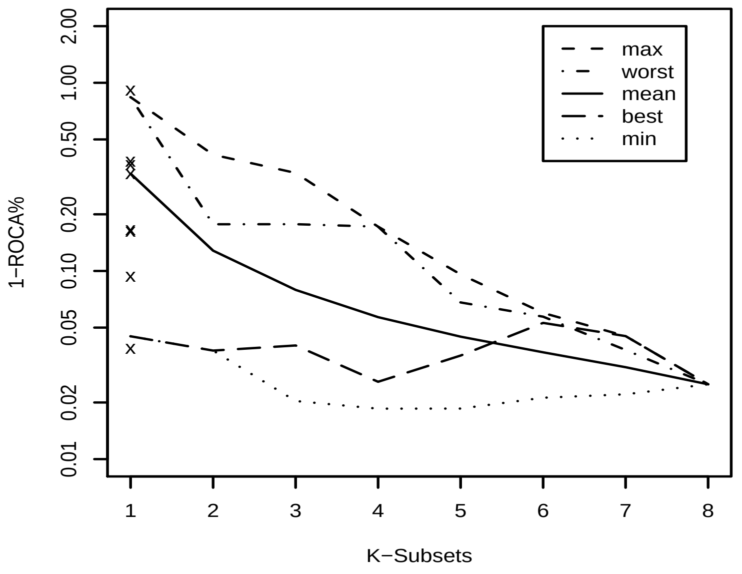


Figure 5.4: Subset Selection(MrX Corpus)

The pilot experiment investigated two naïve fusion methods – voting and normalized score averaging – using eight open-source filters[1] and two test corpora ($n = 6034$[2]; $n = 49086$[3]). Also investigated was the potential impact of subset selection by applying the techniques to all 255 non-empty subsets of base filters.

Figures 5.1 and 5.2 show superior ROC curves for the two fusion methods, as compared to all of the base filters. But only one curve, normalized score averaging on the larger corpus, nets a significantly better (1-ROCA)% statistic ($p < .02$) than the best base filter.

Figures 5.3 and 5.4 shows (1-ROCA)% for normalized averaging over k-subsets of the base runs, as a function of k. The curves labeled *max*, *min*, and *mean* are over the (1-ROCA%) scores yielded by all subsets of size k. The curves labeled *best* and *worst* are yielded by selecting post-hoc the base runs that, taken individually, yield the k best and worst (1-ROCA)% statistics. The x symbols on the 1-axis indicate (1-ROCA)% for each of the base runs.

5.2.2 Pilot Discussion

From the pilots, a conclusion was reached that the naïve combination methods were worthy of further validation. Normalized averaging as a method for combining scores relies on unwarranted assumptions about the distribution of spamminess scores returned by the base filters. Therefore, a method was devised that relied only on the warranted assumption that each filter would attempt to minimize (1-ROCA)%; that is, to minimize the number of pairs of ham and spam messages in which the ham message yielded the higher spamminess score.

K-subset analysis found reason to hypothesize that subsets of the base filters might be found a priori (as opposed to a posteriori in the pilot) that would yield better performance, or that would yield good performance with less computational expense. And if subsets might be learned, so might other linear and non-linear combinations of the scores.

[1]Bogofilter [bogofilter.sourceforge.net],
CRM114 [crm114.sourceforge.net],
dbacl [dbacl.sourceforge.net],
DSPAM [dspam.sourceforge.net],
Popfile [popfile.sourceforge.net],
SpamAssassin (Bayes filter only) [spamassassin.apache.org],
SpamBayes [spambayes.sourceforge.net],
SpamProbe [spamprobe.sourceforge.net].

[2]SA Corpus [spamassassin.apache.org/publiccorpus].

[3]Mr X Corpus [plg.uwaterloo.ca/˜gvcormac/mrx].

5.3 Fusion methods

Best Filter

As a baseline for comparison, the filter achieving the best ROC score on each corpus was selected (a posteriori).

Voting

Each base filter's output consists of a binary classification and a spamminess score. Vote fusion uses only the binary classification output of the base filters. The fused filter's spamminess score for a message is the fraction of base filters that classify it as spam – a number between 0 and 1. The fused filter's binary classification is determined relative to some arbitrary constant threshold $0 < t < 1$; a spam classification is returned when $spamminess > t$. The summary statistics presented are insensitive to our choice of $t = 0.5$.

Log-odds averaging

When a filter reports a spamminess score s_n for the nth message, the estimate L_n, is the odds that the message is spam to be

$$L_n = log \frac{\mid \{i < n \mid s_i \leq s_n \, and \, ith \, message \, is \, spam\} \mid + \epsilon}{\mid \{i < n \mid s_i \geq s_n \, and \, ith \, message \, is \, ham\} \mid + \epsilon} \; .$$

That is, simply count the number of prior spam messages with a lower or equal score and the number of prior non-spam messages with a higher or equal score, and take the log of their ratio. The necessary counting can be done in $O(log\, n)$ time with a suitable data structure [15]. The fused spamminess score is the arithmetic mean of the base filters' L_n scores. With $t = 0$.

SVM

L_i scores were used as features and all prior messages were used as a training set. SVMlight's [67] default kernel and parameters were used. For efficiency reasons, SVMlight was not run after every message; retraining was effected at exponential-like intervals.[4] The SVMlight output was used directly as the fused spamminess score. With $t = 0$.

[4]Increasing training set sizes were used to adapt SVM, a batch method, to on-line classification [35]. Training set sizes of *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000 are used.*

Logistic regression

The LR-TRIRLS logistic regression package [76] was used to find weights such that the weighted average of the base filters' L_i scores best predicted the log-odds of the classification of prior messages. This weighted average was used as the spamminess score, and $t = 0$. Negative weights were assumed to represent over-fitting; an iterative process was used to eliminate them. The filter with the most negative weight was eliminated; regression and elimination were repeated until no negative weights remained. For efficiency reasons, the weights were not recomputed for every message. For the first 100 messages, the weights were fixed at $\frac{1}{f}$, where f is the number of base filters. Thereafter, they were recomputed after every n_j messages where n_1, n_2, n_3, ... forms a exponential-like series.[5]

5.4 Fusion Results

Figures 5.5, 5.6, 5.7 and 5.8 show the ROC curves for the four fusion methods and the best filter for each of the four corpora. Table 5.1 shows the summary statistics for the same runs, with 95% confidence limits and p-values. Each p-value indicates the probability that the statistic's improvement over that of the best filter may be due to chance.

All fusion methods substantially outperformed the best filter. The lack of significance of results with respect to the S.B. corpus may be attributed to its size; 775 spam messages are insufficient to distinguish filters at the error rates achieved. It may also be the case that some effects (notably SVM and logistic-regression stacking) increase with corpus size. Voting – simply counting the binary classification outputs of the filters – is remarkably effective, but appears to yield somewhat less improvement than the other filters. On the other hand, there is reason to believe that voting is more stable, and may perform better on short corpora, or on the first several thousand messages of long corpora. One possible reason for this is that voting is better able to take advantage of prior knowledge incorporated into the individual filters; until reliable estimates of the filter's credibility are obtained, simple voting seems to be the safest choice. Nevertheless, given the diversity of performance among the base filters, it is remarkable that a simple vote works so well. Each filter, no doubt, incorporates several arbitrary parameters set by its authors, not the least important of which is t, the classification threshold. Thus, voting works well due to social behaviour as much as any technical reason.

[5]Increasing training set sizes of *0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 2100, 4100, 9100, 19100, 39100, 69100, 99100, 129100, 159100 is used*

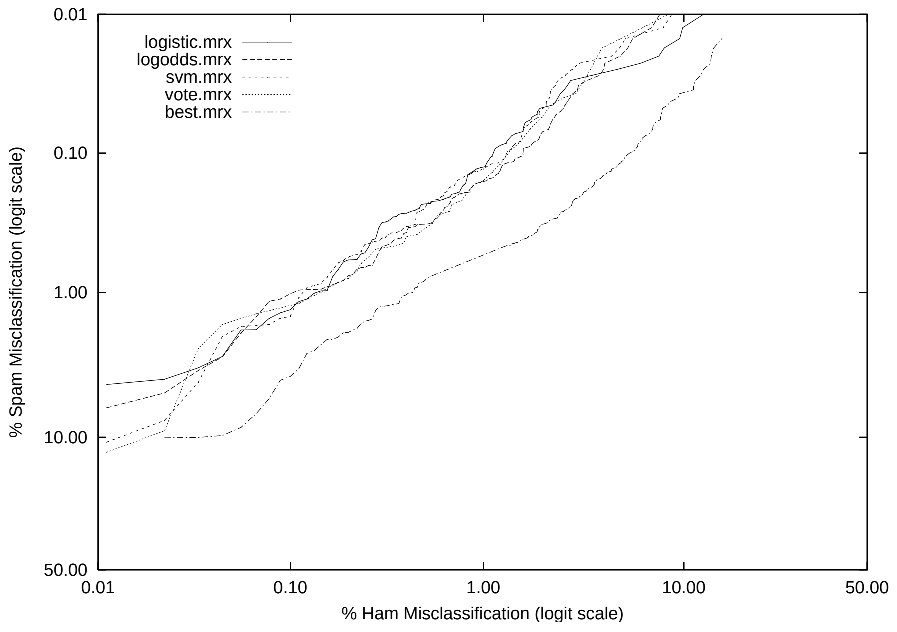


Figure 5.5: Fusion Filters vs. Best Filter(MrX Corpus)

