Personal Email Spam Filtering with Minimal User Interaction

by

Mona Mojdeh

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, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis investigates ways to reduce or eliminate the necessity of user input to learning-based personal email spam filters. Personal spam filters have been shown in previous studies to yield superior effectiveness, at the cost of requiring extensive user training which may be burdensome or impossible.

This work describes new approaches to solve the problem of building a personal spam filter that requires minimal user feedback. An initial study investigates how well a personal filter can learn from different sources of data, as opposed to user's messages. Our initial studies show that inter-user training yields substantially inferior results to intra-user training using the best known methods. Moreover, contrary to previous literature, it is found that transfer learning degrades the performance of spam filters when the source of training and test sets belong to two different users or different times.

We also adapt and modify a graph-based semi-supervising learning algorithm to build a filter that can classify an entire inbox trained on twenty or fewer user judgments. Our experiments show that this approach compares well with previous techniques when trained on as few as two training examples.

We also present the toolkit we developed to perform privacy-preserving user studies on spam filters. This toolkit allows researchers to evaluate any spam filter that conforms to a standard interface defined by TREC, on real users' email boxes. Researchers have access only to the TREC-style result file, and not to any content of a user's email stream.

To eliminate the necessity of feedback from the user, we build a personal autonomous filter that learns exclusively on the result of a global spam filter. Our laboratory experiments show that learning filters with no user input can substantially improve the results of open-source and industry-leading commercial filters that employ no user-specific training. We use our toolkit to validate the performance of the autonomous filter in a user study.

Acknowledgements

I would like to take this opportunity to acknowledge the many individuals who contributed to this thesis in one way or another. I would like to specially thank

- My supervisor, Professor Gordon Cormack, for believing in me when I need it the most. His never-ending encouragement and patience gave me the freedom to choose my own path. His immense knowledge and scientific curiosity will always be sources of inspiration to me.

- The committee members: Professors Charles Clarke, Mark Smucker, and Olga Vechtomova. Their invaluable feedback have really helped my thesis improve. I am also grateful to all the advice they have provided during the years.

- The external committee member, Dr. Richard Segal, for spending time on my thesis, and providing me with detailed and in-depth comments and interesting questions.

- Dr. D. Sculley, for providing valuable suggestions on Chapter 4 of this work.

- Fellow IR/PLG lab members: Ashif Harji, Maheedhar Kolla, John Akinyemi, Robert Warren, and Brad Lushman for all the various discussions we had over the years. I am specially thankful to them for helping me out with debugging the toolkit presented in Section 6.

- My parents, Effat Atefvahid and Hassan Mojdeh, for raising me to be the person that I am and for giving me all the kindness and support that I needed throughout

my many years of studies. Without their selfless attention, I would have never been able to accomplish this.

- My brother Sana Mojdeh. I was really lucky to have him around during the years of my PhD studies. He always amuses me with his diverse personality, both artistically and academically.
- My husband, Amin Mobasher, for always being there for me. There is no word that could describe how much I appreciate his support.

Dedication

... to my husband, my parents, and my brother.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis deals with the problem of learning-based spam filtering with minimal training help from the user. Previous literature shows the superior performance of personal spam filtering when there is extensive training from the user. We investigate ways to improve personal spam filters without putting the burden of labeling email messages on the user.

1.1 Background

The Spam Track at the Text Retrieval Conference (TREC)[1] defines *spam* as "unsolicited, unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient" [Cormack, 2007a]. A huge amount of spam are being generated every day and waste significant Internet resources as well

[1] TREC is a series of workshops on different areas of Information Retrieval

as users' time. It is projected that email traffic would reach 419 billion emails per day, out of which 83% are going to be spam, which translates into 347 billion spam emails each day [rad, 2012]. Spam attacks both the computer and its users. Spam email can contain viruses, keyloggers, phishing attacks and more. These types of malware can compromise a user's sensitive private data by capturing bank account information, usernames and passwords.

A spam filter classifies email messages sent to a user, as accurately as possible, into spam or ham (non-spam). Personal spam filters can classify messages in batch or online modes. In this thesis, we are primarily concerned with the online personal spam filtering process, shown in figure 1.1 [Goodman et al., 2007]. As email messages arrive, the spam filter classifies them as ham, which are put in the inbox, or spam, which are quarantined (i.e. put in the Junk folder). The assumption is that the inbox is read regularly by the user; whereas, the quarantine/Junk/Spam folder is only occasionally searched for legitimate emails. Users may notice misclassification errors by the filter – spam emails in the inbox or ham emails in the quarantine folder - and report them to the learning-based filter. The filter utilizes the feedback to update its internal model, ideally improving future predictive performance. In many email clients, the method of reporting filter errors is cumbersome or absent. In other words, it is unrealistic to assume that the user will promptly and accurately reports errors.

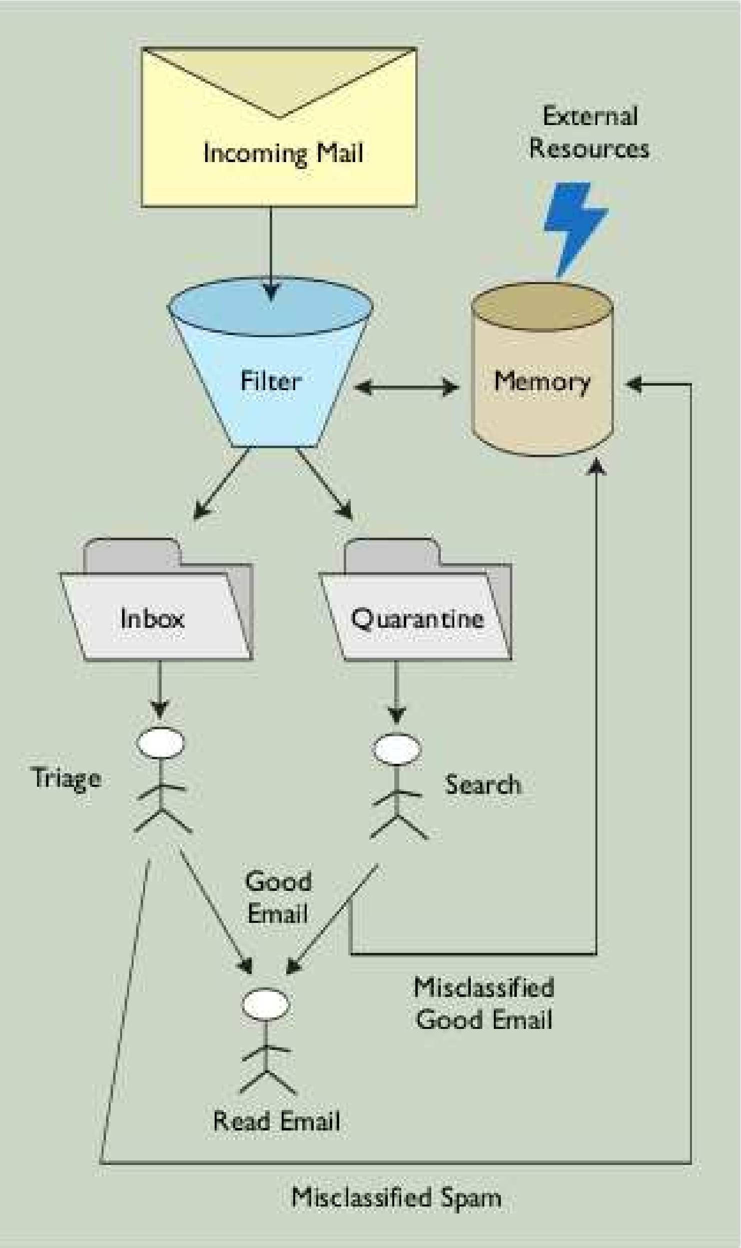


Figure 1.1: Spam Filtering Process

1.2 Motivation

Different spam filters have been proposed, experimented with, and compared. Personal spam filters are evaluated to be the best performing filters; the Spam Track at the Text Retrieval Conference (TREC) [Voorhees, 2007] and the Conference on Email and Anti Spam (CEAS) [cea, 2008a] showed that, in a laboratory setting, personal spam filters perform the best when trained on the ground truth labels of emails. However, this puts a great burden on users to label their emails. In fact, regular email users are very unlikely to cooperate with the spam filter. For example, instead of reporting it as spam, users would rather delete a false negative - that is, spam appearing in the inbox as a legitimate email. Asking users to check Junk folders to find false positives is out of the question.

Another problem for personal spam filters is when a new email client is installed on a machine and the embedded spam filter is asked to classify emails. Without labeled training data, the personal filter has difficulty performing its best, and realizing the user expectations. Reducing the interaction between the user and the spam filter without degrading the filter performance would really help this situation.

1.3 Problem Statement

In this thesis, we investigate ways to improve personal spam filters with the least amount of interaction with the user. Regular users would benefit from a filter which does not rely on their feedback for training purposes. Our findings also help the case of the new email client installation; so the user would not have to label a whole set of emails before getting the spam filter to operate sensibly.

We tackled the problem from different perspectives. Here is the list of questions we addressed, how we approached them, and the results we achieved.

How well can a personal spam filter learn from older emails or other users? In order to minimize user input we need to find other alternatives to provide the needed training data and possible options are user's older emails and emails belonging to other users. We tested well-known techniques, where the training and test sets belong to different users, to evaluate the effectiveness of inter-user training. We did not see any improvement of inter-user learning over intra-user training. We also evaluate the case where the training and test sets belong to the same user, but contain messages from different time periods, that is the case when the filter learns from a set of older messages and is asked to classify more recent messages, and show that well-performing personal spam filters are not helpful in this case. In the third setting, we evaluate spam filtering when the training set belongs to other users, from older time-frames. That is basically a cross-corpus experiment, where training and test sets belong to two different corpora.

Can personal spam filters learn from, as few as, two labeled examples? The user will not have to label all emails, if the spam filter is able to classify the whole inbox by using the labels of only the first few email messages. We present an aggressive graph-based, semi-supervised method that classifies email messages with only a handful of training examples. The aggressive method is given the labeled first few messages as the training set, and the rest of the inbox as the test set. We show our method compares favorably with others when trained on as few as two examples.

Can we design a personal filter that does not ask the user to label any email messages?

A personal autonomous spam filter is a personal filter that does not ask the user to provide labeled email messages for training. We build such a filter by having a personal learning filter trained *only* on the output of a global filter. We evaluate a combination of autonomous filters in extensive experiments and show that a personal spam filter trained on the result of a global webmail is among the best.

Can we design a "privacy-preserving" environment for evaluation of spam filters?

Preserving the privacy of users is usually at odds with conducting user studies, especially when it comes to email messages; users are very protective about their emails. We design and implement a toolkit that allows researchers to perform user studies on email messages without violating the user's privacy. We test the filter in a preliminary experiment.

Can we validate the performance of the autonomous filter by conducting a privacy preserved user study?

We evaluate the autonomous spam filter in a user study utilizing the toolkit. The results confirm our previous findings and show that an online personal filter, trained exclusively on the output of a commercial web-based global spam filter, improves the global filter.

1.4 Contributions

The aim of this thesis is to improve personal spam filtering with the least amount of interaction with the user by answering the above questions. In the following, we briefly summarize the main contributions of this thesis in each chapter.

Chapter 3: Personal Spam Filtering with No Recent Training Data

This chapter investigates how well a personal spam filter learns from other sources of data, as opposed to a user's own messages. The idea of using other sources of information for training is to build a personal spam filter with no reliance on user feedback. The simplest source of information for a personal filter is a set of messages from an older time frame than that of the messages filter is asked to classify. The second source of information that can be used to train a personal spam filter is messages belonging to other users. Another source of information that can potentially be used by a personal spam filter is the combination of the previous two sources.

We used the same datasets as ECML/PKDD Challenge [ecm, 2006] since they matched our problem setting (the training and test sets were drawn from different sources). Also, the best-performing spam filters at the challenge used variant of semi-supervised learning methods. This led us to believe that semi-supervised variants of the best-performing filters at TREC would also give us a better performance over the delayed, cross-user, and cross-corpus datasets drawn from the TREC Spam Track. However, the experiments led us to realize that semi-supervised variants of the best-performing filters in TREC could not reproduce the results of ECML/PKDD and help improve the delayed or cross-user filtering process. Surprisingly, only Transductive

Support Vector Machine (SVM) [Vapnik, 1998] showed better performance over the fully-supervised SVM [Mojdeh and Cormack, 2008b] and that was only when the training and test sets were of a completely different corpus, meaning they were coming from different time frames and belonging to different users [Mojdeh and Cormack, 2008b].

Chapter 4: Personal Spam Filtering with Very Few Training Examples

In this chapter, we investigate a personal spam filter that performs well when there are only few labeled examples available for training purposes and a huge number of unlabeled messages for classification. This can help spam filters perform better when the user is not willing to, or, cannot provide the filter with many training examples, for example, when a new email client is installed on the user's machine. To solve this problems, we adapted and modified a semi-supervised learning algorithm, called Aggressive Consistency Learning Method, based on Global and Local Consistency Learning method of Zhou et al. [Zhou et al., 2003]. To find the most informative terms, Singular Value Decomposition (SVD) [Alter et al., 2000] is applied on the data. We show that the proposed method compares favorably with the competing methods when the number of training examples is, as few as, two [Mojdeh and Cormack, 2010].

Chapter 5: Autonomous Personal Spam Filtering with No User Feedback

In this chapter, the difference between personal and global spam filters is discussed with the intention to design an autonomous personal spam filter that does not rely on a user for feedback. There has been debate on whether personal or global spam filters perform better. The general assumption is that global filters have access to a much broader source of information, and therefore, should perform better. However, several papers indicate the opposite: the best personal filters, of which we are aware, are more accurate than the best global ones. We should emphasize that personal spam filters yield remarkably high accuracy when they are trained on the ground truth labels of emails. Typically, the user interface solicits feedback only for misclassified messages, or for a small subset of hard-to-classify messages. Therefore, the assumption of ideal feedback is not realistic. Also, the same filter might perform very poorly with noisy feedback.

The key idea behind this chapter is to design a personalized spam filter and train it on the result of a global filter, with no user labeling at all. The motivation for this idea is that the global filter's error rate is comparable to the feedback error rate by a typical user of personal filters. To investigate this hypothesis, we conducted many experiments with generative and discriminative filters. Our experiments overwhelming support the hypothesis that a personal filter trained on the results of a global filter exceed the performance of two global filters. We also found that discriminative filters do not work well as autonomous filters, even with noise-tolerant parameter settings. This result is not surprising, as these filters should be able to isolate features that are highly correlated with the global filter's errors, and therefore, effectively reproduce the filters incorrect behavior [Cormack and Mojdeh, 2009a].

Chapter 6: A Toolkit for Privacy-Preserving Spam Filter Evaluation

This chapter introduces a new approach to evaluate spam filters in more genuine settings. The main objective is to provide a reliable and convenient way to assess filters written in TREC format on a real user mailbox, without asking users to donate their email. We present a privacy-preserving toolkit that is developed to perform user studies for the evaluation of spam filters. The toolkit is an extension to the Thunderbird mail client and supports a standardized evaluation of multiple spam filters on private mail streams. The main advantage of the extension is that researchers need not view or handle the subject users' messages. In addition, subject users need not be familiar with spam filter evaluation methodology. We use this extension in the next chapter to evaluate our proposed autonomous spam filter on real users' email.

The extension evaluates a spam filter, assuming the user's existing classification is accurate, and sends only the summary results to the researcher, after allowing the user to verify exactly what will be sent. This plugin addresses an outstanding challenge in spam filtering evaluation which is using a broad base of realistic data while satisfying personal and legislative privacy requirements. This chapter ends with preliminary results utilizing the tool to evaluate some filters previously evaluated at TREC [Mojdeh and Cormack, 2008a].