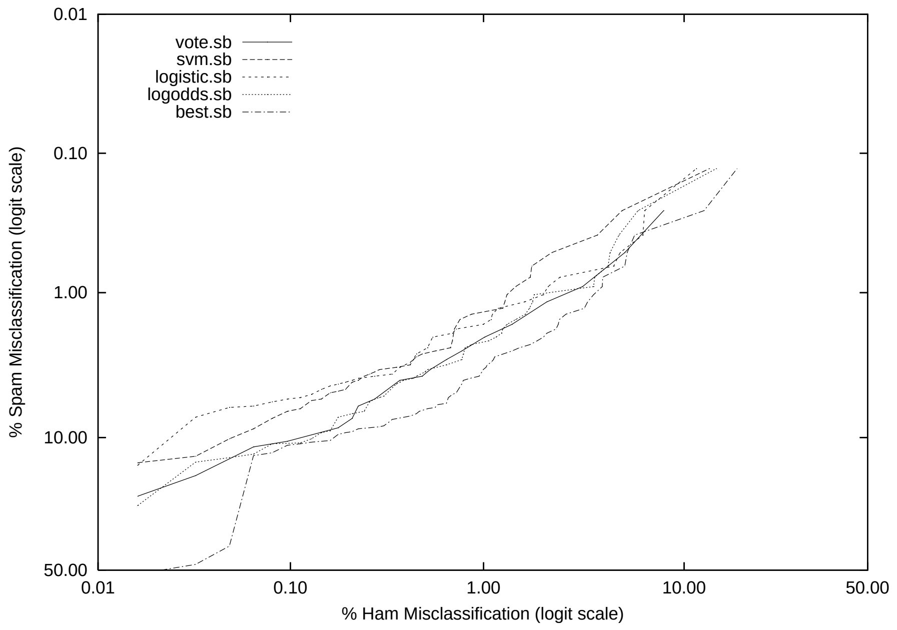


Figure 5.6: Fusion Filters vs. Best Filter(SB Corpus)

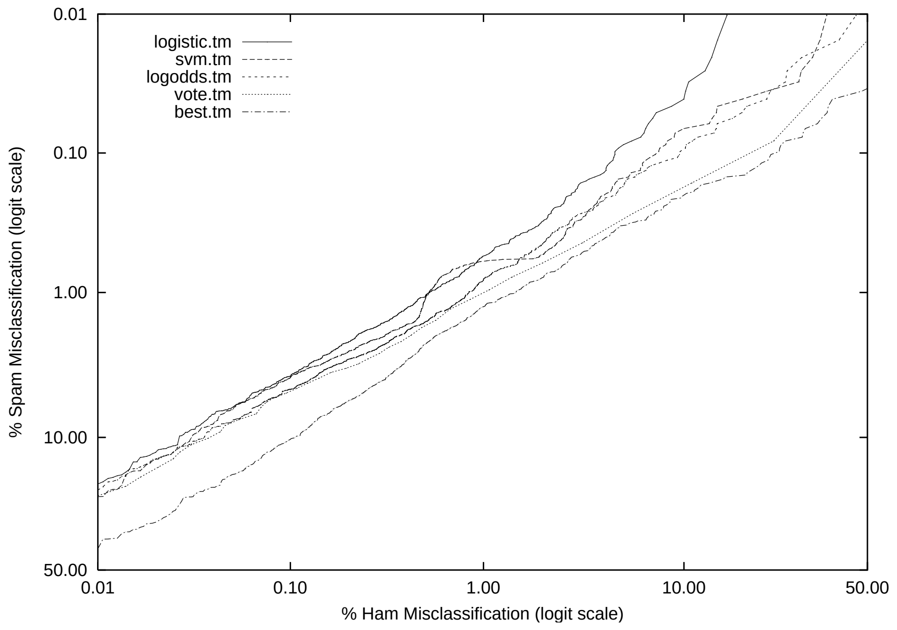


Figure 5.7: Fusion Filters vs. Best Filter(TM Corpus

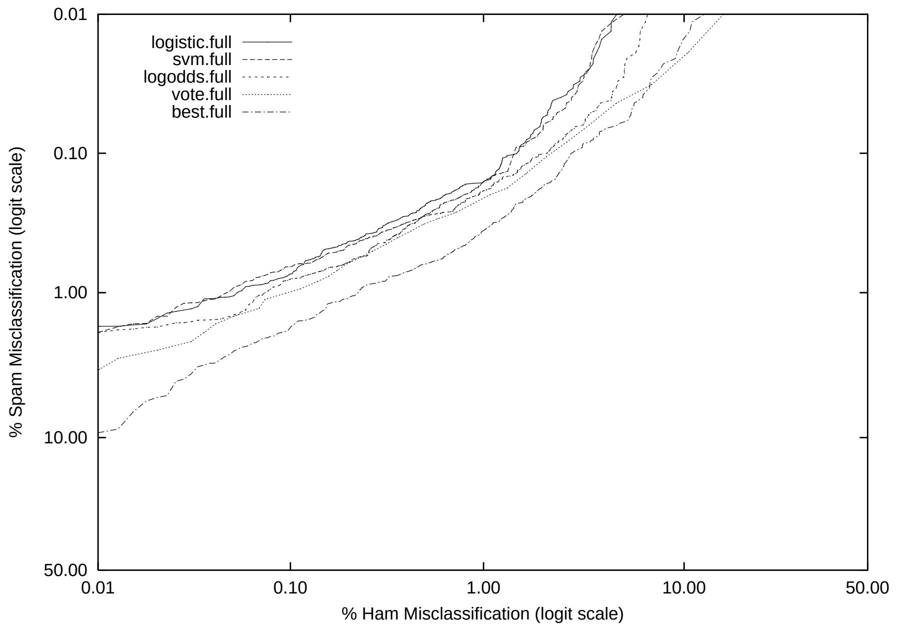


Figure 5.8: Fusion Filters vs. Best Filter(Full Corpus)

Method	$(1 - ROCA)\%$	$sm\%@hm\% = .1$
logistic	.007*** (.005-.008)	.73*** (.55-.98)
svm	.008*** (.005-.013)	.65*** (.55-.77)
logodds	.009*** (.007-.011)	.80*** (.65-.98)
vote	.013* (.010-.018)	1.00*** (.82-1.21)
best	.019 (.015-.023)	1.78 (1.42-2.22)

Full Corpus

Method	$(1 - ROCA)\%$	$sm\%@hm\% = .1$
logistic	.010*** (.007-.014)	1.32* (.68-2.58)
logodds	.011*** (.007-.016)	1.02** (.53-1.97)
svm	.011*** (.007-.017)	1.48* (.73-2.98)
vote	.014*** (.008-.024)	1.21** (.86-1.71)
best	.045 (.032-.063)	3.90 (1.55-9.50)

Mr X Corpus

Method	$(1 - ROCA)\%$	$sm\%@hm\% = .1$
vote	.115** (.071-.184)	10.5 (6.75-15.8)
svm	.155 (.046-.516)	6.71 (3.66-12.0)
logistic	.166 (.057-.483)	5.55 (3.57-8.53)
logodds	.193 (.076-.490)	11.0 (7.01-16.8)
best	.231 (.142-.377)	11.2 (4.38-25.9)

S B Corpus

Method	$(1 - ROCA)\%$	$sm\%@hm\% = .1$
logistic	.036*** (.030-.044)	3.89*** (3.43-4.41)
svm	.055*** (.045-.067)	3.97*** (3.50-4.49)
logodds	.061*** (.045-.067)	4.78*** (4.27-5.33)
vote	.095** (.079-.115)	4.91*** (4.45-5.43)
best	.135 (.111-.163)	10.3 (9.16-11.6)

T M Corpus

Method	$(1 - ROCA)\%$	$sm\%@hm\% = .1$
logistic	.012*** (.010-.015)	1.20*** (1.07-1.35)
svm	.017*** (.015-.021)	1.29*** (1.16-1.45)
logodds	.020*** (.017-.023)	1.78*** (1.64-1.93)
vote	.028*** (.023-.033)	1.66*** (1.48-1.86)
best	.051 (.044-.058)	3.78 (3.36-4.25)

Aggregate Results

improvement on best: $*p < .05$, $**p < .005$, $***p < .0005$

Table 5.1: Fusion Summary Statistics

The log-odds transformation is an essential component of the other techniques – the transformed scores were used directly and also as input to the SVM and logistic regression meta-learning methods. In the pilot experiment, various linear and non-linear combinations of scores were investigated. Although the sum of linear-normalized scores worked acceptably well in the pilot, there was no confidence that it would combine well the diverse score distributions found in the TREC runs. Indeed it did not, performing more poorly than simple voting on the Mr X Corpus. Therefore it was dropped from further consideration and was not tested on the other corpora. Since Mr X was used in the pilot (but with different filters), it was used for testing various parameters and methods, testing only the ones that appeared promising – the ones reported here – on the other corpora. In this sense one may consider the Mr X results to be somewhat "cherry picked" but not the results on the other corpora.

The rationale for the log-odds transformation is as follows. Given a threshold t, messages may be placed in two dichotomous classes: spam messages with spamminess score $s \leq t$, and non-spam messages with $s \geq t$. A new message with spamminess t must necessarily fall into one of these classes. The observed size of these classes was used as an estimate of the odds ratio. That is, the area of the tails of the unnormalized score distribution provides a likelihood ratio multiplied by the prior odds (i.e. the overall odds ratio). Using log-likelihood instead of log-odds was also experimented. Log-likelihood is computed by subtracting log-prior-odds from log-odds; log-prior-odds is easily estimated from the observed spam to non-spam ratio. While log-likelihood makes more "sense" from a probabilistic point of view, it makes no difference to ROC or logistic regression results, and introduces slightly more noise due to the (additional) instability of the log-prior-odds estimate. In addition, positive or negative log likelihood ratios [10] were computed (as appropriate) from the base filters' binary classifications; preliminary testing revealed the average of these works marginally better than voting, but not as well as the average of the log-odds-transformed scores.

Three of the corpora showed better results for log-odds averaging than for voting; two were significant in a 2-tailed test (full, $p < .0002$; mrx, $p < .2$; tm, $p < .0001$), one showed an inferior (sb, $p < .16$) result which is largely due to chance, but may also be due to the small size of the corpus offering insufficient numbers for accurate log-odds estimates. The aggregate "run", which is not a run at all but an amalgam of the other four, shows that log-odds averaging improves on voting ($p < .0001$).

The log-odds transformed scores were used as input features to SVMlight. The untransformed scores and the binary classifications as features were also tried with deleterious results. We also tried several combinations of kernels and parameter settings, but found none that yielded better results. We do not claim to have the exhausted space of features, kernels and settings. SVMlight, using default parameters, improves on voting on the same corpora as does log-odds, and shows a significant improvement in the aggregate ($p < .0001$). While SVM's improvement over log-odds is significant only for the aggregate run ($p < .01$), the consistent improvement over the four corpora leads us to believe that it is better.

Straightforward logistic regression yielded poor performance, even with very large amounts of training data. We observed, as did Hull [66] in a somewhat different context, that negative coefficients were a near-certain sign of over-fitting.[6] But logistic regression constrained to non-negative results is intractable, so we used the simple heuristic of deleting the filter with the most negative coefficient and repeating until no negative coefficients remained. There is no reason to believe that this is the best approach. For example, we could have used significance rather than magnitude as an elimination criterion. But for

[6]We say near-certain because the process did, in fact, discover some valid negative coefficients. Two of the base filters were fusions of other filters, and the regression process yielded a strong negative coefficient for components that were overrepresented.

efficiency we chose a simplistic technique that appeared to work. We leave it to future research to investigate more sophisticated strategies.