Chapter 7: User Study

This chapter examines the hypothesis that it is possible to build an autonomous personal spam filter that does not rely on user feedback, in a retrospective user study. The user study follows our proposed privacy-preserving approach that, to the best of our knowledge, is the first in this field. We show that the performance of an industry leading web mail system (Gmail) is improved when a personal learning-based spam filter is trained on the results of Gmail as the ground truth labels of the emails.

This chapter, first presents the design of the user study, which involved recruiting users, and running a personal filter on their emails. Users were asked to adjudicate the relative effectiveness of Gmail's filter and the best-performing autonomous filter, identified by the previous experiments in Chapter 5. The results of the experiment with statistical analysis on users' emails are also discussed which support the hypothesis.

Chapter 2

Background

The task of email spam filtering — automatically removing unwanted, harmful, or offensive email messages before they are delivered to a user — is an important, large scale application area for machine-learning methods. These techniques have an idealized assumption that a learning-based filter will be given perfectly accurate label feedback for every message that the filter encounters. In practice, label feedback is provided sporadically by users, and is far from perfectly accurate. In this thesis, we are determined to find the most effective way to eliminate spam and minimize the risk of eliminating real mail, while minimizing user feedback.

This thesis shows that high-level spam filtering performance are achievable in settings that are more realistic than the idealized scenario. These settings include scenarios in which training and test sets belong to different users or time frames, in which users are willing to label just a few examples, or in which in a privacy preserving environment personal filters use global filter's output as a source of feedback.

The introductory chapter will present an overview of how spam is filtered and the scope and scale of the spam filtering problem. We also investigate several categories of spam filters and machine-learning methods that have performed well.

2.1 How Does a Spam Filter Work?

The never-ending battle between spammers and spam filtering technology is the foundation of the fast evolution of spam filtering over classical text categorization [Sebastiani, 2002]. Filters need to adapt based on the information that they receive continuously. Spam filters can rely on different sources of information: The content of the email is a primary source of information used by many filters. In this case, filters may use the entire content of the message [Cormack and Mojdeh, 2009b] or select certain features through the feature selection process [Kolcz et al., 2004, Cormack et al., 2007]. A second source of information for spam filters is any system related information, such as time of the delivery of the message, or information in the header of the packet that delivered email. This type of information is used heavily by so-called network level spam filters which use TCP/IP headers to find spam messages [Golbeck and Hendler, 2004, Ramachandran and Feamster, 2006]. A third source of information for any spam filter can be the feedback provided by the user. Spam classifiers generally take the user's feedback as the correct classification of the message, and use it for training purposes. There are two models for user feedback: it can be given to the filter either immediately after the filter classifies the message, or in a batch mode.

Based on the model of user feedback, the spam filtering can be categorized into *batch* classification or *on-line* classification.

In *batch filtering*, the filter is presented with a set of labeled examples, and once the training process is done, the messages to be classified (the test set) are given to the filter for classification. In *online filtering*, as shown in Figure 1.1, the filter is presented with a stream of messages. After the filter classifies each message and puts it into the inbox or junk folder, the correct label of the message is available to the filter for training purposes after which the next message in the stream is given to the filter for classification. In the realistic environment of an email client, the process of revealing the correct label to the filter takes place by the filter assuming that its classification of the email message is correct, unless the user reports the email as misclassified. In this thesis, we are mostly concerned with the on-line spam filtering process.

Messages can be presented to filters in different formats. In the basic case, the filter accesses the messages in their raw format. Some spam filters deal with fingerprints of messages; in collaborative spam filtering, a database of email fingerprints is maintained to which new messages are compared with. In other cases, only some features are selected from each message and presented to the filter. Sculley believes that choosing an appropriate feature selection can have more impact on the quality of results than the choice of learning method [Sculley and Brodley, 2006]. Kolcz et al. study the impact of feature selection and duplicate detection on spam filtering [Kolcz et al., 2004] and also filtering using Naive Bayes [Kolcz, 2005].

Spam filters can also be categorized into global or personal spam filters. Spam filters can be trained globally, where a global model is learned for all users, or personally, where a separate model is learned for each user [Segal, 2007]. In the literature, a globally-trained, anti-spam filter is referred to as a filter that is trained on spam and ham email from a large collection of users, using a wide variety of data sources (possibly including spam-trap data, ham-trap data, and customer-labeled data) to create a globally-learned anti-spam model that is shared by all customers. BirghtMail [bri, 2011] and IBM E-mail Security [IBM, 2011] are examples of such global filters.

Segal shows that global spam filters outperform personal spam filters for both a small and large number of users [Segal, 2007]. The filter used is a variation of Naive Bayes. The observation is that much of the personal data is retained allowing the global filter to benefit from individual preferences. There is also an algorithm, based on a Naive Bayes classifier, proposed to combine personal and global data. The algorithm makes use of both global and personal information to classify each incoming message by having some trade-off between global and personal data made at the word or feature level. The new algorithm seems to improve the performance with a diverse set of users.

In this thesis, a global filter is used with a similar, but slightly different, definition. Here, we define a *global* filter as a filter that does not use any labels for the user's messages. More specifically, a global filter employs some combination of handcrafted rules, pattern-based rules, global whitelists and blacklists, and any non-user-specific training examples or collaborative filtering. Spam traps are also a source of information for global filters. A spam trap is an email honeypot and in its simplest form this is an email address that is used to collect spam emails. Various commercial global spam filters use built-in spam trap to automatically keep the spam filter up-to-date [mda, 2012, cle, 2011]. Cleanmail, for instance, uses spam traps to train a Bayes database with incoming spam mails [cle, 2011].

A *personal* filter, on the other hand, *only* relies on labels of the user's messages. Of course, a hybrid personal/global filter can be imagined. Collaborative filtering, for example, can be thought of a hybrid of global and personal spam filtering systems. Chapter 5 describes in more detail a global filter and how it is used to build an autonomous personal spam filter.

The learning methods employed by filters may be broadly characterized as generative or discriminative. A *generative* model attempts to describe the distribution of all the data, as opposed to a *discriminative* model, which would focus on the differences between them. As an example, a generative model for distinguishing birds from mammals would contain a long description of each one, sufficient to draw many different bird-like creatures, or many different mammalian creatures. A discriminative model, on the other hand, would focus on that fact that mammals almost always have hair and always feed their young with milk, while birds almost always have feathers and almost always have wings [Goodman and tau Yih, 2006].

A generative method for spam filtering models the characteristics of ham and spam independently, and classifies a message according to which model fits best. Naive Bayes (NB) is perhaps the best-known generative method [Sahami et al., 1998]; sequential compression methods like Prediction by Partial Matching (PPM) [Cleary and Witten, 1984] and Dynamic Markov Compression (DMC) [Bratko et al., 2006a] are also generative. A discriminative method models only the characteristics that distinguish ham from spam, ignoring the rest. Common discriminative methods are Logistic Regression (LR) [Goodman and tau Yih, 2006] and Support Vector Machines (SVM) [Sculley and Wachman, 2007a]. Details of these methods can be found in Section 2.2.4.

2.1.1 Challenges for Spam Filters

There are some important challenges involved in building and evaluating spam filters. One major challenge for spam filters is obtaining correct labels from users for training purposes. Users must manually go through a stream of emails to label them, or to look for potential spam emails in the inbox (false negatives), or legitimate emails in the Spam or Junk folder (false positives). Users may find this procedure burdensome as the sizes of folders grow. Email clients have not made this any easier for users. Reporting errors is not always straightforward in email clients and moreover, users are not generally encouraged to provide feedback.

One major problem in filtering spam is the presence of noise in class-labels; when users accidentally or deliberately incorrectly label a message. When explicitly asked to classify messages, human subjects have been reported to exhibit error rates of 3%-7% [Graham-Cumming, 2006b]. Noise in labels can be due to users being fooled by a scam email, to the point that email lottery scams have been reported as ham, or misunderstanding the feedback mechanism. This is a general issue of assessed error in all information retrieval areas [Bailey et al., 2008].

Grey emails, for example, are one source of noisy labeled data. A grey email is a message that could reasonably be considered either spam or ham by different email users [Chang et al., 2008]. Two-thirds of users consider unsolicited commercial emails from senders with whom they have done business to be legitimate, and the remaining think of them as spam. Emails in campaigns are good examples of grey email: They might have been solicited by only a fraction of users, who consider them ham, and not been solicited by the others, and therefore, considered spam. A recent paper studying grey mail cites that one sample of 418 messages contained 163 grey mail messages – nearly 40% [Yin et al., 2007]. The same study compares four simple methods of detecting grey emails among which treating email campaigns with different labels as grey mail is the most reliable one. The study also shows that a global spam filter can benefit from grey mail detection, even when the detector is imperfect.

Class label noise significantly decreases the expected performance of spam filters, particularly those that perform best on noise free data. Even uniformly distributed noise on labels can significantly degrade performance of spam filters [Sculley and Cormack, 2008]. Same literature has proposed modifications to spam filters to make them more robust to label noise, such as less aggressive updates, label cleaning, label correcting, and various forms of regularization. It is also shown that natural noise from real users is more challenging than uniform noise and that uniform class label noise is easier to filter than label noise.

Cormack and Kolcz, however, question this conclusion and show results that are consistent with the hypothesis that the filters perform about as well on both random and natural label noise [Cormack and Kolcz, 2009]. They argue that previous reported results are due to evaluation errors, and propose to use a fully automatic method to fuse the results of candidate filters and get a pseudo-gold labeling. These labels are used as ground truth in evaluating spam filters, and exhibit a lower error rate than labels obtained from natural sources.

The nature of class noise in the spam filtering is investigated by Kolcz and Cormack [Kolcz and Cormack, 2009]. They show that label noise has a clear content-based bias, with certain genres of email being much more likely to confuse than others. The experiments demonstrate that email messages that spam filters find confusing correspond quite closely to messages that human assessors are likely to find confusing.

2.2 How is a spam filter implemented?

2.2.1 Handcrafted Spam Filtering

A handcrafted classifier can be implemented by a person changing the filter, for example, adding a rule that all emails containing the term Viagra are spam. These rules can be updated automatically. With handcrafted spam filters, there is usually no content-based learning involved. These filter methods can be divided into whitelists, blacklists or greylists, challenge response, and methods that combine any number of these approaches. In the following section, each of these techniques will be described along with the most recent and the best methods of applying each.

Whitelists and Blacklists

Whitelists and blacklists use the name, email address, and IP address of the sender to classify spam. These lists can be manually or automatically updated, and can be personal or shared. Shared blacklists may add signatures of entire spam messages so that the messages are blocked if it is received by others [Cormack, 2008]. Research on IP blacklisting focuses mostly on detecting spam-sending machines based on senders IP addresses [Dietrich and Rossow, 2008] or the behaviour of the suspecte bots [Ramachandran et al., 2006]. Ramachandran et al. studied the network-level behaviours of spammers and found the majority of spam messages were sent from a few, concentrated portions of IP address space [Ramachandran and Feamster, 2006]. The use of IP addresses for filtering spam has also been studies by Wei et al. with the assumption that current domain blacklisting may not be effective if spammers regularly replace blacklisted domains with new registered domains [Wei et al., 2010].

Esquivel et al. introduce a network-level spam filtering by having SMTP servers compute light-weight signatures for spammers and then updating the network routers about the signatures [Esquivel et al., 2009]. The idea is that routers would immediately drop connections that match a stored TCP fingerprint signature, therefore reducing the costly traffic of spam. In addition to the IP-level filtering, the transport-level characteristics of spam have also been studied. SpamFlow is a spam classifier that exploits the fundamental characteristic of spam traffic showing relative discriminatory strength of different flow properties such as the congestion window and Round-Trip Time (RTT) [Beverly and Sollins, 2008].

Erickson et al. study the effectiveness of whitelisting [Erickson et al., 2008] where a whitelisting email service is built and deployed with a Thunderbird add-on to try to make whitelists easy to manage [DOE, 2008]. The results indicate 0% false negatives and less than 1% false positives (that is, ratio of legitimate emails being misclassified as spam) using a challenge-response method to identify whether or not unknown senders are legitimate.

Greylisting and Challenge Response Methods

Greylisting is a combination of whitelisting and the temporary deferring of messages from sources not on the whitelist. In greylisting, emails are returned with a "soft" error code to the sender's mail delivery system. The assumption is if the email is legitimate it is more likely to be resent. Greylisting implementations are fairly lightweight and may even decrease network traffic as well as processor load on the mailserver. A simple greylisting implementation in Perl is also available [gre, 2003].

With greylisting, there is a chance that legitimate emails get lost. Also, in order for the spam to be delivered, spammers need only to retransmit the message. There also seems to be a delay in delivering messages using greylisting, however in a case study of greylisting methods on real SMTP servers, measurements do not demonstrate any loss of efficiency [Sochor, 2010].

In a challenge-response method, a message not on the whitelist will be kept in quarantine and an authentication message is sent to the sender which may ask the sender for a response or registration with an online system. The challenge response system is an intrusive mechanism that imposes delay on delivering emails, even the legitimate ones, and there is always a chance for deadlock - that is when everybody employs a challenge response method. More importantly, since most spam messages use forged sender addresses, challenges are usually sent to third parties causing even more spam [Lynam, 2009].

Ad-hoc Methods

Ad hoc filtering is implemented by ad-hoc rules added by the system administrator to block certain kind of spam. Most email servers implement these rules by performing simple pattern matching. *Rule-based* methods are generally considered the same as ad hoc methods, except that rule-based methods usually have more general and sharable rules, defined in a more formal notation. A good example of a rule-based spam filtering is SpamAssassin [spamassassin.org, 2005].

In SpamAssassin, most rules are based on Perl "regular expressions" that are matched against the body or header fields of the message. SpamAssassin also employs a number of other spam-fighting techniques. Rules are called tests in SpamAssassin. A message is classified as spam if the sum of the scores of the triggered tests is greater than a certain configurable threshold.

Hayati et al. describe a rule-based combinatorial approach [Hayati et al., 2010] that automatically distinguishes between spambots and human users, based on their usage patterns. This technique uses web usage behaviour to investigate spambots behaviour on the web and proposes a new rule-based classifier using a tree structure for fast and on-the-fly spambot detection which gains on average accuracy of 93% on spambot detection.

2.2.2 Learning-based Spam Filtering

There are many research articles on learning-based classification techniques in the spam filtering research community. Well-researched methods for text classification include Naive Bayes classifiers [Graham, 2004, Graham-Cumming, 2005, Robinson, 2002, Robinson, 2004], perceptron [Freund and Schapire, 1999], Dynamic Markov Compression techniques [Cormack and Horspool, 1987], Support Vector Machines (SVM) [Vapnik, 1998, Joachims, 1998], and clustering methods such as nearest neighbour [Graham, 2004, Graham-Cumming, 2005].

One of the de-facto standard techniques for practical, learning-based spam filtering, such as SpamAssassin or Mozilla Mail, is "Bayesian" spam filtering [Graham, 2004, Graham-Cumming, 2005, Robinson, 2002, Robinson, 2004]. In these techniques, each message is decomposed into individual tokens or words. By examining labeled messages, the fraction of spam and the fraction of ham messages containing each word or token is computed. 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.

The Perceptron algorithm [Freund and Schapire, 1999] learns a set of weights that defines a hyperplane, separating messages into ham or spam. This algorithm is an online learner using a 0-1 loss function, updating its hypothesis by iteratively attempting to correct all errors by incrementing or decrementing weights for incorrectly classified items. The perceptron is simple and incremental, making it useful for spam filtering. Yet, in practice, it has been found to give poor classification performance in the presence of noise [Khardon and Wachman, 2007].