Logistic regression performed the best on all corpora except S. B.; significantly better than the other methods in the aggregate (vote, $p < .0001$; logodds, $p < .0001$; svm, $p < .0001$). S. B.'s discordant result is not significant and may be due to chance. Examination of the ROC curve (figure 5.1) shows the logistic regression curve apparently superior to the rest, yet the (1-ROCA)% statistic is inferior. Further investigation, and verification of the ROC results with SPSS, shows that an extreme point beyond the scale of the graph accounts for the difference. We note also that $sm\%$ at $hm\% = .1$ shows logistic regression to be superior on the S. B. corpus. While the difference may be due to chance, it is also plausible that stacking methods are superior only on larger corpora, where they have more opportunity to learn.

5.5 Predicting Subset Experiment

To select subsets of the base filters, we employed the same elimination process as logistic-regression stacking. After eliminating the filters corresponding to negative weights, we continued the process – eliminating the filter with the smallest weight – until only k filters remained. These k filters formed the base classifiers for a new fused filter. The resulting filter combines k spamminess scores by multiplying them by their respective weights as determined by the selection process. The subset experiment, unlike fusion, involved a batch process – selection and the computation of weights takes place with respect to a training corpus and the resulting filter is applied to a different test corpus.

To evaluate the subset selection method, we used two corpora – Mr X and S B – as training corpora, and the other two – Full and T M – as test corpora. For each test corpus we computed subsets of size 2, 3, 4, 8, 16, ..., m where m is the largest subset that yields all positive coefficients. Each subset was used in a fusion run on the two test corpora. Tables 5.2 and 5.3 show the results of these four sets of runs. All subsets improve on the best run in both measures, significantly so except for the smaller subsets trained on the S.B. corpus. Performance improves with subset size; performance of the larger subsets is comparable to that of the better fusion methods.

Subset	$(1-ROCA)\%$	$sm\%@hm\% = .1$	Subset	$(1-ROCA)\%$	$sm\%@hm\% = .1$
sb14	.008*** (.007-.010)	1.01*** (.81-1.25)	sb14	.049*** (.041-.059)	5.50*** (4.83-6.27)
sb8	.008*** (.007-.010)	1.02*** (.81-1.28)	sb8	.053*** (.044-.063)	5.78*** (5.01-6.66)
sb4	.010*** (.008-.012)	1.40* (1.07-1.82)	sb4	.058*** (.048-.069)	6.09*** (5.21-7.11)
sb3	.012*** (.010-.015)	1.45* (1.22-1.73)	sb3	.074*** (.061-.089)	7.72*** (6.60-9.00)
sb2	.015*** (.012-.018)	1.51 (1.23-1.84)	sb2	.109** (.087-.136)	8.80*** (7.58-10.18)
best	.019 (.015-.023)	1.78 (1.42-2.22)	best	.135 (.111-.163)	10.3 (9.16-11.6)
	Full Corpus			**T M Corpus**	

improvement on best: *$p < .05$, **$p < .005$, ***$p < .0005$

Table 5.2: SB-derived Subsets on Full and TM Corpora

Subset	$(1-ROCA)\%$	$sm\%@hm\% = .1$	Subset	$(1-ROCA)\%$	$sm\%@hm\% = .1$
mrx23	.007*** (.006-.009)	.79*** (.62-.99)	mrx23	.047*** (.038-.057)	3.84*** (3.41-4.32)
mrx16	.007*** (.006-.009)	.84*** (.69-1.02)	mrx16	.050*** (.040-.062)	3.99*** (3.56-4.48)
mrx8	.009*** (.007-.011)	.88*** (.71-1.08)	mrx8	.055*** (.041-.072)	4.22*** (3.72-4.79)
mrx4	.012*** (.009-.015)	1.07*** (.82-1.39)	mrx4	.084*** (.067-.105)	4.37*** (3.74-5.09)
mrx3	.012*** (.010-.016)	1.15*** (.92-1.44)	mrx3	.081*** (.063-.104)	4.20*** (3.66-4.81)
mrx2	.016 (.012-.021)	1.31** (1.01-1.68)	mrx2	.094*** (.075-.118)	4.40*** (3.90-4.96)
best	.019 (.015-.023)	1.78 (1.42-2.22)	best	.135 (.111-.163)	10.3 (9.16-11.6)
	Full Corpus			**T M Corpus**	

improvement on best: *$p < .05$, **$p < .005$, ***$p < .0005$

Table 5.3: Mr X-derived Subsets on Full and TM Corpora

The stepwise elimination process embodied in the logistic regression approach identifies a subset of the base filters that contribute to the best fusion result. Continuing the elimination process yields smaller subsets which all outperform the best filter; even the subsets of size 2 outperform the best individual filter. Figure 5.9 indicates the number of distinct Mr X-derived subsets in which each filter participates; the filters are labeled and ordered by their individual performances. We note that the best-performing filter is not a member of any of the subsets – many strong filters are excluded in favour of weaker ones. The S B derived subsets show the same effect, from which we may infer that inter-filter correlation is a determining factor in subset selection.

The cross-corpus design of the experiments serves to indicate that a subset of filters chosen using one source of email may be expected to yield a fused filter that works well on another.

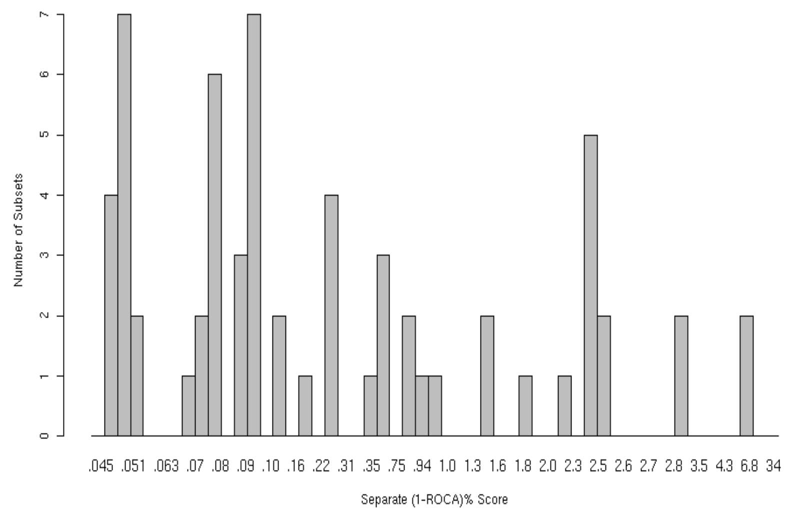


Figure 5.9: Base Filter Participation in Subsets (by Separate Performance)

5.6 Discussion

The potential contribution of machine learning techniques to real spam filtering is as-yet unresolved. In artificial environments they appear to be promising, but this promise is yet to be demonstrated in comparison to existing filters[35]. The potential contribution of real-time network resources and collaborative methods to spam filtering also has yet to be established. Spam filtering is an adversarial task – the degree to which spam is able to adapt to counter advances in filtering has yet to be studied. While constructing controlled experiments to measure these factors presents a significant logistical challenge, our model and evaluation methods are amenable.

The simplest fusion method – voting based on the binary classifications yielded by the individual filters – yields an ROC curve that is clearly superior to the best filter on each of the corpora. Although voting works well, it lacks appeal because it relies on the arbitrarily-set classification thresholds of the individual filters, and its sensitivity can be adjusted only coarsely by specifying the number of filters that must agree to classify a message as spam. The 53 different threshold values afforded by this test were adequate to achieve good ROC results, but we are skeptical as to whether the approach would be practical for a smaller number of filters, unless one had the capability to adjust the individual filters' thresholds.

The score-based fusion methods – log-odds averaging, SVM, and logistic regression

– are more appealing in that they use the score and not the threshold setting from each individual filter. The score-based methods also appear to improve on voting, but the incremental improvement is not nearly as dramatic as that of voting over the best individual filter. The ROC curves for these methods don't clearly dominate voting, and the statistics are superior by a significant margin on only the larger corpora. Of these methods, logistic regression (with elimination of filters with negative coefficients) appears to yield the best performance. On the other hand, log-odds averaging is the simplest of the score-based methods, and the other methods take as input the log-odds transformed scores. That is, the log-odds transformation is the essential basis of all the score-based methods. Because SVM and logistic regression are batch oriented, increasing exponential-like training sizes were required to make the effort tolerable while still being reasonably adaptive. After this work was completed; on-line methods for both logistic regression and SVM have been developed, and they work as well as the batch methods we employed.

In practice, it may not be feasible to run 53 separate filters on each incoming email message. Our experiments indicate that it is possible to select a smaller number – roughly half – without compromising performance. Smaller subsets – perhaps only a handful of filters – compromise performance only slightly. Furthermore, it appears that these subsets may be picked prior, based on a training corpus derived from a distinct source of email.

Fusion experiments may be repeated using the public corpora and the open-source filters. The 53 filters tested at TREC include the best available filters at the time as well as several experimental and less-well-performing filters. We advanced the hypothesis that as new filters were developed and tested, they too would perform best in combination with other independently-developed filters. This hypothesis is supported by later experiments detailed in section 6.4.

Chapter 6

Related Work

This chapter surveys

- reported results of studies evaluating spam filter effectiveness, recast where possible in terms of hm, sm and $ROCA$;
- datasets, tools, and evaluation methodologies;
- progress in spam filtering catalyzed by this work.

The primary focus of this chapter is the evaluation of on-line content filters; that is, filters that use only the content of the messages, along with feedback derived from the user in the course of reading email. Within this context we consider also batch-oriented content filters, but not system-wide approaches.

6.1 Recasting Previous Studies with Standard Measures

Direct comparison of results demands that common data (or at least data sampled from a similar population) be used to test different filters, or that common filters be tested on different data, and that common measures be reported. Valid measurements must be based on realistic assumptions and statistically well founded. With these criteria in mind, we explore the commonality among studies, and, where possible from the published data, recast their results in terms of common measures with confidence intervals.

Sahami et al. [104] conducted an early study that indicated the utility of Bayesian classifiers for spam filtering. One experiment used a corpus of 1789 actual email messages (11.8% ham; 88.2% spam), split chronologically into 1538 training messages and

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

Table 6.1: Sahami et al. Results

251 test messages. Both ham and spam precision/recall curves were calculated. The best-performing system achieved ham recall of 100% and spam recall of 98.3%. From these values and the test sample size we may compute $hm = 0\% \,(0\% - 9.5\%)$ and $sm = 1.7\% \,(0.4\% - 4.6\%)$. A second experiment classified the spam component of a similar corpus into two genres: *pornographic* and *non-pornographic*. The genres were used in an evaluation of ternary classification, but not for a stratified evaluation of binary classification. A third experiment most closely resembles those which we conducted: an individual's email messages were captured over one year, classified manually, and used as training data. The filter was applied to further week's email received by the same individual. The resulting classification table, shown in table 6.1, demonstrates $hm = 1.7\% \,(0.3\% - 4.9\%)$, $sm = 20\% \,(9.6\% - 34.6\%)$. Sahami et al. further examine the three misclassified ham messages, observing two to be newsletter messages and one to be a personal message that includes a spam message as an attachment. The test corpus is unavailable for comparative evaluation.

Ling Spam[6] is an abstraction of 2412 ham messages from a mailing list and 481 spam messages from an individual recipient. We say abstraction because the messages are stripped of headers and line breaks, converted to lower case, tokenized and stemmed. The filters tested on the Ling Spam corpus were purpose-built to use it to evaluate specific machine-learning techniques. Although the results are reported in terms of spam recall, spam precision and weighted accuracy, it is possible to reconstruct the contingency table from these results. Table 6.2, for example, recasts the results of Androutsopoulos et al. [6] in terms of hm and sm.

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

Table 6.2: Androutsopoulos et al. Results

Filter	% Ham Misc.	% Spam Misc.	% Misc.
SpamAssassin	0	95.4	15.9
Bogofilter	0	59.5	9.9
SpamProbe	0	31.3	5.2
CRM114	54.9	11.2	18.5

Table 6.3: Real Filter Results on Ling Spam corpus

Ling Spam is freely available and has been used in many studies [105, 138, 6, 8]. We found that real spam filters were, in general, unable to classify the Ling Spam messages (see table 6.3); we are unaware of how to modify either the corpus or the filters so as to use them together in a valid experiment.

Androutsopoulos et al.[7] define four public corpora – PU1, PU2, PU3 and PUA – with a total of 5957 messages (3508 ham and 2449 spam); each corpus abstracts and also obfuscates email from one recipient, so as to preserve privacy. In addition, repeated ham messages and spam messages from regular correspondents – about half the spam and eighty percent of the ham – are discarded in forming the corpus. As for Ling Spam, experiments using these corpora depend on purpose-built filters. One such filter – Filtron – was trained on PU3 and tested on real email received by an individual over seven months. During this interval, 5109 ham and 1623 spam messages were received and classified. Table 6.4 summarizes the results. Neither Filtron nor the real email corpus is available for comparative study.

Corpus	Contingency table		% Ham Misc.	% Spam Misc.	% Overall Misc.
PU3	2249	90	2.8 (2.1-3.5)	4.9 (4.0-4.9)	3.7 (3.2-4.3)
	64	1736			
Real Email	5057	173	1.0 (0.8-1.3)	10.7 (9.2-12.3)	3.8 (3.3-4.3)
	52	1450			

Table 6.4: Filtron Results

The public *SpamAssassin* corpus[120] consists of 6034 messages – 4149 ham and 1885 spam – gathered from various sources at various times. Although it is not a chronological sequence of messages delivered to a single recipient, the messages contain original headers with minor elision for the sake of privacy. Holden[64] used the SpamAssassin corpus and ten-fold cross-validation to test fourteen open-source filters, including versions of the six tested here. Holden's results are summarized in table 6.5. Holden further tested the filter on one-month's personal email, again using cross-validation; results are shown in table 6.6. Holden provides a qualitative description of the misclassified ham messages,

Filter	Contingency table		% Ham Misc.	% Spam Misc.	% Overall Misc.
Annoyance	4147	209	0.07 (0.01-0.21)	11.0 (9.6-12.5)	3.5 (3.1-4.0)
	3	1688			
Antispam	4016	35	3.2 (2.7-3.8)	1.8 (1.3-2.6)	2.8 (2.4-3.2)
	133	1863			
Bayesspam	4033	77	2.9 (2.3-3.4)	4.1 (3.2-5.0)	3.2 (2.8-3.7)
	117	1863			
bmf	4137	78	0.3 (0.2-0.5)	4.1 (3.3-5.1)	1.5 (1.2-1.8)
	18	1819			
Bogofilter	4145	184	0.12 (0.04-0.28)	9.7 (8.4-11.1)	3.1 (2.7-3.6)
	5	1713			
CRM114	4100	58	1.2 (0.9-1.6)	3.1 (2.3-3.9)	1.8 (1.5-2.2)
	50	1839			
dbacl	309	22	92.5 (91.7-93.3)	1.2 (.73-1.75)	63.9 (62.7-65.1)
	3841	1875			
DSPAM	4137	76	0.3 (0.17-0.5)	4.0 (3.2-5.0)	1.5 (1.2-1.8)
	13	1821			
lfile	4087	129	1.5 (1.2-1.9)	6.8 (5.7-8.0)	3.2 (2.7-3.6)
	63	1768			
qsf	4134	160	0.39 (0.22-0.63)	8.4 (7.2-9.8)	2.9 (2.5-3.4)
	16	1737			
SpamAssassin	4144	74	0.14 (0.05-0.31)	3.9 (3.1-4.9)	1.3 (1.1-1.6)
	6	1823			
SpamBayes	4143	83	0.17 (0.07-0.34)	4.4 (3.5-5.4)	1.5 (1.2-1.8)
	7	1814			
SpamOracle	4150	309	0 (0-0.07)	16.3 (14.6-18.0)	5.1 (4.6-5.7)
	0	1588			
SpamProbe	4144	65	0.14 (0.05-0.31)	3.4 (2.7-4.3)	1.2 (0.9-1.5)
	6	1832			

Table 6.5: Holden Results on SpamAssassin Corpus

Filter	Contingency table		% Ham Misc.	% Spam Misc.	% Overall Misc.
Annoyance	144	34	1.4 (0.17-4.9)	0.96 (0.67-1.3)	0.97 (0.68-1.3)
	2	3523			
Antispam	84	2	42.5 (34.4-50.9)	0.06 (0.007-0.2)	1.7 (1.3-2.2)
	62	3555			
Bayesspam	117	189	19.9 (13.7-27.2)	5.3 (4.5-6.1)	5.9 (5.2-6.7)
	29	3668			
bmf	138	25	5.5 (3.4-10.5)	0.7 (0.5-1.0)	0.9 (0.6-1.2)
	8	3532			
Bogofilter	146	169	0 (0-2.0)	4.8 (4.1-5.5)	4.6 (3.9-5.3)
	0	3338			
CRM114	119	19	18.5 (12.6-25.8)	0.5 (0.3-0.8)	1.2 (0.9-1.7)
	27	3538			
dbacl	137	2224	6.2 (2.9-11.4)	62.5 (60.9-64.1)	60.3 (58.1-61.9)
	9	1333			
DSPAM	146	1723	0 (0-2.0)	48.4 (46.8-50.1)	46.5 (44.9-48.2)
	0	1834			
lfile	144	90	1.4 (0.2-4.8)	2.5 (2.0-3.1)	2.5 (2.0-3.0)
	2	3467			
qsf	146	149	0 (0-2.0)	4.1 (3.6-4.9)	4.0 (3.4-4.7)
	0	3408			
SpamAssassin	146	1799	0 (0-2.0)	50.6 (48.9-52.2)	48.6 (47.0-50.2)
	0	1758			
SpamBayes	145	189	0.7 (0.2-3.8)	5.3 (4.6-6.1)	5.1 (4.4-5.9)
	1	3368			
SpamOracle	144	406	1.4 (0.2-4.8)	11.4 (10.4-12.5)	11.0 (10.0-12.1)
	2	3151			
SpamProbe	136	9	6.8 (3.3-12.2)	0.25 (0.12-0.48)	0.51 (0.31-0.80)
	10	3548			

Table 6.6: Holden Results on Personal Email

Holden		Zhang et al.	
Filter	TCR ($\lambda = 9$)	Method	Best TCR (approx, $\lambda = 9$)
Annoyance	8.0	Naive Bayes	1.9
Antispam	1.5	Max. Entropy	15.2
Bayesspam	1.7	Memory Based	7.0
bmf	7.9	SVMlight	12.1
Bogofilter	8.2	Boost Stumps	10.4
CRM114	3.7		
dbacl	0.1		
DSPAM	9.8		
lfile	2.7		
qsf	6.2		
SpamAssassin	14.7		
SpamBayes	12.9		
SpamOracle	6.1		
SpamProbe	15.8		

Table 6.7: Holden and Zhang et al. Results on SpamAssassin Corpus

observing a preponderance of messages like welcome advertising, news clippings, mailing lists, etc. Many other studies have used the SpamAssassin corpus; Meyer & Whateley[87]; Yerazunis[133]; Zhang et al.[138].

Zhang et al.[138] evaluate several learning algorithms on four corpora – Ling Spam, PU1, SpamAssassin, and ZH1. ZH1 is a private corpus of 1633 Chinese messages with headers; 428 ham and 1205 spam. Only the TCR statistic is reported ($\lambda = 9$ and $\lambda = 999$); from this statistic it is impossible, in general, to recover sm and hm. In the specific case of $\lambda = 999$ we may deduce that the best-ranked classifiers had no ham misclassifications (i.e. $c = 0$) and we may use this deduction combined with corpus statistics to compute hm and sm. TCR for these filters, on the SpamAssassin corpus, was approximately 12. Suppose $c > 0$. Because b and c are whole numbers, we have $b \geq 0$ and $c \geq 1$. We have $TCR = \frac{b+d}{\lambda c+b} \leq \frac{1897}{999+0} = 1.9$, which contradicts the reported value. Therefore $c = 0$, $b \approx 158$, $hm = 0\,(0\% - 0.07\%)$, $sm \approx 8.3\%\,(7.1\% - 9.7\%)$. The confidence interval for hm should be interpreted with caution because Zhang et al. adjusted the threshold parameter θ_{999} so as to optimize TCR, effectively fixing $c = 0$ for the corpus data. For $\lambda = 9$ we are unable to deduce the values of b and c, so in table 6.7 we present as TCR these results, and also Holden's, on the SpamAssassin corpus. We note that this comparison is not entirely valid, as the filters for which the threshold and other parameters were not adjusted to optimize the result (i.e. the filters tested by Holden and, we understand, the Bayes method of Zhang et al.) are at considerable disadvantage.