Support Vector Machine (SVM) is a learning method that is mostly used in classification and regression [Vapnik, 1998]. It has been shown that SVMs yield state-of-the-art performance on text classification [Joachims, 1998]. In the binary classification setting, input is usually a set of data points, each belonging to one of the two possible classes. SVM finds the maximum margin hyperplane which has the largest distance to the nearest data points of any class. This distance is called margin and shows the generalization error of the classifier. Test examples are then mapped to the same space, and labels are predicted based on which side of the hyperplane they are mapped to. A description of how SVM works can be found in Section 2.2.4.

Another filter that is frequently used in text classification is the Dynamic Markov Compression (DMC) based technique, which was first introduced in by Cormack and Horspool [Cormack and Horspool, 1987]. The goal of adaptive data compression algorithms, such as the DMC classifier, is to estimate the probabilities of message generation as closely as possible, in order to enable efficient compression. In general, learning a distribution over all possible messages is intractable. Therefore, DMC algorithm models an information source with a finite state machine with transition probabilities learned from the message data [Cormack and Horspool, 1987]. The algorithm starts in an initial state and moves to the next state after each bit in the sequence is considered.

Throughout this thesis, we are mostly using variations of Logistic Regression, Support Vector Machine, and compression-based methods. More details on specifics of these classifiers are described in Section 2.2.4.

Learning-based spam filtering techniques can be viewed from a different perspective: they can be a supervised learner, a semi-supervised learner, or an unsupervised learner. This categorization is specifically important for us, since, to solve the problem we raised in the introduction chapter, we will be looking into different semi-supervised learning approaches.

Supervised Learning Methods

Supervised learning is a class of machine-learning techniques that infers a function from a set of labeled examples, being the training data. Naive Bayes (NB) is a classifier that has been studied and can be trained very efficiently in a supervised setting. Despite the underlying independence assumption of NB, many studies show the performance of NB on filtering spam [McCallum and Nigam, 1998, Androutsopoulos et al., 2000, Schneider, 2003, Song et al., 2009, Hovold, 2005, Kolcz and Chowdhury, 2005, Metsis et al., 2006]. The main reason is its relatively small computational cost, for example, compared to Support Vector Machine (SVM) algorithms. A better implementation of NB method [Song et al., 2009] tries to minimize false positive rates. The idea is that the cost of misclassifying a legitimate email as spam is much worse than misclassifying spam emails [Kolcz, 2005]. The work improves NB performance by combining three NB classifiers. The base classifier separates the training instances into two parts and linear programming is applied to optimize the decision thresholds of the second stage classifiers. The last step combines the techniques and improves the performance over classic NB and SVM.

Semi-Supervised Learning Methods

Semi-supervised learning is a class of machine-learning techniques that makes use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). In many machine-learning applications, there is great human effort involved in labeling samples, while obtaining unlabeled data is fairly simple. Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with labeled data, to build better classifiers. Because semi-supervised learning requires less human effort and gives higher accuracy, it is of great interest, both in theory and in practice.

Semi-supervised machine-learning methods are of special interest for spam filtering, as there are usually fewer training examples available. During the initial deployment of spam filters, for instance, a user may be willing to provide only a few labeled examples for training, but will expect correct classification of a large number of emails. Another application is personalized spam filtering with low labelling cost, using per-user, semi-supervised filters with few labeled examples to augment a global filter. Among the most popular semi-supervised learning methods are self-training, co-training, transductive Support Vector Machines, and graph-based methods. Zhe offers a complete survey of semi-supervised methods and their applications [Zhu, 2007].

In *transductive* learning, as opposed to inductive learning, the data to be classified are known at the time of classifier construction [Joachims, 1999]. The output of the classifier is geared towards the specific test data and has to be reconstructed for new data. In the *inductive* approach, the labeled data is used to train a supervised learning algorithm and then have it predict the labels for all the unlabeled points. Transductive approaches have the advantage of being able to consider all the labeled and unlabeled data. Therefore, these algorithms usually make better predictions with fewer labeled examples [Joachims, 1999]. Transductive Support Vector Machines (TSVMs) are used for text classification [Joachims, 1999] and for spam filtering [Zhou et al., 2007]. TSVMs are considered inductive learners because the resulting classifiers are defined over the whole space. The name TSVM originates from the intention to work only on the observed data (though people use them for induction), which, according to Vapnik [Vapnik, 1998] is solving a simpler problem.

In Dynamic Markov modeling techniques, the data is modeled by a Markov chain. If such a model can be constructed from the first part of data, it can be used to predict the forthcoming part of the data [Cormack and Horspool, 1987]. A semi-supervised variant of Dynamic Markov Modeling [Cormack, 2006a] which makes use of unlabeled training data ranked first in Task B of ECML/PKDD Discovery Challenge [ecm, 2006]. It improves an initial discriminative classifier derived from labeled data by iterative labeling. The classifier first applies logistic regression on data to find tentative labels and then Dynamic Markov Compression (DMC) is applied online, resulting in new sequences of labels and this process is repeated five times. Since the provided data challenge had a relatively few training data compared to the test data, the best performing submissions applied semi-supervised learning techniques.

A graph-based semi-supervised learning method is a transductive learner. We describe the application of graph-based learning methods to spam filtering in chapter 4, where we propose a graph-based algorithm, based on Local and Global Consistency (LLGC) [Zhou et al., 2003], that learns from very few training examples.

Unsupervised Learning Methods

Unsupervised learning is a class of machine-learning techniques for which there is no labeled data to learn from. Almost all work in unsupervised learning can be viewed in terms of learning a probabilistic model of the data [Ghahramani, 2004]. Many unsupervised learning methods are based on data mining methods used to pre-process data.

SpamCampaignAssassin is an online unsupervised spam learning and detection scheme [Qian et al., 2010]. It identifies campaigns online, generates campaign signatures, and detects spam based on campaign signatures. A spam campaign is a group of highly similar spam email messages reported by a large number of users. The assumption is that the majority of email is spam; spam emails usually belong to a campaign; and spam is created by obfuscating some parts of an email, while leaving the rest intact [Pitsillidis et al., 2010]. The claimed accuracy is comparable to the de-facto supervised spam filters.

Online Learning Methods

The methods discussed so far fall into the category of batch learning, where the training and the test data are static and are provided before the classifier is built. Online learning is the other model of induction where the learner learns one instance at a time and the correct label of the instance is available to the learner right after. The main online learning algorithms are variants of Logistic Regression (LR) [Goodman and tau Yih, 2006], Dynamic Markov Compression (DMC) [Bratko et al., 2006a], Naive Bayes, and online Support Vector Machines (SVM) [Sculley and Wachman, 2007a].

In LR [Goodman and tau Yih, 2006], each message is classified and a single gradient descent step with learning is taken for each feedback message to train the filter. The DMC online learning method [Bratko et al., 2006a] employs Dynamic Markov Compression introduced by Cormack et al. [Cormack and Horspool, 1987]. Classification is done by first building two compression models from the training corpus, one from examples of spam and one from legitimate email. The compression rate achieved using these two models on the target message determines the classification outcome. Relaxed Online SVM (ROSVM) has been studied [Sculley and Wachman, 2007a] and, with some parameter tuning, performs better in many spam filtering tasks.

Active Learning Methods

Another class of machine-learning techniques is active learning where the learner is responsible for asking for the labels of data. The learner is often given a quota of the number of labels it can ask for. Active learners face the same issue as the semi-supervised learners; that is, labeled examples are scarce.

Two main categories of active learners are *pool-based* and *online*. Pool-based active learning assumes that the learner has access to a pool of n unlabeled examples, and is able to request labels for up to $m << n$ examples on each labeling round. In the online active learning framework, messages come to the filter in a stream and the filter must classify them one by one. At each point, the filter has the option of requesting a label for the given message, and the goal for the filter is to achieve strong classification performance with as few label requests as possible and to stay within its assigned quota.

Sculley explores several approaches to determine when to request labels for new examples [Sculley, 2007a]. These methods improve the results over uniform subsampling and over prior pool-based active learning methods, reducing the number of labels needed to achieve high performance with negligible computational cost.

Segal et al. propose using a pool-based active learning approach to build a large labeled corpus of emails [Segal et al., 2006]. In this method, an approximate uncertainty sampling method is applied to overcome the complexity issue of the original uncertainty sampling algorithm where in each round, the leaner has to go through all unlabeled messages to find the subset of messages that it is the most uncertain about. In the proposed approximate uncertainty sampling, the uncertainty of each messages is recorded and only re-evaluated for a subset of examples in each round. This way, the complexity of the uncertainty algorithm is much reduced. It is also shown that both approximate uncertainty sampling and the original uncertainty sampling converge to nearly identical error rates after a certain number of queries.

Active learning with clustering is among the most recent works on the use of clustering by Whissel and Clarke [Whissell and Clarke, 2011]. It first builds clusters from a random sample of the training data, then asks users to judge one message from each cluster. Next, a classifier is used to classify test data using the judged messages as the training set. The main contribution of this algorithm is the relatively high performance with few training examples.

Combined Methods

Many spam filters combine different methods in order to form a classifier. SpamGuru [Segal et al., 2004] is a server-side spam filter that utilizes different approaches to build a very customizable spam filter. The infrastructure it offers, adheres to three principles: (i) Simple administration by automation of different tasks such as maintaining black- and whitelists. (ii) Customizable by both system administrators and end users, and (iii) Offers a combination of classifiers and tokenization techniques that could be plugged-in and tuned in order to achieve a more robust filtering technique.

There are a number of interesting features in SpamGuru. First, there is a user friendly process called voting, in which users can label messages as spam or ham. Second, there is a folder called "borderline", which is designed to contain a small set of messages which the anti-spam server is relatively uncertain. Since this folder also contains borderline ham messages, the user is more inclined to scan it for possible false positives. Analysis of outgoing email is another feature of SpamGuru which helps build better global and personal whitelists. At the end, the filter assigns a score between 0 and 1000 to each message, with the higher score showing higher spamminess of the message. SpamGuru allows four options for each message classified as spam: delete, archive, send to a challenge queue that provides a challenge-response verification of sender identity, or label as probable spam for the user to take care of.

SpamGuru could be called a generic spam filter since it contains a pipeline of modules, each of which assigns a score to the message and passes it to the next module; the last module being an aggregator module which is responsible for combining all the scores. The version of SpamGuru participating at TREC 2005 [Segal, 2005], has three modules. The first module is a variant of Naive Bayes, which relaxes the assumption of independence. The second module is an SMTP path analysis. This module only looks into the "received" line of the message and filters based on the IP address sequence, so has to deal with spoofing and dynamic IP address issues. The third module is a linear classifier than aggregates the scores of the two previous classifiers. With the TREC 2005 datasets, SpamGuru performs well with false positive rates at or below 0.01%, Segal et al. argue that false positive rates above this is unacceptable to most users.

Reporter reputation-based filtering was first introduced by Prakash et al. [Prakash and O'Donnell, 2005] and later by Zheleva et al. [Zheleva et al., 2008], where a version of a reporter reputation-based filter was used in conjunction with existing spam filtering techniques. A reporter based system generally uses two types of reports:

This Is Spam (TIS) and This Is Not Spam (TINS). The former is used when a user reports spam in the inbox (false negative) and the latter is used when the user reports a misclassified email in the Junk folder (false positive). The proposed framework has a trust-maintenance component for users, based on their spam-reporting behaviour, and allows users to gain or lose reputation. The main advantage of reporter-based systems is reduction of costs by identifying spam campaigns at earlier stages.

SpamAssassin, as discussed before, is another good example of a filter that combines different spam filtering techniques in addition to ad-hoc rules. SpamAssassin provides a semi-supervised learning mechanism, called "auto-learning" (see section 2.2.2) which automatically re-trains the Naive Bayes classifier using some unlabeled emails collected during filter operation. It works based on self training in which training examples are ones with scores higher than a threshold. SpamAssassin can also update the Naive Bayes classifier when a user gives feedback on the label of any email.

SpamAssassin can also be trained using an active semi-supervised learning method [Xu et al., 2009]. The proposed method clusters unlabeled messages, queries the label of one email per cluster, and propagates such label to the most similar emails of the same cluster. Different approaches of active learning, such as random and uncertainty selection, and a variety of parameter settings are experimented and compared [Xu et al., 2009].

2.2.3 Collaborative Spam Filtering

The idea behind collaborative spam filtering is that each message is sent to a number of recipients. A certain message could have been received and classified or judged by another user. Collaborative spam filtering is the process of capturing, recording, and querying those early judgments. The false positive and false negative rates of this approach depend on both human and technical factors which limit the timeliness, completeness, and accuracy of these three steps [Cormack, 2008].

A collaborative spam filter must capture spam and be able to recognize duplicates over a large number of systems. Preserving the privacy of users is a challenging issue in collaborative spam filtering. 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 is as effective as its completeness [Lynam, 2009].

Due to privacy, messages need to be encrypted which, in turn, limits the ability to find near duplicates. Fingerprinting is a common technique in collaborative filtering in which a database of email signatures is maintained and each new message is compared with the database for a match. I-Match is a matching technique proposed in previous literature [Kolcz et al., 2004, Kolcz and Chowdhury, 2007]. A more recent study that addresses the privacy issue in collaborative spam filtering proposes a finger-print generation technique, called feature-preserving transformation, that captures similarity information [Li et al., 2009]. This system also contains a privacy-preserving protocol, designed to control the amount of information to be shared among the collaborating entities and the manner in which the sharing is done.

Among content-based collaborative spam filters in the literature is a hashing trick method [Attenberg et al., 2009] which builds a personalized global classifier using hashing-trick. The hashing trick maps each word to a feature vector. It allows the compression of many classifiers into a single finite-sized weight vector and maps the global and all personal classifiers into a single low-dimensional feature space, in which the method trains a single weight vector capturing the individual aspects of each user. A second independent hash function is used to determine if the particular hashed-dimension of a token should be incremented or decremented to make the hashed feature vectors unbiased. It is also proposed to mitigate the influence of consistently malicious or otherwise low-quality labelers by utilizing individual classifiers to discriminate spam.

There is also a collaborative filtering scheme proposed which applies the negative selection to decrease the amount of unwanted random matching between unrelated emails [Sarafijanovic et al., 2008]. The algorithm builds two sets of signatures: the first set contains good signatures from personal emails, and the second set contains signatures from spam in the collaborative manner. When a new email arrives, its signature is first compared to this set; if the test does not pass, then the signature of the newly arrived email is compared against the collaborative set of emails. Spamato [Albrecht et al., 2005], an open source spam filter, is another collaborative spam filtering application.

2.2.4 Filters

In this section, we describe more details of the classifiers we adapted in our experiments throughout the thesis. We use the following notation: In a dataset consisting of n messages, each message i is represented as a feature vector x_i where $1 \leq i \leq n$, with an associated label $y_i \in \{-1, +1\}$ for ham and spam message, respectively.

Logistic Regression

Logistic Regression (LR) is a linear classifier with a gradient descent update model. It has been shown that LR performs very well in spam filtering [Cormack, 2007a]. The same method was applied by Cormack et al. to filter web spam in TREC Web ad hoc task [Cormack et al., 2011].

Like any other filter, the filter has two modes: classification and training [Cormack and Mojdeh, 2009b]. In the classification mode, the probability of message i being spam, $Pr(y_i = 1|x_i)$, is a logistic function of a *score* which is equalized to the inner product of a feature vector, x_i, and a weight vector, β:

$$\Pr(y_i = 1|x_i) = \frac{1}{1 + e^{-score}}, \tag{2.1}$$

where $score =< \beta, x_i >$ and $<, >$ represents the inner product operation.