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

Table 6.8: Tuttle Results

Tuttle et al.[130] evaluate three common machine-learning algorithms – naïve Bayesian classifiers, support vector machines, and boosted decision trees – within the context of an enterprise mail system. They deployed a novel architecture to capture email messages and judgements from several users, keeping this information private and under the control of the users to whom the messages belonged. Test runs were "pushed" to the users' corpora, and only statistics were reported back to the central system. Seven users participated in the study, and corpora consisting of up to 800 messages per user were subject to ten-fold cross-validation. Results for each of the seven corpora were computed and the mean of these results was reported. The primary experiment used individual corpora with 400 messages each, approximately 62% spam, and reported piece-wise ROC curves (see table 6.8) for $hm \in \{0.0\%, 0.5\%, 1.0\%, 2.0\%, 5.0\%\}$. Other experiments fixed $hm = 1.0\%$ as a proxy for the operating range. The published averages yield insufficient information to compute confidence intervals but, we note, the overall sample size of 2800 suggests that they would be comparable in magnitude to those for Sahami et al. and Androutsopoulos et al. Tuttle et al. perform a 2-factor analysis of variance and conclude that there is a significant difference in results among the seven corpora, but not among the three filters.

Kolcz and Alspector[73] model the cost of misclassifying various genres of messages. This approach stands in contrast to the cost-sensitive methods discussed above, which assume the cost to be dependent only on whether the message is ham or spam. It also stands in contrast to ours, in which the test method and quantitative evaluation measures assume no particular cost model, and message genres are treated qualitatively. Kolcz and Alspector assembled a corpus of 11408 messages (6043 ham; 5365 spam) which were labeled according to category; each category was assigned an estimated cost of misclassification. The corpus was split into training and test sets in a 3:1 ratio. Results are reported in terms of TCR and ROC analysis. Although they published their intent to do so, Kolcz and Alspector were unable to publish their corpus due to corporate restrictions on the data.

Method/features	trec05p-1	trec06p	trec07p
Fusion[33, 83]	0.007	0.020	
SVM/4-gram[109]	0.008	0.023	
ROSVM/4-gram[109, 108]	0.009	0.024	0.009
DMC+LR/4-gram[39]	0.006		
LR/4-gram[39]	0.012	0.006	
DMC[17, 38]	0.013	0.033	0.008
Perceptron/string[110]	0.017	0.041	
PPM[17]	0.019	0.061	0.011
Clustering/string[92]	0.011		
OSBF-Lua[9]	0.019	0.054	0.028
LR/words[53]	0.022		
Bogofilter[37]	0.048		
SpamAssassin/"Bayes"[37]	0.059		
LR/words[35]	0.068		
SVM/words[35]	0.075		
SpamAssasinRules+Bayes[37]	0.345		

Table 6.9: 1-ROC(%) – TREC Public Corpora

6.2 Independent Studies and Methodology

This section contains an overview of independent studies completed between 2004 and 2009. All results are presented using the (1-ROCA)% measure.

TREC Spam Track

Repeated in 2006 and 2007, the TREC Spam Track is the largest and most realistic laboratory evaluation of spam filters to date. Over the three years of the TREC Spam Track more than one hundred filters were evaluated and ten corpora (721,461 messages) were created. Of the created corpora, four are publicly available.[1] The results of the best-performing filters on the public corpora are presented in table 6.9[38].

Each year TREC adds new tasks. The original spam filtering task, immediate feedback, is described in Chapter 4. In 2006 a delayed feedback task was introduced. The delayed feedback task would inform the filter of adjudicated classification only after some number of additional messages had been classified by the filters. TREC 2006 also introduced an (batch) active learning task in which filters were presented with an unlabeled

[1] `http://trec.nist.gov/data/spam.html`

Team	(1-ROCA)%
Lahore University of Management Sciences[68]	4.93
University of Waikato[95]	5.09
Inductis India Pvt Ltd[59]	5.13
National Technical University of Athens National Technical University of Athens [128]	6.35
Beihang University[16]	7.23

Table 6.10: ECML/PKDD Discovery Challenge 2006 Task A

Team	(1-ROCA)%
University of Waterloo[32]	5.35
National Technical University of Athens National Technical University of Athens [128]	8.18
Inductis India Pvt Ltd[59]	9.26
Moscow State University[16]	10.08
Apex Data & Knowledge Management Lab[16]	10.67

Table 6.11: ECML/PKDD Discovery Challenge 2006 Task B

training set and were allowed to choose a set number example for which adjudicated classification would be given. TREC 2007 introduced an on-line active learning task. Also introduced was an extreme delayed feedback in which feedback was given immediately for the first few thousand messages and never given again. TREC 2007 also had a fourth task, partial feedback, in which feedback was given for only a subset of the messages.

2006 ECML/PKDD Discovery Challenge

The 2006 ECML/PKDD Discovery Challenge[16] consisted of two batch oriented partial feedback tasks. Task A used a single train set of 4000 messages and tested on three test sets of 2500, each though only one of the test tests was used for feedback. Task B used three training sets of 100 each and fifteen test sets of 400 messages each, though on one of the test sets contain labels. The results for task A are in table 6.10 and task B in table 6.11.

CEAS 2008 Challenge

The 2008 CEAS Live Spam Challenge[112] extended upon the evaluation completed in the TREC spam tracks. TREC methodology was modified in several ways. First, email was captured and distributed in real-time. Second, the standard email protocols (SMTP, IMAP, POP) were used in place of the toolkit command-line interface. Finally, there were no limits to the external resources filters could use for the live competition. Four task were completed:

- 72-hour live email spam filtering competition
- laboratory simulation of the live steam using the same messages
- laboratory evaluation on a large stream of real private data collected by a service provider
- laboratory active learning on the private data stream.

Tables 6.12, 6.13, 6.14 show the top result for the live competition,laboratory live stream and laboratory private data.

Team	(1-ROCA)%
OSBF-Lua (TREC06-ofl)[9]	0.01270
Solido Systems[126]	0.01310
CRM114 (TREC07-crm1)[69]	0.03590
Logistic Regression + DMC (TREC07-wat3)[39]	0.03840
Heilongjiang Institute of Technology - Entry 1[98]	0.03980

Table 6.12: CEAS 2008 Challenge Live Competition

Team	(1-ROCA)%
Communication and Computer Network Lab - Entry 3[26]	0.01781
Heilongjiang Institute of Technology - Entry 2[39]	0.01968
PPM Compression (TREC07-ijsppm)[34]	0.02333
Dynamic Markov Compression(TREC07-wat2)[39]	0.02347
Heilongjiang Institute of Technology - Entry 3[39]	0.02770

Table 6.13: CEAS 2008 Challenge Lab Live Stream

Team	(1-ROCA)%
Logistic Regression + DMC (TREC07-wat3)[39]	5.671
Logistic Regression (TREC07-wat1)[39]	5.752
Dynamic Markov Compression (TREC07-wat2)[39]	6.766
IGF (gor Assis Braga) - Entry 2[112]	12.629
Kosmopoulos Aris - Entry 1[112]	28.745

Table 6.14: CEAS 2008 Challenge Lab Private

Vendor	fpr(%)	fnr(%)
BorderWare	0.12	1
Postini	0.24	3
CipherTrust	0.36	3
Symantec	0.48	3
Advascan	0.57	3
NetCleanse	0.46	5
Proofpoint	0.87	3
Barracuda	0.9	5
Cloudmark	1.05	5
Spamfighter	1.02	14

Table 6.15: Network World Test

Network World Test

Network World[119] conducted a parallel test of 41 commercial spam filters. Over a two-week period the message traffic to a server was captured, screened for viruses and backscatter, and passed on to the spam filters under test. In total 11,000 (8027spam, 2386 ham, 600 other) messages were passed to the filters. The 600 other ones that were rejected due to screening problems, duplicate message identifiers and adjudicators unable to determine the correct classification. Filters were allowed unlimited access to external resources normally used. Results are found in table 6.15.

VeriTest

The VeriTest Anti-Spam Benchmark Service[129] provides a service that measures the effectiveness of anti-spam technologies. Filters are installed and operate normally. VeriTest send the servers ham mail from real sources and spam from traps. The tests are completed of a corpus of around 10,000 messages. The results of the 2005 test are in table 6.16.

SEWM

The SEWM 2007 and 2008 Spam Tracks [4, 3] adopt the toolkit and methodology for large-scale evaluation of Chinese email. SEWM materials and results are published in Chinese only.

Vendor	fpr(%)	fnr(%)
BitDefender	0	3
Email Systems	0	1
Kaspersky Labs	0	3
MXSweep	0	1
SoftScan	0	3
Sophos	0	3
SurfControl	0	1

Table 6.16: Veritest 2005 Results

6.3 Datasets and Tools

Since 2004 many new spam corpora have been created. The majority of these corpora have been created using the TREC 2005 public corpora creation methodology. There has been no real development in evaluation tools other than updates to the TREC Spam Filtering Evaluation Toolkit.

6.3.1 Corpora

Table 6.17 describes the details of spam corpora that have been used in studies to evaluate spam filters. Corpora that have public availability are in the public domain and can be used for future studies; however, corpora with private availability can only be used by the original authors. If one considers statistical significance, many studies use under-size corpora. This is especially true for the studies that use large training sets and small test sets.