In our settings, this method examines only the first 3500 bytes of each message, treating each overlapping character 4-gram as a feature [Cormack and Mojdeh, 2009b]. Feature vector x_i is a binary vector, containing one as its j^th element, if the corresponding feature exists in the message, and zero, otherwise. Feature space is reduced to 10^6 dimensions using hashing and ignoring collisions.

In the training mode, the weight vector β, which is initialized to zero, is updated using a gradient descent approach. The way the weight vector is updated involves both the ground truth classification and the probability that the classifier has assigned to the message. So, the farther the misclassified message is to the weight vector, the more it will be updated.

$$\beta = \beta + \delta(c - \Pr(y_i = 1|x_i)).x_i \tag{2.2}$$

Here, parameter c is 1 if the message is spam and 0, if not. Since, in most of our experiments, LR is used in an online manner, the training occurs right after each message has been classified. The parameter δ is the learning rate at which a single gradient descent step is applied and is tuned to 0.004.

Dynamic Markov Compression

Dynamic Markov Compression (DMC) was first introduced by Cormack et al. [Cormack and Horspool, 1987]. The goal of adaptive data compression algorithms, such as DMC classifier, is to estimate the probabilities of message generation as closely as possible, in order to enable efficient compression. In general, learning a distribution over all possible messages is intractable. Therefore, DMC algorithm models an information source with a finite state machine with transition probabilities learned from the message data [Cormack and Horspool, 1987]. The algorithm starts in an initial state and moves to the next state after each bit in the sequence is considered.

In spam filtering, a classifier based on DMC, first, builds two compression models from the training corpus, one from examples of spam, and one from legitimate emails. Next, the compression rates, achieved by compressing the message using the two compression models, are calculated. The message is classified as spam if the compression rate using the spam model is better than the one using the ham model, and vice versa. In other words, the DMC algorithm performs the classification by adding the message to the entire set, using both models to compress it.[1] The model that adds the minimal increase in the description length of the entire corpus is considered the class of the message [Bratko et al., 2006b].

Support Vector Machines

Support Vector Machine (SVM) is a learning method that is mainly used in classification and regression [Vapnik, 1998]. In the binary classification setting, input is usually a set of data points, each belonging to one of the two possible classes. SVM finds the maximum margin hyperplane which has the largest distance to the nearest data points of any class. This distance is called the margin and shows the generalization error of the classifier. Test examples are then mapped to the same space, and labels are predicted based on which side of the hyperplane they are mapped to.

In a more formal definition of SVM, the input consists of n points $x_i \in R^D, 0 \leq i \leq n$, where D is the dimension of the input space, and each point belongs to one of the two classes labeled by $+1$ or -1, i.e. $y_i \in \{-1, +1\}$. In a linearly

[1] It is not necessary to actually compress the messages in order to classify them; learning and applying the probability estimates is sufficient.

separable dataset, the output of SVM is a hyperplane that divides points into two classes, in which the hyperplane has the maximum distance to the closest points in the dataset.

Any hyperplane can be written as $< w, x > -b = 0$, where w is the normal vector of the hyperplane. So, the SVM problem can be formulated as finding w and b such that, for all $1 \leq i \leq n$

$$< w, x_i > -b \geq +1 \; if \quad y_i = +1 \tag{2.3}$$

$$< w, x_i > -b \leq -1 \; if \quad y_i = -1 \tag{2.4}$$

The two equations can be combined as

$$y_i(< w, x_i > +b) \geq +1, \quad 1 \leq i \leq n \tag{2.5}$$

The closest points to the separating hyperplane that form hyperplanes $H1$ and $H2$, as shown in Figure 2.1 [Zhu, 2007], are called *support vectors*. The distance from $H1$ or $H2$ to the separating hyperplane, named $d1$ or $d2$ in Figure 2.1, respectively, is called the margin and is equal to $\frac{1}{||w||}$. Therefore, SVM can be described as an optimization problem as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 \\ s.t. \quad & y_i(< w, x_i > -b) \geq +1, \\ & 1 \leq i \leq n \end{aligned} \tag{2.6}$$

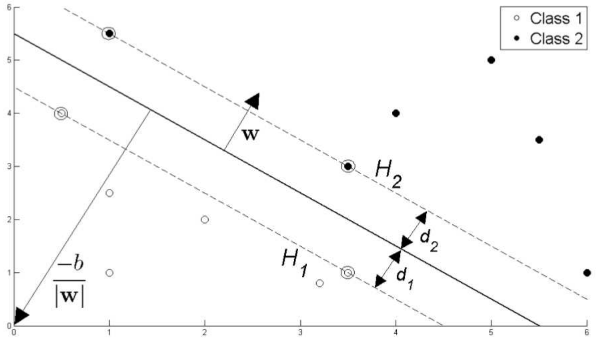


Figure 2.1: Support Vector Machines on Linearly Separable Data

Transductive SVM (TSVM) is the semi-supervised variant of SVM [Joachims, 1999]. Here, the training set also contains unlabeled examples, and the hyperplane should separate both labeled and unlabeled data with maximum margin. As indicated in [Zhu, 2007], TSVMs are in fact inductive learners, because the resulting classifiers are defined over the whole space, and unseen test examples can also be mapped to the same space. Figure 2.2 [Zhu, 2007] shows how TSVM performs over unlabeled data.

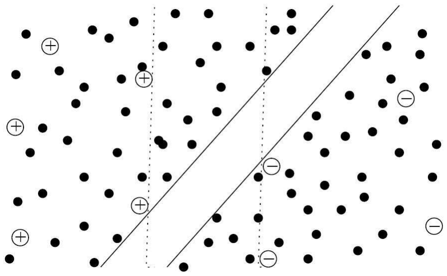


Figure 2.2: Transductive Support Vector Machines on Linearly Separable Labeled and Unlabeled Data

With only the labeled data, plus and minus, the margin boundary would be the dotted lines, however, with unlabeled examples, black circles, the margin boundary is solid lines.

SVMlight [svm, 2008] is an implementation of SVM in C, which we have utilized throughout this work. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently. It provides features such as fast optimization, handles several hundred thousand training examples, and uses sparse vector representation. It also includes an algorithm for approximately training large TSVMs.

2.3 How is a spam filter evaluated?

There are number of issues that need to be addressed when dealing with the evaluation of spam filters.

- The framework in which the spam filters can be evaluated needs to be a setting resembling the real environment in which they are deployed in real life.
- Since spam filters deal with the most private data (emails), preserving privacy of the owner of emails is of the utmost imprtance.
- Datasets on which the spam filters are evaluated are also part of the challenge in the evaluation of spam filters.

- The performance of filters needs to be evaluated using measurements which most accurately capture the actual performance of spam filters.

Next, we discuss the most related and realistic evaluation efforts on spam filtering methods.

2.3.1 TREC

The Spam Track at the Text Retrieval Conference ran between the years 2005 and 2007, and provided probably the most realistic framework for evaluation of spam filters. In this 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. In this framework, messages were presented to the spam filter for classification in chronological order. In the filtering task, the messages were presented one at time to the filter, and the filter's result was compared to a human adjudicated gold standard. The filter also calculated a *spamminess* score, intended to reflect the likelihood that the classified message was spam, which is the subject of post-hoc ROC analysis, described later in Section 2.3.4. After the filter's classification, 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. Summary statistics were derived from the results, assuming the final gold standard to be ground truth.

The TREC Spam Track modeled four different forms of user feedback: immediate feedback, delayed feedback, partial feedback and active learning. With immediate feedback, the gold standard for each message was communicated to the filter immediately following classification. With delayed feedback, the gold standard was communicated to the filter sometime later (or potentially never). With partial feedback the gold standard for only a subset of email recipients was transmitted to the filter; and with active on-line learning the filter was allowed to request immediate feedback for a certain quota of messages which was considerably smaller than the total number.

Since human adjudication is tedious and also error-prone, a bootstrap method was applied to improve the efficiency and accuracy of the gold standard [Cormack and Lynam, 2007]. The bootstrap method builds an initial set of gold standard G_0. After running a couple of messages through the filter and returning their feedback, the evaluation component reports 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 $G_n = G_{n+1}$.

In the toolkit that was used for TREC Spam Track 2007, filters implemented the following commands [Cormack, 2007a]:

- initialize: created any files or servers necessary for the operation of the filter
- classify message: returned ham/spam classification and spamminess score for message
- train ham message: informed the filter of correct (ham) classification for previously classified message

- train spam message: informed the filter of correct (spam) classification for previously classified message

- finalize: removed any files or servers created by the filter

Two test corpora email messages plus gold standard judgments were used to evaluate subject filters. In 2007, one public corpus (trec07p) was distributed to participants, who ran their filters on the corpora using a toolkit implementing the framework and the four kinds of feedback. One private corpus (MrX3) was not distributed to participants; rather, participants submitted filter implementations that were run, using the toolkit, on the private data. The winning methods in all tasks on both trec07p and MrX3 are variants of SVM [Sculley and Wachman, 2007b], variants of logistic regression and Dynamic Markov Compression.

This public corpus contained all the messages delivered to a particular server from April 8 through July 6, 2007. The server contained many accounts that had fallen into disuse but continued to receive a lot of spam. A number of "honeypot" accounts, published on the web and used to sign up for a number of services, were added to these accounts. All messages were adjudicated using the methodology developed for previous spam tracks at TREC [Cormack and Lynam, 2005a]. This corpus is the first TREC public corpus that contained exclusively ham and spam sent to the same server within the same time period. The messages were unaltered except for a few systematic substitutions of names.

TREC provided raw text of messages, each message in a different file. The index file contained one line for each message, with the correct label and the location of the message file. The Spam Track framework, fed each message to the filter. In the immediate feedback task, the correct label of the message was revealed to the filter right after it classified the message. The TREC 2007 public corpus consisted of a total of 75,419 messages with 25,220 ham and 50,199 spam messages [Cormack, 2007a] .

2.3.2 CEAS

Conference on Email and Anti-Spam organized a spam challenge in the years 2007 and 2008. The challenge was a follow up on the Spam Track at TREC and used the same methodology for evaluation, however, it proposed the first live spam challenge in 2008 in addition to the laboratory experiment [cea, 2008b]. The live spam real-time competition collected and delivered a sequence of email messages over three days to participating filters. The challenge worked at the end, despite some problems with error-based feedback and some intermittent server overload. The challenge went through a major problem by judging 500 CNN virus messages as ham and causing incorrect feedback during competition; however, this was corrected in final scoring. The live competition data consisted of 143K messages, 115K spam and 28K ham. Emails included spam messages from spam traps and donated ham messages, with headers rewritten to look live.

The second task at the CEAS spam challenge was a laboratory evaluation and a continuation of TREC Spam Track [cea, 2008a]. The laboratory experiment was a reprise of live stream plus some private data, simulating the real-time experiment using the same sequence of messages. The order of the messages and the same relative timing of feedback was preserved, feedback errors were also reproduced. Filters could request labels for up to $1K$ messages. Therefore, the results for each filter were directly comparable.

The lab corpus at CEAS consisted of the exact messages used in the CEAS 2008 live spam challenge. The corpus consisted of a sequence of 123,100 messages (23,844 ham and 99,256 spam) with labels indicating the filter result and also the gold standard as adjudicated by CEAS.

2.3.3 ECML/PKDD Discovery Challenge

The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) ran a challenge on spam filtering in 2006. In the ECML/PKDD Discovery Challenge labeled training data was collected from publicly available sources [ecm, 2006]. The evaluation emails came from several inbox folders and from different users. The goal was to correctly classify the messages in each inbox as spam or ham. No labeled training examples from the inbox folders were available, instead, one common set of labeled data was given. The labeled data and the inbox folders are governed by different distributions. A learning algorithm cannot rely only on the labeled data because the bias between training data and the inbox folders hinders learning of a correct classification model for the inboxes. The unlabeled data in the inbox folders need to be used to adapt to their distributions.

In this challenge, the training examples came from the same sources, but their differences were the size of the training sets. Blacklist servers of spamhaus [McMillan, 2006] contributed to 50% of the training set as spam. 40% of the training examples were non-spam emails from the SpamAssassin corpus and 10% of that were non-spam emails sent from about 100 different subscribed to English and German newsletters. Task A and B had 4000 and 100 training examples, respectively. In addition, the raw text of the emails was not provided, instead, emails were in a bag-of-words vector space representation. Attributes were the term frequencies of the words. Words with less than four counts in the dataset were removed, resulting in a dictionary size of about 150,000 words. Moreover, the dataset files were in the sparse data format used by SVMlight [svm, 2008]. Each line represented one email, the first token in each line was the class label (+1=spam; −1=non-spam; 0=unlabeled-evaluation-data). Finally, the tokens following the label information were pairs of word IDs and term frequencies in ascending order of the word IDs.

The competition was split into two tasks: (i) Task A, which inuded many training emails and many unlabeled emails for each user; and (ii) Task B, which included many separate inboxes with little training data and little unlabeled data available. The focus of Task A was on adaptation to the user specific characteristics of the inboxes which is critical for a good classification performance. Task B was looking for a learning algorithm to be generalized over different users in a way that user specific properties were taken into account; however, the data from the other users should enhance the classification performance.

Other Evaluation Frameworks

In addition to the TREC, CEAS and ECML datasets, there are other publicly available corpora for evaluation, the ones most considered by researchers are listed here:

- The public SpamAssassin corpus [spamassassin.org, 2003] is a collection of real email from multiple users and news groups. It was the most realistic corpus until 2004. The ham and spam were drawn from different sources during different time intervals.

- Ling Spam [lin, 2003] is an abstraction of 2,412 ham messages from a mailing list and 481 spam messages from an individual recipient; variants employing stop words and stemming are available.

- PU1, PU2, PU3 and PUA [lin, 2003] have a total of 5,957 messages (3,508 ham and 2,449 spam); each corpus abstracts and also obfuscates email from one recipient, so as to preserve privacy.

2.3.4 Evaluation Measurements

The goal of the measurement is to evaluate how well the spam filter approximates the ground truth classification of emails. Here, we elaborate on the measurement values we use to compare different filters throughout this thesis.

Table 2.1: Contingency Table

		Gold Standard	
		ham	spam
Filter	ham	a	b
	spam	c	d
	Total	n	p

TREC Spam Track Evaluation Measures

There are four possible outcomes for given messages; the contingency table shown in table 2.1 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.

Traditional IR evaluation measures, i.e. spam precision and recall, $\frac{d}{c+d}$ and $\frac{d}{b+d}$ respectively, and some other evaluation measurements have been used in previous literature [Graham, 2002, Raymond et al., 2004, Sahami et al., 1998]. In this thesis, we follow TREC Spam Track evaluation measures which are shown in Table 2.2 [Cormack and Lynam, 2005b].

Here fpr (False Positive Rate) and fnr (False Negative Rate) are the basis for the evaluation of spam filters. As shown in Table 2.2, fpr or *hm (ham misclassification)* is the fraction of ham messages that are misclassified by the filter. fnr or *sm (spam*

Table 2.2: Evaluation Parameters

Parameter	Definition
p	number of spam messages
n	number of ham messages
$tp = d$ (true positive)	number of spam messages correctly classified
$fp = c$ (false positive)	number of ham messages misclassified as spam
$tn = a$ (true negative)	number of ham messages correctly classified
$fn = b$ (false negative)	number of spam messages misclassified as ham
$fpr = \frac{fp}{n} = \frac{c}{a+c}$ (false positive rate)	fraction of ham messages that are misclassified
$fnr = \frac{fn}{p} = \frac{b}{b+d}$ (false negative rate)	fraction of spam messages that are misclassified

misclassification), is the fraction of spam messages that are misclassified by the filter. Similarly, *tpr* (True Positive Rate) is the fraction of spam messages that are classified as spam, and *tnr* (True Negative Rate) is the fraction of ham messages that are classified as ham. It is easily seen that $fnr = 1 - tpr$ and $fpr = 1 - tnr$.

Spam filters are usually identified as either hard or soft classifiers. For hard classifiers, which return a categorical result, the effectiveness is determined by a pair (fpr, fnr). Soft classifiers return a score s that resembles the *spamminess* of the message. These filters operate by comparing that score to a threshold t, reporting spam if $s > t$ and ham if $s \leq t$. Increasing t reduces fpr at the expense of fnr, and vice versa.

In this case, the efficiency of the filter is evaluated by the Receiver Operating Characteristic (ROC) curve which is simply the set of all (fnr_t, fpr_t) achievable for some threshold t, which is basically a plot of false positive and false negative rates as the threshold sweeps through all possible values. The TREC Spam Track uses ROC curves plotted on a logit [2] scale, which is similar.

The Area Under the ROC curve (AUC) provides an estimate of the effectiveness of a soft classifier over all threshold settings. AUC also has a probabilistic interpretation: the probability that a classifier will award a random spam message a higher value of s than a random ham message. Typical AUC values are of the order of 0.999 or greater. For clarity, we usually report $(1 - AUC)\%$, the area above the ROC curve, as a percentage. So $AUC = 0.999$ would be reported as $(1 - AUC)\% = 0.1$.

As an example, Figure 2.3 shows the performance of three filters. The star point shows the performance of filter 1 which has produced only one (fpr, fnr), categorial result. The two curves are ROC curves and show the performance of filter 2 and filter 3, both producing spamminess scores and letting the threshold act like a knob and produce all possible values for (fpr, fnr). As seen in the Figure, the curve at the top left has a better performance, in this case, filter 2.

To evaluate single points, Logistic Average Misclassification rate (LAM) has shown to be useful as an alternative to AUC for summarizing filter evaluations reporting a single (fpr, fnr) pair:

$$LAM = logit^{-1}\left(\frac{logit(fpr) + logit(fnr)}{2}\right), \tag{2.7}$$

[2] $logit(x) = \log \frac{x}{1-x}$

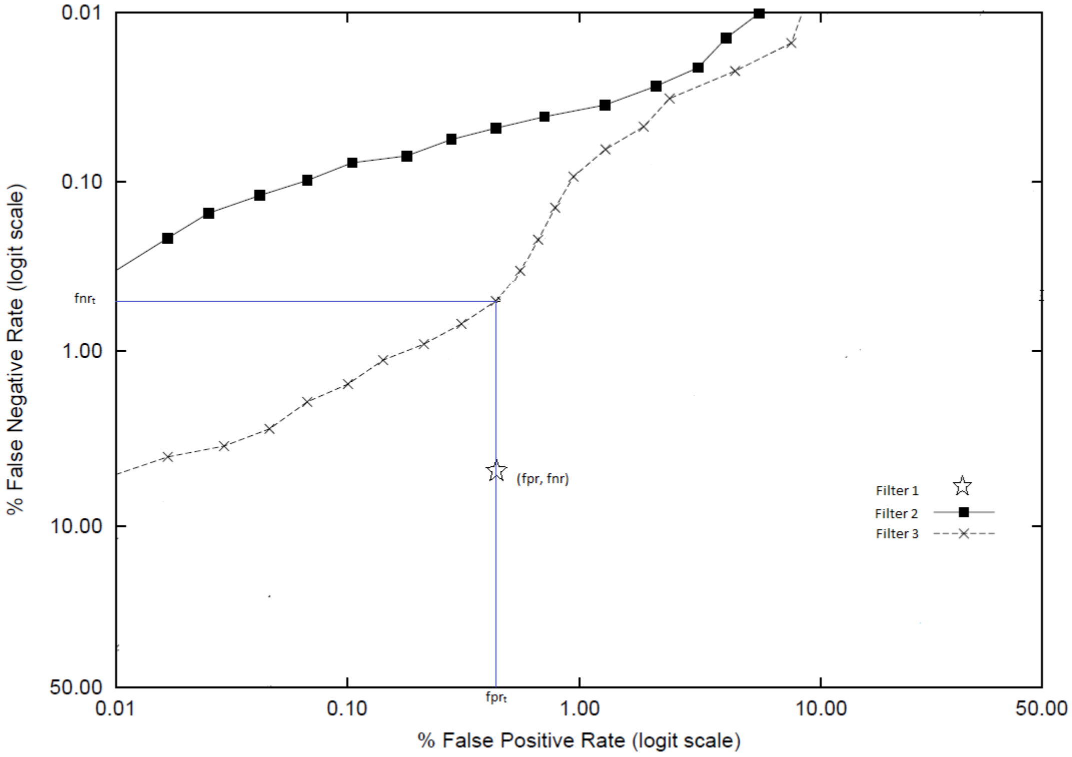


Figure 2.3: (fpr, fnr) of Filter 1, Producing Categorical Result, and ROC curves of Filter 2 and 3 Producing Scores

where $logit(x) = \log\left(\frac{x}{1-x}\right)$ and $logit^{-1}(x) = \frac{1}{1+e^{-x}}$.

False Negative Improvement Factor

One major contribution of this work is the introduction of evaluation measurements which enable us to compare the performance of the two types of classifiers introduced before. As mentioned in the previous section, we use a ROC curve to evaluate a soft classifier, whereas, a hard classifier only outputs a single (fpr, fnr) point. To be able to compare the ROC curve of a soft classifier with a single (fpr, fnr) point of a hard classifier, we introduce *false negative improvement factor: $fnif$*.

On a ROC curve, setting t such that $fpr_t = fpr$, *false negative improvement rate* will be $fnif = fnr/fnr_t$. Informally, a filter with $fnif = k$ is "k times better".

We could equally use *false positive improvement factor* $fpif = fpr/fpr_t$, which fixes $fnr_t = fnr$. The use of one or the other does not imply that the value of fpr or fnr is particularly apt; a superior curve will have both $fnif > 1$ and $fpif > 1$. The choice of measure is not critical and we follow the convention of considering the y-axis fnr_t as the dependent variable.

To compare the ROC curve of filter 2 with the single star point of (fpr, fnr), showing the performance of Filter 1, in figure 2.3, t is found such that the ROC curve of Filter 2 has the same fpr as fpr of Filter 1, and for such threshold t, $fnif$ is calculated as $fnif = fnr/fnr_t$. In this case, since the star point is below the ROC curve, fixing the fpr, the ROC curve has a lesser value for fnr (i.e. $fnif < 1$) and hence, performs better. Notice that in the y axes lower values for fnr are at the top of the axes.

The $fnif$ evaluation measure is particularly important when evaluating the performance of the proposed autonomous spam filters (producing ROC curves) with the performance of the underlying commercial global filters. In the commercial filters we only have access to the final classification of the email (spam or ham), as opposed to any internal score used to derive that classification.

Chapter 3

Personal Spam Filtering with No Recent Training Data

The main question addressed in this chapter is how well a personal spam filter learns from different sources of data, as opposed to a user's own messages. The idea of using other sources of information for training a personal spam filter follows the main goal of the thesis: to build a personal spam filter with little or no reliance on user feedback. These different messages can either be from older time frames, or belong to other users.

When looking at different sources of information that can be used for training a personal spam filter, different categories come to mind. Some of these categories are discussed in the following. We model them in our experiments and show how a personal spam filter can incorporate them. The experiments investigate how well a filter can learn from different sources of information; in particular, how well a personal spam filter can classify user's current messages when it is only trained on each of the following information sources.

Messages from older time frames: The simplest source of information for a personal filter is a set of messages from an older time frame than that of the messages the filter is asked to classify. The motivation behind the idea is, first, the fact that training examples are obviously from the past, while the filter is called upon to classify future messages; and second, older messages have higher chances of being labeled by the user. The effect on filter performance when the filter is trained on spam and ham messages sent months before the ones to be classified has been investigated by Cormack and Martins [Cormack and Martins da Cruz, 2009]. The results show that the use of old training data degrades performance of the filters.

Messages belonging to other users: A second source of information that can be used to train a personal spam filter is messages belonging to other users. Notice that the personal spam filter is still built for the specific user. The assumption is that different users may receive messages with somewhat similar characteristics. For example, in spam campaigns, millions of messages are sent in an effort to circumvent filtering before the new format is learned. Collaborative filtering – the use of labeled data delivered to other users – might provide superior performance to personal filtering; for example, in the early detection of spam campaigns. Furthermore, a brand-new user will have received no messages with which to train the filter; ideally, the filter should perform well "out-of-the-box" in spite of the lack of user-specific training data.

Messages from older time frames and belonging to other users: Another source of information that could potentially be used by a personal spam filter is older messages belonging to other users. In this category, messages used to train a personal spam filter are completely different from the messages the filter is asked to classify. In other words, the training examples the filter uses to learn from belong to other users and are coming from older time frames.

The question is which class of machine learning techniques should be utilized in order to investigate these different sources of information and which filter would perform better. Semi-supervised learning methods, as described in Section 2.2.2, are believed to help when there are not enough labeled examples for the learner to train upon. In such techniques, the training examples contain both the labeled and the unlabeled data. Furthermore, semi-supervised learning methods were among the best-performing methods at the ECML/PKDD Discovery Challenge [ecm, 2006], where the dataset somewhat resembled our problem setting.

In the ECML/PKDD Discovery Challenge, the training and test sets were drawn from different sources [ecm, 2006]. The provided dataset contained labeled training data and unlabeled test data which belonged to different users, and no user-specific labeled message was provided for training purposes. The results at the challenge suggested that semi-supervised learning methods performed best in this setting. In this chapter, we are also looking at training a personal spam filter with messages that belong to other users and are coming from older time frames. Therefore, we conjecture that semi-supervised personal filters should be used to investigate our hypothesis.

The ECML Discovery Challenge [ecm, 2006] had several limitations. The provided datasets were somewhat unrealistic, gathered from diverse sources, and with different formats and characteristics. Filters submitted to the challenge could not easily be reproduced, and most importantly, as we will see in section 3.1, the best results at the ECML challenge are orders of magnitude worse than the best results at the TREC Spam Track.

On the other hand, the TREC Spam Track [Cormack, 2007a] is the most realistic evaluation of spam filters to date, identifying the most up-to-date performance of spam filters. The TREC Spam Track introduced two tasks that also fit well into our problem scope. The delayed and inter-user (partial) feedback are the baseline for our experiments. The former models the case that some users report errors with delay, and the latter models the users who never report errors. TREC provides actual datasets without any preprocessing for the experiments. A more detailed description of the datasets available from the ECML/PKDD Challenge and TREC Spam Track is in section 3.1 and 3.2 of this chapter.

In our attempt to answer the question of how well personal filters learn from messages belonging to different sources. We describe, in detail, the ECML and the TREC spam filtering competitions. For each, we look into the datasets they provide and the filters that performed best. Next, we investigate the results of the ECML Discovery Challenge, by running SVM and TSVM on ECML dataset. Last part of this chapter, which discusses the main point, investigates whether best performing approaches at ECML (the semi-supervised methods) also perform well on the TREC dataset. The results, however, in contradiction to our hypothesis, show that semi-supervised methods yield inferior classifiers to those constructed using supervised learning on the labeled data alone [Mojdeh and Cormack, 2008b].

3.1 ECML-PKDD Discovery Challenge

3.1.1 ECML Dataset

The ECML-PKDD Discovery Challenge consisted of two tasks: Task A and Task B. The training data for both tasks consisted of 50% spam and 50% non-spam collected from publicly available sources. This manually balanced dataset is one main drawback of ECML which makes it less realistic. The test data were coming from several Inbox folders of different users. Task A labeled 4000 emails for training purposes. The test data were collected from 3 Inboxes and the tuning data were collected from one Inbox, each Inbox having 2500 emails.

Task B, on the other hand, had very few training examples - about 100 emails. The test and tuning data came from 15 and 2 Inbox folders, respectively, each Inbox containing 400 emails. The tuning data used different feature IDs from training and could only be used for parameter tuning. The raw text of emails were not provided. Instead, emails were abstracted in a bag-of-words vector space representation, using term frequencies as attributes. Features with fewer than four document frequencies were removed from the dataset resulting in a dictionary size of 150,000 words. Table 3.1 shows the diversity of emails in the ECML dataset.

Table 3.1: Composition of Labeled Training Data for the ECML dataset

Source	Task A	Task B
Emails sent from blacklisted servers (Spam)	2000	50
SpamAssassin Emails (Ham)	1600	40
Newsletters (Ham)	400	10
Total	4000	100

3.1.2 Best-Performing Filters at ECML

Each participant at the ECML-PKDD Challenge had to submit the output of his or her filter for each inbox separately. The evaluation was based on the average of all the Area Under ROC curve (AUC) values over all the Inboxes. An ROC curve, as explained in section 2.3.4, is the the set of all (fnr_t, fpr_t) achievable for some threshold t. Notice that the smaller the value of (1-AUC)%, the better the performance of the filter.

Well-performing entries at ECML Challenge used some form of semi-supervised learning algorithms, with the best achieving (1-AUC)% score of about 5% – orders of magnitude worse than the TREC results, but substantially better than the baseline supervised learning methods, which achieved (1-AUC)% score of about 15%.

In particular, the winner at Task A had an average (1-AUC)% of 4.9333%, using statistical personalized spam filtering [Junejo et al., 2006]. Junejo et al. propose an algorithm which is first trained on a training dataset and then updates its model in a two-pass phase over the personal user's emails [Junejo et al., 2006]. In the training phase, the algorithm learns a statistical model of spam and ham features from the training dataset in a single pass over the training dataset. The second phase consists of two passes over the user's Inbox (test dataset). The first global phase is supervised and the second personal phase is totally unsupervised. One main component of the algorithm is the feature selection part which is implemented by finding significant words. For a word to be considered significant, the difference of its occurrence in spam and ham messages should be above a threshold. This way of selecting features reduces the number of words that are of interest, simplifying the model and its computation.

The second winner of Task A (by an average (1-AUC)% of 5.0906) [Pfahringer, 2006] was a variant of the method proposed by Zhou et al. [Zhou et al., 2003] based on considering both local and global consistency. This algorithm is a semi-supervised graph-based method and is the basis of the algorithm proposed in chapter 4, to build a spam filter that learns from very few labeled examples.

The winner of Task B was a semi-supervised variant of Dynamic Markov modeling which made use of unlabeled training data [Cormack, 2006a], achieving an average (1-AUC)% score of 5.3531%. This algorithm improves an initial discriminative classifier derived from labeled data by iterative labeling. It ignores the separation of emails into inboxes and pools all inboxes into one unlabeled set. The method first applies Logistic Regression to both training and test sets generating a log-likelihood ratio for each message. Each message is labeled based on the log-likelihood ratio value; if positive, the message is treated as spam, and if negative, the messages is labeled as ham. Next, Dynamic Markov Compression (DMC) is applied on the whole dataset in an online manner. The data is the labeled training data in the original order, and then, the test data in the ascending order of the log-likelihood ratio. Each test message is again labeled based on the results of DMC and this process is repeated five times. Finally, the six log-likelihood ratios of each test message are averaged to get a final label.

3.2 TREC Spam Track

The 2007 TREC Spam Track used two types of corpora: public and private [Cormack, 2007a]. The public corpus (trec07p) was distributed to participants, and participants, by implementing toolkit's interface, could run their filters on the public corpus. The private corpus (MrX3), however, was not distributed to participants. Later, submitted filters, implementing the toolkit's interface, were run on the private corpus by the organizers and the results were returned to participants. The 2007 TREC Spam Track [Cormack, 2007a] included the following four tasks:

Immediate Feedback

This task was similar in both TREC 2005 [Cormack and Lynam, 2005b] and TREC 2007 [Cormack, 2007a], in which a sequence of messages was presented to the filter in chronological order using the toolkit. The filter returned a spamminess score, representing how confident the filter was about the message being spam. The correct label of each message was revealed to the filter right after the filter classifies the message.