- The Enron-spam[86] is a successor to the Ling spam and PU corpora. The corpus contains chronological email messages from six Enron employees. There are two forms of the corpus: one with the emails in their raw form, the other with similar abstractions to Ling Spam and PU corpora. The corpus is divided into six train and test sets representing the six employees.
- Gen Spam[85] corpus contains training messages (38246), adaptation messages (600) and test messages (1551). The headers have been reformatted into XML and any identifying information has been obfuscated.
- The Holden2[64] is a private corpus that contains one month of personal email from the same source as holden1 but newer mail. The corpus includes qualitative descrip-

Corpus	# email	# ham	# spam	availability	source
Holden2	3703	146	3557	private	1 mth personal
ZH1	1633	428	1205	private	chinese personal
Gen Spam	40414	9072	32332	public	personal sample
SpamGuru	170000	40000	130000	private	multiple user personal
Enron-spam	33716	16545	17171	public	personal sample
MrX	40048	9038	49086	private	8 mths personal
MrX2	49174	9039	40135	private	7 mths personal
MrX3	161975	8082	153893	private	7 mths personal
trec05p	39399	52790	92189	public	personal sample,spam traps
trec06p	37822	12910	24912	public	web, traps
trec06c	64620	21766	42854	public	chinese mailing list,traps
trec07p	75419	25220	50199	public	multiple,personal sample
SB	7006	6231	775	private	1 year personal
SB2	11969	9274	2695	private	1 year personal
TM	170201	150685	19516	private	1+ year personal
Microsoft	915000	?	?	private	random sample hotmail
ECML/PKDD(Task A)	11500	5750	5750	public	multiple, personal sample
ECML/PKDD(Task B)	6300	3150	3150	public	multiple, personal sample
ceas08(private)	198000	90000	109000	private	real email from one domain
ceas08(public)	143000	28000	115000	private	ham/spam traps
SEWM07(public)	60000	15000	45000	public	chinese unknown
SEWM08	70000	20000	50000	public	chinese unknown

Table 6.17: Spam Corpora 2004-2009

tions of the misclassified ham messages, observing a preponderance of messages like welcome advertising, news clippings, mailing lists, etc.

- ZH1[138] is a private corpus of 1633 Chinese messages with headers; 428 ham and 1205 spam.
- SpamGuru[114] corpus is a collection of mail from 200 IBM employees.
- TREC corpora were created using the corpus creation methodology presented earlier. Results for TREC05p are found in figure 4.7. TREC06p[33] was creating in two steps, by searching and download ham messages from the web, inject spam collected from traps to look like the downloaded ham. TREC06c is a collection of ham from Chinese mailing list and spam from traps. TREC07p[34] includes mail sample from multiple users as well has spam from traps.
- SB[30] and SB2[33] are two corpora of personal email from the same user over two different periods, 2005, 2006 respectfully, The corpora is in raw unmodified spool form. SB and SB2 were also created using the TREC corpus creation methodology. SB was created for TREC 2005 and SB2 was created for TREC 2006.
- TM[30] is personal email from one person over a period of more than a year. The corpus is in raw unmodified spool form. TB was used in TREC05.
- Microsoft[135] corpus is a collection of random hotmail over a period of five months. The messages were converted to small sets of features. A 3% human classification error rate was found.
- SEWM07(public)[3] and SEWM08[4] are a collection of Chinese mail used in a Chinese version of the TREC spam track. The corpora appear to be in TREC spam filter evaluation toolkit format.
- The CEAS challenge used two different corpora; one public and one private. CEAS08 public[112] is a collection of email accumulated through both spam and ham traps. CEAS08 private[112] is a collection of randomly sampled email from one domain. The gold standard for the corpus reproduced actual user feedback. This include delays, partial and incorrect feedback. CEAS08 corpora were also created using the TREC corpus creation methodology.
- The creation of MrX corpora is described in Chapter 3. MrX2[33], and MrX3[34] are corpora from the same user and in the same format as MrX but are newer email streams.

- ECML/PKDD corpora are emails in a bag-of-words vector space representation. Attributes are the term frequencies of the words. Words with less than four occurrences are removed from the data set resulting in a dictionary size of about 150,000 words. Only 6500 of data set A are label and only 700 of data set B.

6.3.2 Tools

The TREC Spam Filtering Evaluation Toolkit was been updated to include the ability to delay feedback, give partial feedback and perform active on-line learning. In 2006 delay and partial feedback were added by Cormack [33]. In 2007 active on-line learning was added by Sculley [34, 106]. Sculley and Cormack measure the effect of training label errors on on-line filter performance [107].

6.4 Evolution of Spam Filtering

On-line spam filter effectiveness has improved considerably, arguably due to our standardized evaluation, test data and tools. Several methods have improved on the best-performing filters of 2004:

- orthogonal sparse bi-gram feature engineering with threshold training [9, 117];
- on-line logistic regression using gradient descent[53];
- on-line support vector machines with character 4-gram features [108];
- sequential data compression models [17];
- spam filter fusion [83].

Figure 6.1 shows the top ten independently evaluated filters for 2004, 2005, 2006 and 2007. 2004 filters were evaluated in the study On-line Supervised Spam Filter Evaluation. 2005, 2006 and 2007 filters were evaluated at TREC 2005, 2006 and 2007 respectively. All evaluations were completed on one of the MrX email streams. 2004 and 2005 filters used MrX, 2006 filters used MrX2 and 2007 filters used MrX3. The squares represent individual filters while the triangles represent the log-odds fusion of all the filters evaluated at TREC. Even as filters improve, fusion still outperforms the best filter by a significant margin.

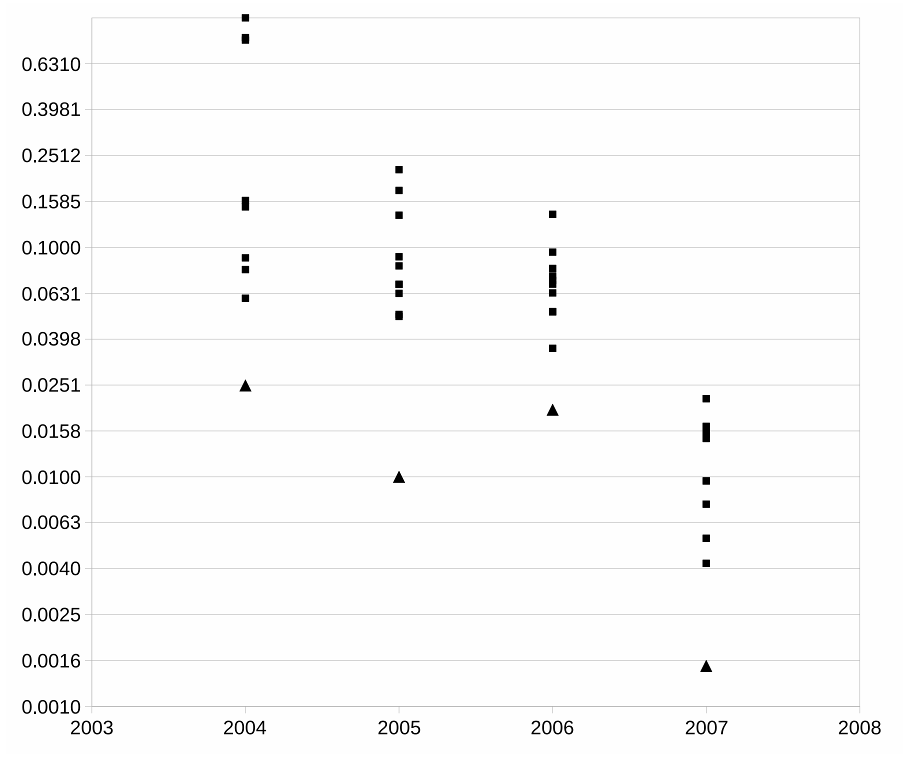


Figure 6.1: Spam Filters Improvement(1-ROCA%) - Top 10 per Year

Chapter 7

Conclusion

Sound quantitative evaluation of spam filtering has been central to improvements in filter effectiveness since 2004. From our first study to date, the best individual filters have improved substantially. Even as individual filters improve, fusion continues to improve on the best by a substantial margin.

The specific contributions of this work are:

- A realistic methodology for the evaluation of on-line spam filters with user feedback, incorporating a standard method of simulating filter deployment and a standard set of effectiveness measures;
- The first comparative study of on-line filter performance on a large, comprehensive, real email stream;
- Deployment of the methodology through a free toolkit and the TREC Spam Track;
- Two archival corpora for filter evaluation, one private and one public, and a methodology for creating them;
- Spam Filter Fusion, which consistently improves on all other techniques.

Limitations of the work suggest opportunities for enhancement along several lines:

- The laboratory environment mandates certain simplifying assumptions which may be unrealistic. In particular, the simulation of feedback assumes an ideal user who promptly reports filter errors to the filter. A more realistic assumption is that users report some but not all errors, and are more likely to report false negatives than false positives, because the latter are rare and more likely to remain unnoticed. The toolkit has proven amenable to enhancements that better emulate user behaviour; the limiting factor in such efforts is coming up with and validating a model of user behaviour to be simulated using the tools.

- Restrictions on external resource access for the sake of repeatability preclude the evaluation of techniques like RBLs and collaborative filtering. A hybrid approach, employed by CEAS, tests such filters in real-time, capturing the messages for use in later laboratory evaluation. In this way, the effectiveness of content-based filters may be compared to that of filters relying on system-wide resources. An alternative approach is to simulate the behaviour of the entire system, as proposed by Aycock[11, 91]. A trade off exists between the comprehensiveness of the simulation and the realism of its components.

- The CEAS experiments and others suggest that filters employing user feedback are more effective than those that do not. It has been suggested that a substantial number of users are unable or unwilling to provide feedback. We suggest that this thesis is far from established, and that the design and evaluation of interfaces to engage users is a promising avenue for future research. The issue of how to improve filter effectiveness without user feedback is, in itself, well worthy of further study. The intellectual uncertainty in this effort is the discovery of the techniques, as the methodology is easily applied to their evaluation. All that needs to be done is to suppress user feedback during the evaluation. Batch evaluation may also be simulated using the toolkit [35].

- The laboratory environment largely precludes the evaluation of popular techniques like greylisting and challenge-response which engage the sender. At the time of writing, claims regarding the efficacy of these methods are unsubstantiated by sound measurement. In particular, *hm* is difficult to measure, as messages deemed to be spam are never delivered.

- It is possible to compare filters outside the confines of the laboratory environment. In this context, repeatability is established using statistical inference to bound the imprecision in measurement due to external variances. Standard methods and experimental designs of social and medical research may be employed [103]. These methods involve orders of magnitude more time and expense than laboratory evaluation, and are subject to their own limitations. Nevertheless, they yield valuable confirmation (or refutation) of the results of laboratory evaluation. Possible designs include:

  - Randomized controlled trials, in which users from a source population are selected at random to employ different filtering techniques. Randomized trials are among the most effective, but also the most intrusive and expensive, designs.
  - Crossover studies, in which a user switches between filters, and the results are compared. Like randomized controlled trials, crossover studies are intrusive; however, a much smaller number of users is required to achieve adequate statistical repeatability. Crossover studies are limited by crossover effects, in that the user's behaviour with respect to one filter is influenced by prior use of the other.
  - Cohort studies in which the results of filters that happen to be in use by particular users are compared. Cohort studies are less intrusive than randomized or crossover designs, but some method of capturing filter results is required. Controlling for factors predisposing users to a particular filter is a substantial challenge.
  - Case-control studies examine the characteristics of users and filters based on observed outcomes. For example, one might solicit contributions to a spam archive, and determine the filter that was used by the contributor along with that filter's result. These results may be combined to estimate the effectiveness of the filters involved. Case-control studies are among the cheapest to conduct, but controlling for confounding factors is a formidable challenge.

- Regardless of the experimental design, the standard measures of filter effectiveness are applicable. That said, the measures cover only certain aspects of filter performance. The resource consumption and delay associated with filter use may be important considerations both from the system and user perspective. Different messages may have different value to the user; aptly modeling this value remains an outstanding research area. While the summary measures ROCA and LAM are robust to obvious differences in deployment parameters, and yield similar filter rankings, they may not aptly reflect effectiveness in any particular situation.

Overall, I conclude that while spam filters are effective enough to be useful for their intended purpose, there is still much room for improvement. The methods and tools described serve as impetus for the improvement of filter effectiveness and of filter evaluation methods.

References

[1] The definition of spam. `http://www.spamhaus.org/definition.html`. 1

[2] Ling-spam, pu, enron corpora. `http://www.iit.demokritos.gr/skel/iconfig/downloads`. 28

[3] Sewm 2007 spam filtering system evaluation description. `http://net.pku.edu.cn/~webg/cwt/2007WebTrack/SEWM2007SpamTrackGuide.pdf`, 2007. 103, 106

[4] Sewm 2008 spam filtering system evaluation description, 2008. 103, 106

[5] Alan Agresti. *An Introduction to Categorical Data Analysis.* Wiley, New York, 1996. 36

[6] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, , and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France, 2000. 62, 94, 95

[7] I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. Technical Report 2004/2, NCSR Demokritos, 2004. 24, 28, 95

[8] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *CoRR*, cs.CL/0006013, 2000. informal publication. 28, 95

[9] Fidelis Assis. OSBF-Lua - A Text Classification Module for Lua The Importance of the Training Method. In *Fifteenth Text REtrieval Conference (TREC-2006)*, Gaithersburg, MD, 2006. NIST. 100, 102, 107

[10] John Attia. Moving beyond sensistivity and specificity: using likelihood ratios to help interpret diagnostic tests. *Australian Prescriber*, 26(5):111–113, 2003. 87

[11] John Aycock, Heather Crawford, and Rennie deGraaf. Spamulator: the internet on a laptop. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 142–147, New York, NY, USA, 2008. ACM. 110

[12] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, 1994. 29

[13] N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. In *TREC-2: Proceedings of the second conference on Text retrieval*, pages 431–448, Gaithersburg, 1995. NIST. 29

[14] Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. The combination of text classifiers using reliability indicators. *Inf. Retr.*, 8(1):67–100, 2005. 29

[15] Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, 1979. 82

[16] Steffen Bickel. ECML-PKDD Discovery Challenge 2006 Overview. In *Proc. ECML/PKDD Discovery Challenge Workshop*, 2006. 101

[17] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7(Dec):2673–2698, 2006. 100, 107

[18] Laird Breyer. Laird breyer's free software - dbacl. `http://www.lbreyer.com/gpl.html`, 2005. 21

[19] brightmail. Symantec brightmail antispam, 2004. 18, 21

[20] Brian Burton. Spamprobe - a fast bayesian spam filter. `http://spamprobe.sourceforge.net`, 2002. 21, 34

[21] Marco Caliendo, Michel Clement, Dominik Papies, and Sabine Scheel-Kopeinig. The Cost Impact of Spam Filters: Measuring the Effect of Information System Technologies in Organizations. *SSRN eLibrary*, 2008. 2

[22] Claudio Carpineto, Giovanni Romano, and Vittorio Giannini. Improving retrieval feedback with multiple term-ranking function combination. *ACM Trans. Inf. Syst.*, 20(3):259–290, 2002. 14

[23] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001. Available: `http://www.lsi.upc.es/~carreras/pub/boospam.ps`. 15, 16

[24] K Chellapilla, K Larson, K Simard, and M Czerwinski. Designing human friendly human interactive proofs (HIPS). In *CHI '05: SIGCHI Conference on Human factors in computing systems*, pages 711–720, 2005. 14

[25] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs. In *CEAS 2005 – The Second Conference on Email and Anti-Spam*, 2005. 14

[26] Bin Chen, Shoubin Dong, and Weidong Fang. Introduction of fingerprint vector based bayesian method for spam filtering. In *5th Conference on Email and Anti-Spam*, 2008. 102

[27] Cloudmark. Sendmail chooses cloudmark as the best anti-spam solution for its thousands of enterprise customers after evaluating more than 40 anti-spam products. `http://www.cloudmark.com/en/company/xsrelease.html?release=2003-11-03`, 2003. 21

[28] CMU. Enron email dataset. Available at `http://www.cs.cmu.edu/~enron/`. 62

[29] William K. Cole. Blacklists, blocklists, dnsbl's, and survival:. `http://www.scconsult.com/bill/dnsblhelp.html`. 13

[30] G. V. Cormack and T. R. Lynam. TREC 2005 Spam Track overview. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. 1, 4, 32, 106

[31] Gordon Cormack and Thomas Lynam. A study of supervised spam detection applied to eight months of personal email. `http://plg.uwaterloo.ca/~{}gvcormac/spamcormack.html`, 2004. 44

[32] Gordon V. Cormack. Harnessing unlabeled examples through iterative application of Dynamic Markov Modeling. In *Proc. ECML/PKDD Discovery Challenge Workshop*, 2006. 101

[33] Gordon V. Cormack. TREC 2006 Spam Track Overview. In *Fifteenth Text REtrieval Conference (TREC-2006)*, Gaithersburg, MD, 2006. NIST. 1, 100, 106, 107

[34] Gordon V. Cormack. TREC 2007 Spam Track Overview. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD, 2007. NIST. 1, 102, 106, 107

[35] Gordon V. Cormack and Andrej Bratko. Batch and on-line spam filter evaluation. In *CEAS 2006 – The 3rd Conference on Email and Anti-Spam*, Mountain View, 2006. 82, 90, 100, 110

[36] Gordon V. Cormack and Thomas R. Lynam. Spam corpus creation for TREC. In *Proc. CEAS 2005 – The Second Conference on Email and Anti-Spam*, 2005. 1, 37

[37] Gordon V. Cormack and Thomas R. Lynam. On-line supervised spam filter evaluation. *ACM Transactions on Information Systems*, 2007. 1, 100

[38] G.V. Cormack. Email Spam Filtering: A Systematic Review. *Foundations and Trends in Information Retrieval*, 1(4):335–455, 2007. 100

[39] G.V. Cormack. University of waterloo participation in the TREC 2007 spam track. In *Sixteenth Text REtrieval Conference (TREC-2007)*, 2007. 100, 102

[40] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. 15, 16

[41] B.V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991. 15, 16

[42] Jamie Deguerre. The mechanics of Vipul's Razor. *Network Security*, pages 15–17, September 2007. 13

[43] Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000. 29

[44] Nathan Dimmock and Ian Maddison. Peer-to-peer collaborative spam detection. *ACM Crossroads*, 11(2), 2004. 13

[45] H. Drucker, Donghui Wu, and V. N. Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054, 1999. 21, 25, 28

[46] Saso Dzeroski and Bernard Zenko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, 2004. 30

[47] Tom Fawcett. 'in vivo' spam filtering: A challenge problem for data mining. *KDD Explorations*, 5(2), December 2003. Available: `http://www.hpl.hp.com/personal/Tom_Fawcett/papers/spam-KDDexp.pdf`. 26

[48] Tom Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. Technical report, HP Laboratories, 2003. 25

[49] R. A. Fisher. Theory of statistical estimation. *Proceedings of the Cambridge Philosophical Society*, 22:700–725, 1925. 33

[50] Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999. 15

[51] SI Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990. 15

[52] Afina S. Glas, Jeroen G. Lijmer, Martin H. Prins, Gouke J. Bonsel, and Patrick M. M. Bossuyt. The diagnostic odds ratio: a single indicator of test performance. *Journal of Clinical Epidemiology*, 56(11):1129–1135, 2003. 61

[53] Joshua Goodman and Wen tau Yih. Online discriminative spam filter training. In *The Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006. 100, 107

[54] Joydeep Gosh. Multiclassifier systems: Back to the future. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems (MCS2002)*, volume LNCS 2364, pages 1–15. Springer, 2002. 77

[55] Paul Graham. A plan for spam. In *Reprinted in Paul Graham, Hackers and Painters, Big Ideas from the Computer Age, O????Really, 2004*, 2002. Available: `http://www.paulgraham.com/spam.html`. 11, 15, 34

[56] Paul Graham. Better bayesian filtering. In *Proceedings of the 2003 Spam Conference*, Jan 2003. 11, 15, 17, 21, 34

[57] John Graham-Cumming. People and spam. In *The Spam Conference*, 2005. 15

[58] John Graham-Cumming. Why I hate challenge-response. *JGC's Anti-Spam Newsletter*, Feb. 28, 2005. 14

[59] Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, and Chirag Taneja. Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data. In *Proc. ECML/PKDD Discovery Challenge Workshop*, 2006. 101

[60] Evan Harris. Greylisting: The next step in the spam control war. `http://projects.puremagic.com/greylisting/`, 2003. 13

[61] Peizhou He, Yong Sun, Wei Zheng, and Xiangming Wen. Filtering short message spam of group sending using captcha. In *Proc. International Workshop on Knowledge Discovery and Data Mining WKDD 2008*, pages 558–561, 23–24 Jan. 2008. 14

[62] José M. Gómez Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *Proceedings of SAC-02, 17th ACM Symposium on Applied Computing*, pages 615–620, Madrid, ES, 2002. 11, 24, 25

[63] Paul Hoffman. Unsolicited bulk email: Definitions and problems, 1997. 8

[64] S. Holden. Spam filtering II. *Hakin9*, 02/2004:68–77, 2004. 95, 104

[65] Sam Holden. Spam filtering. *Freshmeat*, 2003. 21, 29

[66] David A. Hull, Jan O. Pedersen, and Hinrich Schutze. Method combination for document filtering. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 279–287. ACM Press, 1996. 29, 30, 87