Delayed Feedback

The delayed feedback task was first introduced in TREC 2006 [Cormack, 2006b] with the idea that users do not immediately label their emails. In TREC 2006, the delayed feedback labels for some messages were revealed to the filter after a random delay. In TREC 2007, for the first several thousand messages (10,000 in trec07p), labels were immediately revealed and for the remaining messages labels were never revealed to the filter.

Cross-user (Partial) Feedback

The cross-user or partial feedback task was very similar to the delayed feedback task, except that messages, for which labels were given, belonged to a subset of the users in the corpus. This task fits well into our problem setting where filters are trained on one set of users, but asked to classify messages belonging to other users. Section 3.5 describes how we utilize the partial feedback task in our experiments.

Online Active Learning Feedback

In the online active learning feedback task, each filter could only ask for the correct label of a *quota* number of messages. For each message, the filter returned another value indicating whether or not it required the label. If a filter had not implemented this feature, then labels for the first quota number of messages were given to the filter, and so the active feedback task would be the same as the delayed task.

3.2.1 TREC Datasets

Experiments in this chapter used the corpora in TREC 2005 and 2007 Spam Tracks and, hence in this section, we focus on the details of datasets provided at these tracks.

TREC 2005

Among the different corpora that TREC 2005 Spam Track provided [Cormack and Lynam, 2005b], we focus on the public and private corpora in Table 3.2, where the details of these datasets have been shown. The public corpus (trec05p) consists of messages released publicly from Enron. These messages are part of the one million messages and files from the email folders of 150 Enron employees that were released to the public by the Federal Energy Regulatory Commision's investigation. The corpus contains about 92000 emails, out of which 39000 emails are spam and the rest are ham.

The main private corpus of TREC 2005 is MrX which was created by Cormack and Lynam in 2004 [Cormack and Lynam, 2005a]. The email collection consists of the 49000 messages received by an individual, X, from August 2003 through March 2004. X has had the same email address for twenty years; variants of X's email address have appeared on the Web and in Usenet archives. X has subscribed to services and purchased goods on the Internet. X used a spam filter "Spamassassin 2.60" during the period in question, and reported observed misclassifications to the filter.

Table 3.2: TREC 2005 Spam Track Datasets

Corpus	Number of Spam	Number of Ham	Total
trec05p (Public) Corpus	52,790	39,399	92,189
MrX (Private) Corpus	40,048	9,038	49,086

TREC 2007

TREC 2007 Spam Track provided two corpora [Cormack, 2007a]: public and private. Table 3.3 shows details of them. The public corpus contains about 75,000 messages, of which 50,000 are spam and the remaining are ham. All the messages in the pubic corpus were delivered to many accounts on a particular server between April 8 and July 6, 2007. The corpus also contains messages delivered to accounts created on honeypots.

The private corpus, MrX3, same as MrX corpus [Cormack and Lynam, 2005b], are messages received by X, from December 2006 to July 2007. This corpus contains about $160,000$ messages, of which about 153,000 are spam and the remaining are ham.

Table 3.3: TREC 2007 Spam Track Dataset

Corpus	Number of Spam	Number of Ham	Total
trec07p (Public) Corpus	50,199	25,220	75,419
MrX3 (Private) Corpus	153,893	8,082	161,975

3.2.2 Best-Performing Filters at TREC

TREC Spam Track uses Receiver Operating Characteristic (ROC) and the Area Under the ROC curve (AUC) to evaluate the performance of submitted filters. A brief review of the best-performing filters at the TREC Spam Track is discussed here, with the focus on the filters that outperformed others in the delayed and partial feedback tasks.

The same filters shared the top ranking places in both private and public corpora [Cormack, 2007a]. In the public corpus a fusion of Logistic Regression and Dynamic Markov Compression performed the best in both the immediate feedback and the delayed feedback tasks, achieving (1-AUC)% score of 0.0055 and 0.0086 in the two tasks, respectively [Cormack, 2007a]. Variants of online relaxed Support Vector Machines (ROSVM) [Sculley and Wachman, 2007a] performed very well in all four tasks. This method achieved a (1-AUC)% score of 0.0093 in the immediate feedback task, 0.0214 in the delayed feedback task, 0.0858 in the partial feedback task, and 0.0144 in the online active learning task.

The TREC Spam Track results [Cormack and Lynam, 2005b, Cormack, 2007a] showed that personal spam filters – those trained only on past email received by the recipient – achieved very strong results, with typical AUC scores of 0.9999 or better [i.e. (1-AUC)% = 0.01%]. It may be argued that these results are unrealistically good because no user would provide perfect feedback. TREC experiments which measured the effects of delayed and missing feedback showed some degradation, but still very strong results. Further degradation, but still strong performance with (1-AUC)% scores of 0.1%, was observed when training was affected using messages from users separate from those for whom filter performance was tested. It must be noted that none of the well-performing filters at TREC harnessed unlabeled data, although it was available in the form of previously classified messages.

3.3 Filters

Our hypothesis in undertaking the experiments reported in this chapter is that semi-supervised learning methods can outperform fully-supervised algorithms, when training and test sets belong to different sets of users, different time frames, or older messages belonging to a different set of users. The basis for this assumption is the results at the ECML Discovery Challenge [ecm, 2006]. We evaluated supervised and semi-supervised variants of three spam filter methods known to perform well at TREC [Cormack, 2007a] and ECML/PKDD [ecm, 2006]. A detailed explanation of each method is described in Section 2.2.4.

3.3.1 Support Vector Machines

The first filter evaluated in this chapter is the Support Vector Machine (SVM), which is a learning method that is mostly used in classification and regression [Vapnik, 1998]. In classification theory, input is usually a set of data points, each belonging to one of many possible classes. SVM finds the maximum margin hyperplane which has the largest distance to the nearest data points of any class. Labels of the test examples are predicted based on which side of the hyperplane they are mapped to. Transductive

SVM (TSVM) is the semi-supervised variant of SVM, where the training set contains both labeled and unlabeled examples, and the hyperplane should separate both labeled and unlabeled data with maximum margin through regularization. Both SVM and TSVM are described more elaborately in Appendix 2.2.4.

SVMlight [svm, 2008] is an implementation of SVM in C, which we have utilized in this experiment. The algorithm has scalable memory requirements and can efficiently handle problems with many thousands of support vectors. It provides features such as fast optimization, handles several hundred thousand training examples, and uses sparse vector representation. It also includes an algorithm for approximately training large TSVMs.

Optimal Setting for Transductive SVM

In experiments with both SVM and TSVM, all parameters were set to the default in SVMlight. TSVM was also run with the optimal value setting for the parameter p (to get an upper bound for its performance). The optimal Transduction option in SVM, p, varies between 0 and 1 and determines the fraction of unlabeled examples to be classified into the positive class. In the case of spam filtering, we have $p = \frac{\text{Number of spam messages}}{\text{Total number of messages}}$. The default value for this parameter is the ratio of positive examples to the total number of examples in the training set, and the optimal value is the same ratio in the test set. Note that setting this parameter to the optimal value (the ratio of positive examples to the total number of messages in the test set) requires advance knowledge of the test set.

In this section, we report the performance of TSVM when this parameter is set to both its default value and its optimal value. The result of TSVM with the optimal value of p is shown in parenthesis. This result, however, indicates an upper bound for the performance of TSVM rather than an actual achieved result. The default value for p is 50%.

3.3.2 Dynamic Markov Compression

Dynamic Markov Compression (DMC) [Bratko et al., 2006b], first, classifies a message by building two compression models from the training corpus, one from examples of spam and one from legitimate email. Next, the compression rate achieved by compressing the message using the two compression models is calculated. The message is classified as spam if the compression rate using the spam model is better than the one using the ham model. In other words, DMC classifies each message by adding it to the entire set, using both models, and the model which adds the minimal increase in the description length of the entire corpus is considered as the class of the message. A more detailed description of DMC is in Appendix 2.2.4. DMC with self-training is a semi-supervised version that is used in this experiment.

3.3.3 Logistic Regression

Logistic Regression (LR) is a linear classifier with a gradient descent update model [Goodman and tau Yih, 2006]. The classifier consists of a weight vector, and the probability of each message being spam is equal to logistic function of the inner product of the message's feature vector and the weight vector. In the update mode, the weight vector is updated according to the message's predicted probability of being spam and the message's ground truth label. A detailed description of both the classification and the learning stages of LR is presented in Appendix 2.2.4. The semi-supervised variant of LR that we use in this chapter is LR with self-training.

3.3.4 Self-training

The semi-supervised variant of both LR and DMC utilizes self-training. *Self-training* is a semi-supervised approach used to enhance the classifier with unlabeled data. Some literature has used the term "weakly-supervised" for self-training [He and Gildea, 2007]. In self-training, the learner is iteratively improved by using self-generated feedback. The learner usually starts by using the labeled training data to build a classifier which then classifies the unlabeled data. Those examples meeting a selection criterion are added to the labeled set and the classifier is retrained on this newly labeled data set. This training procedure continues for several rounds.

In the literature, self-training has been previously studied with SVM [Li et al., 2008]. It has also been utilized with a variant of Naive Bayes as an underlying classifier, to extract subjective information from text [Wang et al., 2008]. Vinh et al., applied self-training to improve a scoring function in a ranking algorithm [Truong et al., 2009]. A recent work used consensus-based self-training with different base classifiers to categorize multilingual text [Amini et al., 2010]. In this algorithm, a committee of classifiers are trained on labeled examples and, independently, each classifies unlabeled examples. Those examples, in which all classifiers agree on the classification, are added to the training set. This procedure repeats until a stop condition is met.

3.4 Experiments on ECML Dataset

Our first experiment is reproducing results at ECML which suggested that semi-supervised algorithms performed the best, with the datasets provided at ECML [ecm, 2006], being very diverse. Among the well-performing filters at ECML, we experiment with SVM and its semi-supervised variant, TSVM. There was no tuning needed for parameter p in the SVM, since the default value of p is 50%, the same ratio of spam in the dataset provided. ECML dataset is in a bag-of-words model, with feature weights being term frequency. We also tried the binary weights for this experiment which did not improve the results. Because the format of the ECML dataset, we could not apply the original versions of DMC or LR.

The evaluation measure used in this experiment is the (1-AUC)%, where AUC is the Area Under ROC curve, see 2.3.4.Table 3.4 shows the results of SVM and TSVM on the ECML Task A dataset. As mentioned in Section 3.1, the ECML dataset has three users, and the results also show the average over all users. Although the results that are presented here are worse than those reported for the ECML, our results does follow the same trend [ecm, 2006]; that is the semi-supervised approach does outperform the supervised version of SVM. However, the results achieved here are by an order of magnitude worse than the results at TREC.

Table 3.4: Results for Different Users of ECML Dataset, (1-AUC)%

Method	User 0	User 1	User 2	Avg
SVM	29.066	22.976	9.760	20.601
TSVM	20.294	19.563	3.349	14.016

3.5 Experiments on TREC Dataset

In order to experiment with the TREC public corpus (trec07p), we consider four different scenarios. The first is the baseline which incorporates the full feedback model. The second investigates whether semi-supervised methods can help the delay issue between training examples and test examples and potentially solve the first question we raised in the introduction of this chapter. The third scenario is a cross-user experiment where we look into the performance of semi-supervised methods, when the training and test sets belong to different users. The results are interesting in terms of answering the second question. The fourth experiment deals with comparing supervised and semi-supervised methods on the training and test examples belonging to two different corpora, and so belonging to different users in different time-frames.

We evaluate different variants of SVM, LR, and DMC. The TREC dataset includes emails in raw format. For the experiments with SVM, we preprocessed the TREC dataset into a SVM-compatible input file; that is, the bag-of-words model with features being overlapping 4-gram characters. A 4-gram is contiguous sequence of 4 characters. Feature weights are the term frequency of the feature. For LR and DMC methods, features have been selected as overlapping 4-gram characters. All filters use the first

3500 bytes from the raw message format. The next sections describe each scenario with the results.

3.5.1 Baseline

In the baseline experiment, we investigate the results of the LR and DMC in an online manner with immediate feedback on the TREC 2007 dataset. Table 3.5 shows the baseline performance of the DMC and LR filters with full feedback; that is, the filters are operated online and the correct label for each message is communicated to the filter immediately after classification. The results show the (1-AUC)%, AUC being the Area Under Curve. Notice that the smaller the value of (1-AUC)%, the better the performance of the filter. As shown in the table, both DMC and LR perform extremely well in the presence of full feedback. SVMlight is not adaptable in an online format, and hence, it does not appear in our baseline experiments.

Table 3.5: Baseline Results (1-AUC)%

Method	Online with Feedback
DMC	0.007
LR	0.040

3.5.2 Delayed Feedback

Here, the TREC 2007 methods and datasets were adapted for batch filtering as opposed to online filtering [Cormack and Bratko, 2006] to maximize the opportunity for semi-supervised learning. The *delayed feedback* task from TREC was modeled using the following training and test sets:

- Training set: The first 10,000 messages of the TREC 2007 public corpus

- Test set: The remaining 65,000 messages from TREC 2007 are divided into six batches of 10,000 messages, and each set is used as a test set. The remaining $5,000$ messages are not used.

In this experiment, filters are trained on the training set in the batch mode and no label is released to the filters for the test sets. Table 3.6 shows the results of this experiment. All results are presented as the average of (1-AUC)% over all test sets. The DMC and LR results in the first column, compared to the results in Table 3.5, show that both filters suffer from lack of feedback (DMC suffers more). The SVM filter shows an intermediate performance.

The second column of Table 3.6 shows the result of the semi-supervised variants of the three methods: TSVM, DMC and LR with self-training. None of the filters was significantly improved by semi-supervised methods that worked well for the ECML/PKDD challenge. The parenthesized result (0.053) indicates the result that SVM would have achieved with an optimal parameter setting, p, indicating the prevalence of spam. Since this parameter was computed with knowledge of the data, it indicates an upper bound rather than an achieved result.

Table 3.6: Delayed Feedback Results (1-AUC)%

Method	Supervised	Semi-Supervised
DMC	0.016	0.090
LR	0.049	0.046
SVM	0.030	0.230 (0.053)

3.5.3 Cross-user Feedback

The question we address in this section is whether personal filters can learn from other users' messages. We evaluate this question by having personal filters trained on messages belonging to some users, and testing the filter on messages belonging to different users. Following the hypothesis presented before, we evaluate the fully supervised and the semi-supervised variants of the best-performing methods at ECML/PKDD and TREC, in an attempt to see whether semi-supervised learning would help a filter perform better when only cross-user feedback is provided.

As described in section 3.2.1, in the *partial feedback* task at the TREC Spam Track [Cormack, 2007a], the feedback is given to the filters for messages belonging to a subset of users. We model the cross-user feedback using the following training and test sets.

- Training set: 30338 messages for which feedback was given (the messages addressed to a particular subset of users).
- Test set: the remaining 45081 messages (the messages addressed to other users).

These two sets are then swapped, resulting in two experiments.

Table 3.7 shows the results of cross-user training, averaged over the two sets. Results show the (1-AUC)% for both the fully supervised and the semi-supervised variants of LR, DMC, and SVM. The performance of the supervised filters is compromised far more substantially by cross-user training than by training delay. The semi-supervised methods perform much worse.

Table 3.7: Cross-user Results (1-AUC)%

Method	Supervised	Semi-Supervised
DMC	2.17	9.97
LR	1.00	10.72
SVM	1.06	24.3 (1.89)

3.5.4 Cross-corpus Feedback

The purpose of this experiment is to show how well filters can learn from totally different messages, that is, messages belonging to different users and times frames. A cross-corpus experiment is conducted here in an effort to reproduce the results of the ECML/PKDD Discovery Challenge. As for the filters, both fully supervised and semi-supervised variants of LR, DMC, and SVM are evaluated. TREC 2005 and TREC 2007 corpora are used to model the cross-corpus experiment. The training and test sets for the cross-corpus feedback task are as follows:

- Training set: First $10,000$ messages of the TREC 2005 public corpus.
- Test set: TREC 2007 public corpus, split into batches of $10,000$ messages.

The results of this cross-corpus evaluation are shown in Table 3.8. The results show the (1-AUC)%, averaged over all segments. As the table shows, all the filters perform poorly – at least compared to the other tasks – but are still better than chance. Self-training is again harmful; however, SVM shows considerable improvement, yielding the best-performance on this dataset, even with the default prevalence parameter.

Table 3.8: Cross-corpus Results (1-AUC)%

Method	Supervised	Semi-Supervised
DMC	14.9	49.1
LR	17.8	43.0
SVM	23.5	13.1 (10.3)

3.6 Contributions and Discussion

The question addressed in this section is whether personal spam filters can learn from different messages – much older messages or messages belonging to other users. The idea of using other sources of information for training is to build a personal spam filter with no reliance on user feedback. ECML/PKDD Challenge provided datasets that matched our problem setting, and the best-performing spam filters at the challenge used variant of semi-supervised learning methods. This led us to believe that semi-supervised variants of the best-performing filters at TREC would also give us a better performance over the delayed, cross-user, and cross-corpus datasets drawn from the TREC Spam Track.

The experiments included Logistic Regression, Dynamic Markov Compression, and SVM. Self-training was used as the semi-supervised variant of the first two, and for SVM, Transductive SVM was used. However, the experiments led us to realize that semi-supervised variants of the best-performing filters in TREC could not reproduce the results of ECML/PKDD and help improve the delayed or cross-user filtering process. Surprisingly, only Transductive SVM showed better performance over the fully-supervised SVM and that was only when the training and test sets were of a completely different corpus, meaning they were coming from different time frames and belonging to different users.

Chapter 4

Personal Spam Filtering with Very Few Training Examples

The question we are dealing in this chapter is how well a personal filter performs when there are only few training examples available. Despite significant progress in recent years, many spam filtering techniques require large number of training examples (often hundreds or thousands). We investigate ways to have a personal filter which performs well when there are only few labeled examples available for training purposes and a huge number of unlabeled messages for classification.

In the case of spam filters, a normal user might be willing to provide a few labeled examples, but simply cannot go through all messages and label them. Yet, the user still expects correct classification for all the messages. An example for this case is during the initial deployment of a spam filter on the user's machine. The spam filter is expected to perform well without any labeled messages as training data. The user, in this scenario, might be willing to help the spam filter by labeling just a few messages. In another example, the spam filter can classify a few messages and ask the user to adjudicate those messages. The spam filter can then proceed to classify the rest of the messages.

The idea behind spam filtering with few training examples follows the main motivation we raise in this thesis, which is how to build a personal spam filter with minimal user feedback. In the previous chapter, we investigated the performance of a personal spam filter with no recent source of user feedback for training purposes. The results were encouraging, but nothing like when the spam filter is presented with a full set of feedback. This led us to investigate the question of the performance of the personal spam filter in the presence of just a few recent labeled examples belonging to the user.

4.1 Introduction

Designing a spam filter where the number of training examples is very small and the dimensionality of the feature space very large is extremely challenging. This problem has been addressed by many researches and several promising techniques have been proposed. As we emphasized in section 2.2.2, a number of semi-supervised learning algorithms have been introduced in the literature aiming to improve the performance of classification in supervised spam filters by using unlabeled emails together with the labeled ones [Cormack, 2006a, He et al., 2007, Gao et al., 2008, Zhu et al., 2003, Zhu, 2007]. In these techniques, labeled and unlabeled emails are often used simultaneously or in close collaboration in many semi-supervised spam filtering methods. The unlabeled emails help to capture the underlying structure of the data, whilst the labeled emails are mainly used for classification task within the provided structure.

In Chapter 3, we investigated on the performance of semi-supervised spam filter when using other sources of information for training to build a personal spam filter with no reliance on user's feedback. However, we showed that semi-supervised methods such as Transductive SVM (TSVM), Logistic Regression (LR) and Dynamic Markov Compression (DMC) with self training for spam filtering yield poor results. In specific, only Transductive SVM showed better performance over the fully supervised SVM and that was only when the training and test sets were coming from different time frames and belonging to different users.

In this chapter, we are focused on the special situation in which only the first handful of messages are labeled and they are used to filter the rest. Here, we show that the former techniques are not providing a good result when there are only a few labeled emails. We present an aggressive graph-based iterative solution, modeled after the local and global consistency learning method of Zhou et al. [Zhou et al., 2003]. A variant of this method ranked first in Task A of the ECML/PKDD Discovery Challenge [Pfahringer, 2006]. To find the most informative terms, we also apply Singular Value Decomposition (SVD) [Alter et al., 2000]. Our experiments show a comparatively high performance in the presence of very few training samples.

The organization of this chapter is as follows: First, a literature of graph based semi-supervised methods is presented in Section 4.2. Section 4.3 describes the adapted algorithm based on, the global and local consistency method. The design of the experiments including the datasets, the composition of training and test sets, and the tuning procedure are described in Sections 4.4 and 4.5. Finally, we present the encouraging results of our proposed method and its comparison to other methods, in Section 4.6.

4.2 Graph Based Semi-supervised Learning

Graph-based semi-supervised methods define a graph where the nodes are examples in the dataset, and the edges, usually weighted, show the similarity between nodes. These nodes include both the labeled and the unlabeled examples. The idea behind graph-based semi-supervised learning is to find the class of unlabeled examples by propagating the class from the labeled ones.

Graph-based methods are transductive, meaning that the built classifier is geared towards the examples provided (training and test sets), and can only be used to classify the examples in the test set. A transductive learner is in contrast to an inductive learner, where the learner builds a model that can be later used to classify any example. We should mention that Transductive SVMs are in fact inductive, since they build a classifier that could be used later to classify any example.

In graph-based semi-supervised methods, there are two assumptions:

1. The nearby points (nodes) are likely to have the same label; and
2. The points on the same structure are likely to have the same label. This structure is usually called a cluster or manifold [Chapelle et al., 2002].

Without going into theoretical details, we will discuss related literature about graph-based semi-supervised methods, specifically, the ones applied to spam filtering.

The learning problem on graphs can be considered as estimating a classifying function f which should be close to a given function y on the labeled data and should be smooth on the whole graph [Zhou et al., 2003]. For most of the graph-based methods, this can be expressed in a regularization framework [Zhu et al., 2003] where f is a loss function and y is a regularizer. The loss function should be minimized on the whole graph using (iterative) tuning of the edges values. Different graph-based methods mainly vary by the choice of the loss function and the regularizer [Zhu et al., 2003]. However, these differences are not actually crucial. What is far more important is the construction and the quality of the graph, which should reflect domain knowledge through the similarity function which is used to assign edges (and their weights).

Zhou et al. propose a method called learning from Local and Global Consistency (LLGC) [Zhou et al., 2003] where its main goal is to balance two potentially conflicting goals: (i) *local consistency*, similar examples should have similar class labels, and (ii) *global consistency*, the predicted labels should agree with the given training labels. Note that this can be considered as designing a function f that satisfies both local and global consistency assumptions. The LLGC method aims to achieve a stable state by allowing every point iteratively spread its label information to its neighbors. Therefore, LLGC can be considered as the process of information diffusion on graphs [Kondor and Lafferty, 2002]. The weights are scaled by a parameter σ for propagation. During each iteration, each point receives the information from its neighbor and also retains its initial information. A parameter α adjusts the relative amount of information provided by the neighbors and the initial one. When convergence is reached, each unlabeled point is assigned the label of the class it has received most information for during the iteration process. A variant of LLGC is also applied for the detection of web spam [Castillo et al., 2007].

As stated by Zhou et al. [Zhou et al., 2003], the closest related graph-based approach to LLGC is the method using Gaussian random fields and harmonic functions presented in [Zhu et al., 2003]. In this method, the label propagation is formalized in a probabilistic framework. The probability distribution assigned to the classification function f is a Gaussian random field defined on the graph. This function is constrained to give their initial labels to labeled data. The minimization of the cost function results in a harmonic function.

Relying on the fact that LLGC has demonstrated impressive performance on relatively complex manifold structures, Pfahringer et al. apply a scalable variant of the standard LLGC algorithm to cope with its computational complexity, to broaden its range of applicability, and to improve its predictive accuracy [Pfahringer et al., 2007]. The first change in their method is using the cosine-based similarity to measure the similarity between labeled and unlabeled points. The second change is allowing prelabeling by using class-distributions for unlabeled data that have been computed in some way using the training data. Lastly, the complexity of LLGC is reduced by sparsifying the affinity matrix by only including the k nearest neighbors in the matrix, making sure that the matrix remains symmetric in a post-processing step after the kNN computation. This method ranked first in Task A of the ECML/PKDD Discovery Challenge [Pfahringer, 2006]. This challenge was explained more thoroughly in

Section 2.3.

4.3 Aggressive Consistency Learning Method

Following the good performance of LLGC algorithm and its variants [Pfahringer et al., 2007], we propose another variant of semi-supervised LLGC algorithm which performs relatively well with a few training examples. This technique is called Aggressive Consistency Learning Method, which we describe in detail in Algorithm 1.

The following notation is used in the rest of this chapter. Given a sequence of n email messages and the labels denoting the true class – *spam* or *ham (non-spam)* – for each of the first $n_{labeled} \ll n$ messages, we consider the problem of finding the class of the remaining $n - n_{labeled}$ messages. The input matrix X_{n*m} represents the feature vector of the messages, where n is the number of messages and m is the number of terms. The vector Y_{n*1} represents the labels of the messages, where each element of this vector is

$$\{y_i \in \{-1, 1\} \text{ for } i \leq n_{labeled} \text{ and } y_i = 0 \text{ for } i > n_{labeled}\}. \tag{4.1}$$

The output of the algorithm is the labels for the last $n - n_{labeled}$ messages:

$$\{y_i \in \{-1, 1\} \text{ for } i > n_{labeled}\}. \tag{4.2}$$

Algorithm 1 Aggressive Consistency Learning Method (ACLM)

```
Input: $X$, $Y$, $\alpha$, $\sigma$
1: Compute Affinity matrix
   $$A_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} & \text{for } i \neq j \\ 0 & \text{for } i = j, \end{cases} \tag{4.3}$$
2: For all $j$ such that $\sum_i A_{i,j} \approx 0$ : $A_{rj} = 1$ where $r = \arg\min_j \|x_r - x_j\|$
3: Compute $L = D^{-1/2} A D^{-1/2}$ where
   $$D_{ii} = \sum_{j=1}^{n} A_{ij} \,. \tag{4.4}$$
4: $\mathbf{Y} = (\mathbf{1} - \alpha)(\mathbf{I} - \alpha \mathbf{L})^{-\mathbf{1}} \mathbf{Y}$
```

The $n \times n$ symmetric Gaussian affinity matrix A captures the similarity between each pair of messages $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is the Euclidean distance between messages $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that other than the $A_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$ similarity measure, we also evaluated the cosine similarity: $A_{ij} = \frac{x_i . x_j}{\|x_i\| \|x_j\|}$, and the inverse Euclidean distance: $A_{ij} = \frac{1}{\|x_i - \mathbf{x}_j\|^2}$. Among which, the similarity measure $A_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$ resulted in the best performance, and therefore, is used in our algorithm. This is in contrast to previous literature that proposed using cosine similarity for text classification in this method [Pfahringer, 2006].

The second step in our algorithm performs two tasks: The first task is to aggressively update the affinity matrix. In all the columns r of the affinity matrix A with summation very close to zero[1], we aggressively set the element corresponding to $\min_j \|x_r - x_j\|$ to 1. Even though, this step changes the symmetrical property of the

[1] The threshold for summation in our program is set to 0.001, i.e., if the summation of the elements of the column r is less than 0.001, we keep only one element in that column.

affinity matrix A, it can be easily fixed in the next step by normalization. The second task is reducing computational complexity by sparsifying the matrix A, as it is the main source of complexity in the LLGC algorithm. This step reduces the amount of memory needed to store the matrix ($O(n)$ compared to $O(n^2)$) and simplifies calculation of the inverse of the affinity matrix.

In step 3, the affinity matrix A is normalized symmetrically by constructing $L = D^{-1/2}AD^{-1/2}$ [Zhou et al., 2003]. It should be mentioned that because matrix L is symmetric, the information is spread symmetrically. Note that steps 1 and 3 are the same as those in spectral clustering [Ng et al., 2001].

Step 4 is in fact the result of the convergence of the iteration $F(t+1) = \alpha LF(t) + (1-\alpha)Y$, where $F(0) = Y$ (see [Zhou et al., 2003]). During each iteration, each point receives the information from its neighbors (first term), and also retains its initial information (second term). The parameter $\alpha \in (0, 1)$ determines the relative amount of information that each node in the graph receives from its neighbors. When convergence is reached, each unlabeled point is assigned the label of the class it has received most information for, during the iteration process, in which $\lim_{t\to\infty} F(t) = (1-\alpha)(I - \alpha L)^{-1}Y$.

In summary, the main contribution of this algorithm is its aggressive approach in updating the affinity matrix. A large number of elements in the affinity matrix are approximately zero due to the large Euclidean distances between messages, meaning that the messages do not share many terms. In our aggressive definition of the affinity matrix, for all zero rows or columns in equation (4.3), a "1" (equivalently, a link in the graph) is inserted where the distance between the two corresponding messages is minimum in that column or row. Although adding a link in this case may seem too aggressive, the simulation results demonstrate an improved performance.

4.3.1 Dimensionality Reduction using SVD

In order to better handle the sparsity of the affinity matrix A, we propose to reduce the dimensionality of matrix X by applying Singular Value Decomposition (SVD) [Alter et al., 2000]. SVD is a method for factorization of a matrix and has many applications such as data analysis, signal processing, pattern recognition, image compression, weather prediction, and Latent Semantic Analysis (LSA) – also referred to as Latent Semantic Indexing (LSI).

If M is the matrix to be reduced in dimensionality, the following SVD factorization always exists [Alter et al., 2000]

$$M_{n\times p} = U_{n\times n} \times \sum_{n\times p} \times V^T_{p\times p} \tag{4.5}$$

where the matrix M is of size $n \times p$, and matrices U and V are unitary and of sizes $n\times n$ and $p\times p$, respectively. Matrix $\sum$ is a diagonal matrix of size $n\times p$ and it contains the singular values of M. When reducing the dimensionality of M, the singular values are sorted, and the ones above a certain threshold are selected and the rest are set to zero. The result of applying the new matrix $\sum$ in the equation 4.5 reduces the dimensionality of matrix M.

Returning to our algorithm, by applying SVD on matrix X, we find the most informative terms in X by first finding matrices U, $\sum$ and V such that $X' = U \sum V^{-1}$, keeping the singular values greater than a certain threshold t in matrix $\sum$, and setting the rest to zero $\{\sum_{i,i} = 0, \ \forall \ i > t\}$. Next, matrix X is recalculated using the new matrix $\sum$ and is passed as the input to Algorithm 1.