[67] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998. 82

[68] Yousaf Junejo and A. Karim. A two-pass statistical approach for automatic personalized spam filtering. In *Proc. ECML/PKDD Discovery Challenge Workshop*, 2006. 101

[69] M. Kato, J. Langeway, Y. Wu, and W.S. Yerazunis. Three NonBayesian Methods of Spam Filtration: CRM114 at TREC 2007. In *Proceedings of TREC*, 2007. 102

[70] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998. 29

[71] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. 62

[72] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995. 26

[73] Aleksander Kolcz and Joshua Alspector. SVM-based filtering of e-mail spam with content-specific misclassification costs. In *Proceedings of the TextDM'01 Workshop on Text Mining - held at the 2001 IEEE International Conference on Data Mining*, 2001. 20, 21, 25, 28, 99

[74] Aleksander Kolcz and Abdur Chowdhury. Hardening fingerprints by context. In *CEAS 2007 – The Third Conference on Email and Anti-Spam*, 2007. 13

[75] Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. The impact of feature selection on signature-driven spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Available: `http://www.ceas.cc/papers-2004/147.pdf`. 13

[76] Paul Komarek and Andrew Moore. Fast robust logistic regression for large sparse datasets with binary outputs. In *Artificial Intelligence and Statistics*, 2003. 83

[77] Wai Lam and Kwok-Yin Lai. A meta-learning approach for text categorization. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 303–309. ACM Press, 2001. 29

[78] Johannes Lenhard. Models and statistical inference: The controversy between Fisher and Neyman-Pearson. *British Journal for the Philosophy of Science*, 2006. 33

[79] John Levine. Experiences with greylisting. In *CEAS 2005: Second Conference on Email and Anti-Spam*, 2005. 13

[80] B Lieba, J Ossher, V T Rajan, R Segal, and M Wegman. SMTP path analysis. In *2nd Conference on Email and Anti-spam*, 2005. 13

[81] Thomas Lynam and Gordon Cormack. TREC spam filter evaluation took kit. `http://plg.uwaterloo.ca/~{}trlynam/spamjig`, 2005. 1

[82] Thomas R. Lynam, Chris Buckley, Charles L. A. Clarke, and Gordon V. Cormack. A multi-system analysis of document and term selection for blind feedback. In *CIKM '04: Thirteenth ACM conference on Information and knowledge management*, pages 261–269, 2004. 29

[83] Thomas R. Lynam and Gordon V. Cormack. On-line spam filter fusion. In *29th ACM SIGIR Conference on Research and Development on Information Retrieval*, Seattle, 2006. 1, 100, 107

[84] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The DET curve in assessment of detection task performance. In *Proc. Eurospeech 97*, pages 1895–1898, Rhodes, Greece, 1997. 25

[85] Ben Medlock. An adaptive, semi-structured language model approach to spam filtering on a new corpus. In *Proc. CEAS 2006 – Third Conference on Email and Anti-Spam*, Mountain View, CA, 2006. 104

[86] Vangelis Metsis. Spam filtering with naive bayes - which naive bayes. In *In 3rd Conference on Email and Anti-Spam, Mountain View, ca*, 2006. 104

[87] T.A 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)*, 2004. 98

[88] Eirinaios Michelakis, Ion Androutsopoulos, Georgios Paliouras, George Sakkis, and Panagiotis Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. 21, 28, 62

[89] Trend Micro. Mail abuse prevention system. `http://www.mail-abuse.com/`, 2005. 13

[90] Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *CIKM '02: Eleventh international conference on Information and knowledge management*, pages 538–548, 2002. 29

[91] Margaret Nielsen, Dane Bertram, Sampson Pun, John Aycock, and Nathan Friess. Global-scale anti-spam testing in your own back yard. In *5th Conference on Email and Anti-Spam*, 2008. 110

[92] Junyu Niu, Jun Xu, Jing Yao, Jiaquan Zheng, and Qi Sun. WIM at TREC 2007. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD, 2007. NIST. 100

[93] Patrick Pantel and Dekang Lin. SpamCop: A spam classification and organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. 17, 19, 21, 28

[94] Tim Peters. Spambayes: Bayesian anti-spam classifier in python. `http://spambayes.sourceforge.net/`, 2004. 34

[95] Bernard Pfahringer. A semi-supervised spam mail detector. In *Proc. ECML/PKDD Discovery Challenge Workshop*, 2006. 101

[96] popa3d. popa3d - a tiny pop3 daemon. `http://www.openwall.com/popa3d/`. 66

[97] The Penny Black Project. The penny black project. `http://research.microsoft.com/research/sv/PennyBlack/`. 14

[98] H. Qi, X. He, M. Yang, J. Li, G. Lei, and S. Li. Joint NLP Lab between HIT 2 at CEAS Spam-filter Challenge 2008. In *CEAS*, 2008. 102

[99] Eric S. Raymond, David Relson, Matthias Andree, and Greg Louis. Bogofilter. `http://bogofilter.sourceforge.net/`, 2004. 21, 34

[100] Gary Robinson. Spam detection. `http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html`, 2002. 11, 15

[101] Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, 107(107), 2003. 11, 15, 34

[102] Gary Robinson. Gary robinson's spam rants. `http://radio.weblogs.com/0101454/categories/spam/`, 2004. 11, 15, 34

[103] Kenneth J. Rothman and Sander Greenland. *Modern Epidemiology.* Lippinscott Williams and Wilkins, 1998. 110

[104] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. 17, 19, 21, 25, 28, 93

[105] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proc. 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 44–50, Pittsburgh, PA, 2001. 21, 30, 95

[106] D. Sculley. Online active learning methods for fast label-efficient spam filtering. In *CEAS 2007 – The Third Conference on Email and Anti-Spam*, 2007. 107

[107] D Sculley and Gordon V. Cormack. Filtering spam in the presence of noisy user feedback. *Tufts University*, 2008. 107

[108] D. Sculley and Gabriel M. Wachman. Relaxed online support vector machines for spam filtering. In *30th ACM SIGIR Conference on Research and Development on Information Retrieval*, Amsterdam, 2007. 100, 107

[109] D. Sculley and Gabriel M. Wachman. Relaxed online SVMs in the TREC Spam Filtering Track. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD, 2007. NIST. 100

[110] D. Sculley, Gabriel M. Wachman, and Carla E. Brodley. Spam classification with on-line linear classifiers and inexact string matching features. In *Proc. 15th Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, November 2006. 100

[111] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. 8, 24

[112] Richard Segal, Andrej Bratko, and Gordon Cormack. Ceas 2008 spam filter challenge. In *Fifth Conference on Email and Anti-Spam*, 2008. 101, 102, 106

[113] Richard Segal, Jason Crawford, Jeff Kephart, and Barry Leiba. SpamGuru: An enterprise anti-spam filtering system. In *First Conference on Email and Anti-Spam (CEAS)*, 2004. 30

[114] Richard Segal, Jason Crawford, Jeffrey Kephart, and Barry Leiba. Spamguru: An enterprise anti-spam filtering system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Available: `http://www.ceas.cc/papers-2004/126.pdf`. 106

[115] Vipul Sharma and Adam O'Donnell. Fighting spam with reputation systems. *ACM Queue*, November 2005. 13

[116] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *Text REtrieval Conference*, 1994. 29

[117] Christian Siefkes, Fidelis Assis, Shalendra Chhabra, and William S. Yerazunis. Combining Winnow and orthogonal sparse bigrams for incremental spam filtering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino

Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, volume 3202 of *Lecture Notes in Artificial Intelligence*, pages 410–421. Springer, 2004. Available: `http://www.siefkes.net/papers/winnow-spam.pdf`. 15, 107

[118] P Y Simard, R Szeliski, J Benaloh, J Couvreur, and I Calinov. Using character recognition and segmentation to tell computer from humans. In *ICDAR '03: Seventh International Conference on Document Analysis and Recognition*, 2003. 14

[119] Joel Snyder. Spam in the wild, the sequel. *Network World*, 12/20/04, 2004. 103

[120] SpamAssassin. The spamassassin public mail corpus. `http://spamassassin.apache.org/publiccorpus`, 2004. 28, 61, 62, 95

[121] SpamAssassin. The apache spamassassin project. `http://spamassassin.apache.org`, 2005. 14, 34, 37

[122] Spambase. Spambase. `http://mlearn.ics.uci.edu/databases/spambase`. 28

[123] spastic. SPASTIC anit-spam filter. `http://www.nuclearelephant.com/projects/dspam/`, 2002. 21

[124] Springer. *Text categorization with support vector machines: learning with many relevant*, 1998. 15, 16

[125] J. A. Swets. Effectiveness of information retrieval systems. *American Documentation*, 20:72–89, 1969. 25

[126] Solido System. Solido spam filter. `http://www.solidosystems.com/products/spamfilter/`. 102

[127] Brad Templeton. Reaction to the dec spam of 1978. `http://www.templetons.com/brad/spamreact.html`. 1

[128] N. Trogkanis and G. Paliouras. TPN 2: Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2006. 101

[129] tumbleweed. Veritest Anti-Spam Benchmark Service autumn 2005 report. `http://www.tumbleweed.com/pdfs/VeriTest_Anti-Spam_Report_Vol4_all_c.pdf`, 2005. 103

[130] Andrew Tuttle, Evangelos Milios, and Nauzer Kalyaniwalla. An evaluation of machine learning techniques for enterprise spam filters. Technical Report CS-2004-03, Dalhousie University, Halifax, NS, 2004. 25, 29, 99

[131] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992. 29

[132] Bill Yerazunis. Better than human. `http://www.paulgraham.com/wsy.html`, 2002. 12, 17, 21

[133] Bill Yerazunis. The spam-filtering accuracy plateau at 99.9past it. In *Proceedings of the Spam Conference*, 2004. 98

[134] William S. Yerazunis. CRM114 - the controllable regex mutilator. `http://crm114.sourceforge.net/`, 2004. 21, 34

[135] W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006. 106

[136] Jonathan Zdziarski. Dspam v3.2 beta-1 released. `http://developers.slashdot.org/article.pl?sid=04/09/23/123231`, 2004. 17, 21

[137] Jonathan A. Zdziarski. The DSPAM project. `http://www.nuclearelephant.com/projects/dspam/`, 2004. 34

[138] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004. 95, 98, 106

[139] Yi Zhang. Using Bayesian priors to combine classifiers for adaptive filtering. In *SIGIR '04: The 27th Conference on Research and Development in Information Retrieval*, pages 345–352, 2004. 29