4.4 Filters and Corpora

In order to evaluate our proposed algorithm, we evaluate the following filters and use TSVM as the baseline:

Aggressive Consistency Learning Method (ACLM) with and wihout SVD:
This method is described in Algorithm 1 in Section 4.3. This algorithm is implemented by an aggressive approach in updating the affinity matrix. In order to evaluate the ACLM filter, we considered implementing it with and without SVD. The main motivation behind adding the SVD algorithm is to reduce the dimensionality of the problem. Our experiments show that ACLM with SVD outperforms ACLM without SVD.

Support Vector Machines (SVM):
Support Vector Machines (SVM), are learning methods, used for classification and regression [Vapnik, 1998]. In the binary classification setting, input is usually a set of points, each belonging to one class. SVM finds the maximum margin hyperplane which has the largest distance to the nearest data points of any class. This distance is called the margin and shows the generalization error of the classifier. Test examples

are then mapped to the same space, and labels are predicted based on which side of the hyperplane they are mapped. We use the established free-to-use SVMlight classifier [svm, 2008].

Transductive Support Vector Machines (TSVM):

Transductive SVMs (TSVMs) is the semi-supervised variant of SVM. Here, the training set contains unlabeled examples, and the hyperplane should separate both labeled and unlabeled data with maximum margin. TSVMs are, in fact, inductive learners because the resulting classifiers are defined over the whole space, and unseen test examples can also be mapped to the space. More details about SVM and TSVM can be found in Appendix 2.2.4.

It is worth mentioning that when SVMs are applied with SVD for text classification, the results, although generally reasonable, are worse than when a SVM is used with the full feature set [Pereira and Gordon, 2006, Kim et al., 2005]. Hence, in our experiments, we don't apply dimensionality reduction on SVM and TSVM, and only evaluate them with the complete features.

The datasets we use for this comparison are two known datasets, previously described in Section 2.2.4: TREC 2007 and CEAS 2008.

TREC 2007:

TREC Spam Track 2007 has a publicly available corpus with 75,419 messages (25,220 ham and 50,199 spam messages). The messages are in raw format, and in chronological order. More about the TREC 2007 corpus is described in chapters 2.3 and 3.

CEAS 2008:

The lab corpus at CEAS consists of the exact messages used in the CEAS 2008 live spam challenge. The corpus consists of a sequence of 123,100 messages (23,844 ham and 99,256 spam) with labels indicating the filter result and also the gold standard as adjudicated by CEAS.

Both of these corpora provide messages in their raw format. For the sake of the experiments, we consider the first 10000 messages of each corpus and datasets are tokenized. More about the methodology used in the experiment appears in the next section.

4.5 Experimental Design

The experiments aim to evaluate and compare the methods explained above with the presence of very few training examples. The results show the performance of the filters as the size of the training set grows, starting from as small as two.

From the TREC 2007 and CEAS 2008 datasets, the first 10000 messages are considered for the experiments. The first 1000 messages of those are used for tuning proposes and the remaining 9000 messages are used for evaluation.

4.5.1 Training and Test Sets

The 9000 messages are divided into batches of 1000 messages. The training and test sets are selected from each batch and the results show the average of the results over all the batches. From each batch of 1000 messages, the first 100 messages are used to build the training set, and the remaining 900 messages are used for the test set.

The training set is a balanced set of positive and negative examples (spam and ham messages, respectively) from the 100 messages. For a training set size of two, for example, the first spam message and the first ham message are selected from the 100 messages to form the training set. We report the results for each training set size.

4.5.2 Feature Selection

For the process of feature selection, each message, including the header is tokenized into words. Words with document frequency fewer that 5 are removed from the feature set. The document frequency is the number of documents in both the training and the test sets, which contain the specific term. Next, each message is abstracted as a feature vector, features being the words.

We evaluated the actual and the binary term frequencies. The binary term frequency simply assigns one to a feature, if the feature appears in the document, and assigns 0 if it does not. The results of our preliminary experiments let us believe that binary feature vectors outperform the actual weights and, hence, we continue using binary term frequency as feature vectors.

4.5.3 Parameter Tuning

For the purpose of tuning, the first 1000 messages from each dataset are selected, of which the first 100 messages are used to select a training set, and the remaining 900 messages are used for evaluation. The training set is chosen from the 100 messages as described before and different values for each parameter is evaluated for different sizes of the training set. The best value appeared to be the same for all training sizes.

There are three parameters in the Aggressive Consistency Learning Method that need tuning: σ, α, and t. Out of the several values tried for each parameter, the following values resulted the best: $\sigma = 1/sqrt(2)$, $\alpha = 0.99$. Previous literature using the Local and Global Consistency algorithm in text classification report $\alpha = 0.95$ [Pfahringer, 2006] and $\alpha = 0.99$ [Zhou et al., 2003]. In the SVD process, and to find a certain threshold to cut off the singular values in the diagonal matrix $\sum$, t was to set to the 2.5% of the largest singular value in matrix $\sum$, so all the values lower than 2.5% of the largest singular value are set to zero.

For parameters of SVM and TSVM, several values were adjusted using the same scenario, but no improvement over their default values was observed. The p parameter in TSVM, representing the proportion of spam messages to be expected, was tuned using our tuning set of emails.

4.5.4 Evaluation Measures

We follow previous literature [Zhou et al., 2003, Pfahringer, 2006] and use error rate is used to capture the efficiency of the explained methods, which is the percent of messages in the test set that are misclassified by the classifier. The results in the plots show the mean error rate over all the test batches.

Sine the error rate The results are also evaluated using Logistic Average Misclassification (LAM) as discussed in 2.3.4. This metric incorporates both the false positive and false negative rates.

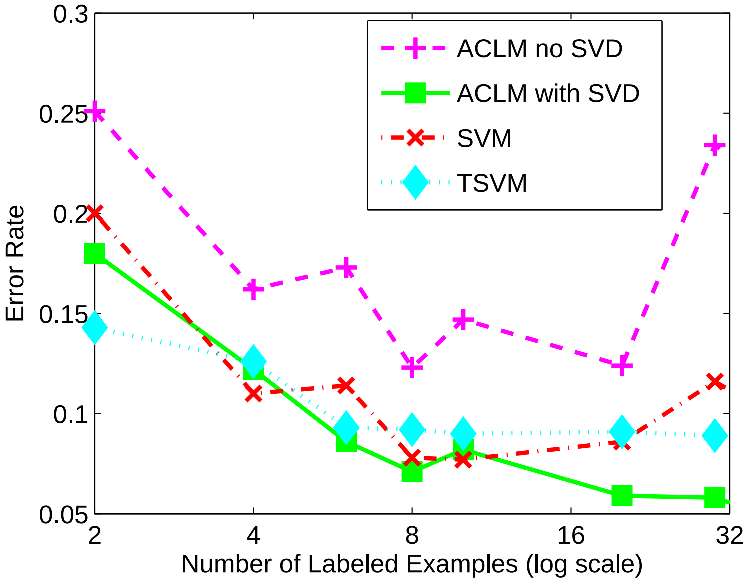
4.6 Results

4.6.1 TREC 2007

Figure 4.1 shows the results of the methods on the TREC 2007 dataset. The results show the mean error rate of the methods for different sizes of training sets. ACLM with SVD generally gives the best performance of all methods between 4 and 32 labeled examples, having less than 0.01 error rate for most of the training sizes. Only SVM slightly outperforms ACLM in two training sets of sizes 4 and 10. ACLM without SVD performed poorly for all training sizes.

Figure 4.2 shows the LAM results for the four filters on the TREC 2007 dataset. LAM curve of ACLM with SVD outperforms all other filters with all sizes of the training data.

Figure 4.1: Error Rate for ACLM (with SVD and without), SVM, TSVM on the TREC 2007 Corpus

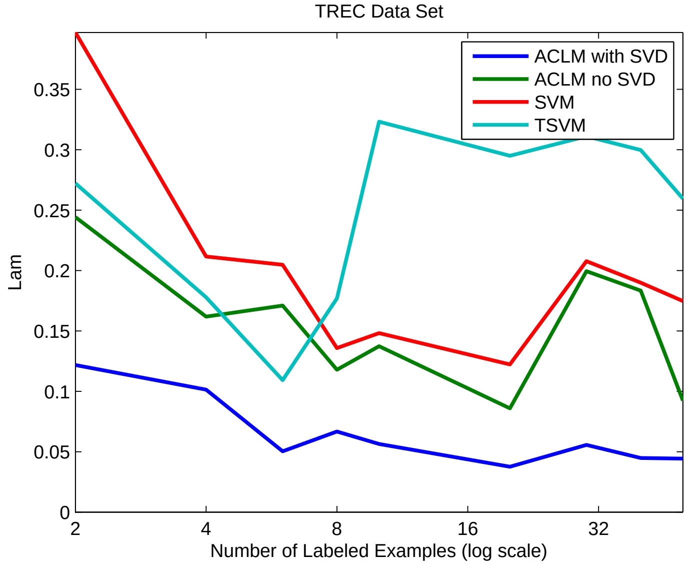


4.6.2 CEAS 2008

The result of the experiments on the CEAS 2008 dataset is illustrated in Figure 4.3. ACLM outperforms all methods when training sizes are between 4 and 32, with the mean error rate of under 15% for all sizes of training sets. SVM has the best error rate only when the training set is of size 8. Transductive SVM seems to have a steady performance over all sizes. Similar to the previous experiment, ACLM without SVD performs badly compared to all other methods.

Figure 4.4 shows LAM results of the four filters on the CEAS 2008 corpus. Better LAM results of TSVM with training sizes below 16 is because TSVM has zero false positives for these training sizes, but this comes at the cost of very high false negatives,

Figure 4.2: Logistic Average Misclassification for ACLM (with SVD and without), SVM, TSVM on TREC 2007 Corpus



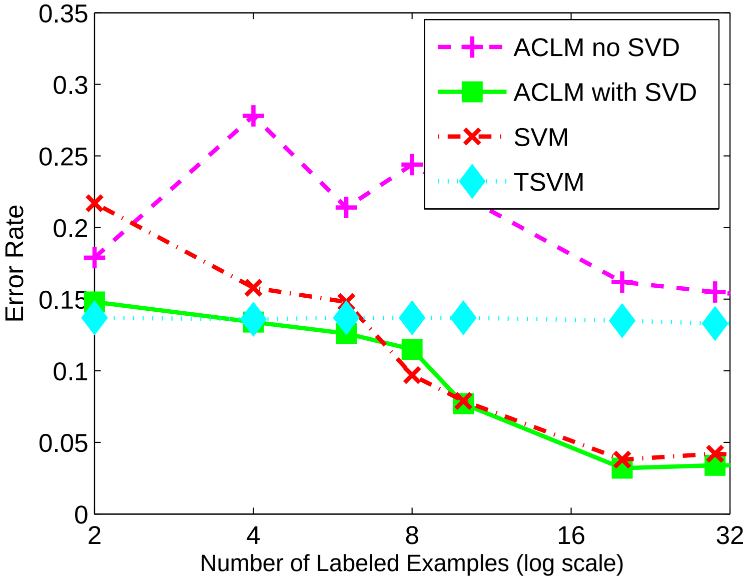
which is not incorporated in the LAM curve. Note that LAM values are smoothed to avoid zero values.

In both experiments, TSVM only performs best with fewer than four examples. We have previously seen similar results in chapter 3 where TSVM was performing better than SVM *only* when the training and test sets were from two completely different sources.

4.7 Contributions and Discussion

In this chapter, we raised the problem of personal spam filters when they are trained only on a few examples. This would help a personal spam filter perform well when the

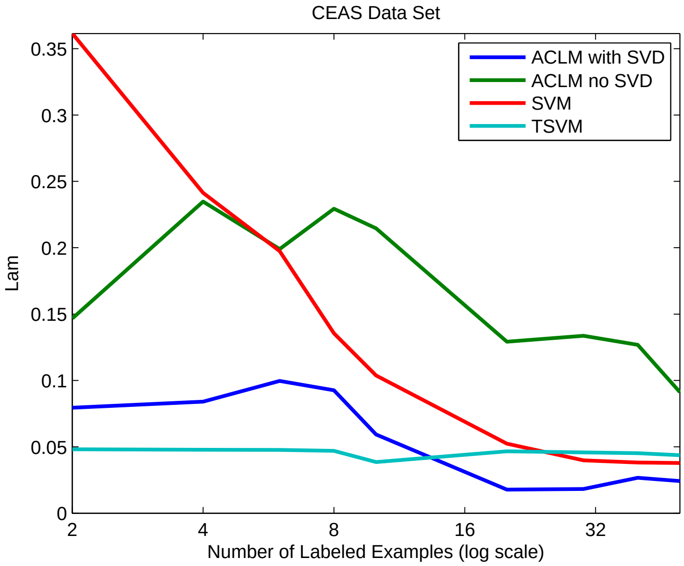
Figure 4.3: Error Rate for ACLM (with SVD and without), SVM, TSVM on CEAS 2008 Corpus



user is willing to provide only a small number of labels to the filter, yet expects the filter to perform reasonably well.

To solve the problem, we adapted and modified a semi-supervised learning algorithm based on global and local consistency. We compared the method with Support Vector Machines in a fully supervised and transductive version. We showed that the Aggressive Consistency Learning Method compares favorably with the competing methods when the number of training examples is very low. This can help spam filters perform better when the user is not willing to, or, cannot provide the filter with many training examples, for example, when a new email client is installed on the user's machine.

Figure 4.4: Logistic Average Misclassification for ACLM (with SVD and without), SVM, TSVM on CEAS 2008 Corpus



Chapter 5

Autonomous Personal Spam Filtering with No User Feedback

5.1 Introduction

In this chapter the difference between personal and global spam filters is discussed with the intention to design a personal spam filter that does not rely on a user for feedback. The key idea behind this chapter is to design a personalized spam filter and train it by the output of a global filter, with no user training at all. This is in consistent with the thesis motivation of building a personal spam filter with minimal user feedback.

A *personal filter* uses characteristics of emails sent to a particular user to classify emails. In other words, personal filters model the characteristics of spam and ham from messages sent to a particular user. A *global filter* employs some combination of hand-crafted rules, pattern-based rules, global whitelists and blacklists, and basically any non-user-specific training examples or collaborative filtering. Note that within this definition, a global filter can learn from other users' emails.

There has been debate on whether personal or global spam filters perform better. The general assumption is that global filters have access to a much broader source of information, and therefore, should perform better. However, several papers indicate the opposite: the best personal filters, of which we are aware, are more accurate than the best global ones.

Previously, it has been illustrated that personal filtering methods yield substantially better accuracy *when trained promptly by the user* [Cormack and Lynam, 2005b, Cormack, 2006b, Cormack, 2007a]. Also, Cormack and Lynam [Cormack and Lynam, 2007] showed that a number of open-source personal spam filters outperform SpamAssassin [spamassassin.org, 2005], a popular filter that combines hand-crafted rules, blacklists and whitelists, and an internal "Bayesian" filter. This performance comes at the cost of users' ideal feedback, a huge burden on the users [Cormack, 2007a, Sculley and Wachman, 2007b]. It is also known that presence of noisy feedback impacts the best-performing personal spam filters employing LR and SVM [Cormack, 2007a, Sculley and Wachman, 2007b].

One of the first controlled studies that has compared personal spam filters and global filters is the CEAS 2008 Live Spam Challenge [cea, 2008a]. It is unique in that it facilitated the comparison, in real-time, of the accuracy of personal and global filters. Email messages from several live sources were combined, assessed, and distributed concurrently to a number of spam filters (personal and global), configured and operated by various participants. In addition, the messages and results were captured, so that the task could be repeated in the laboratory, in simulated real time.

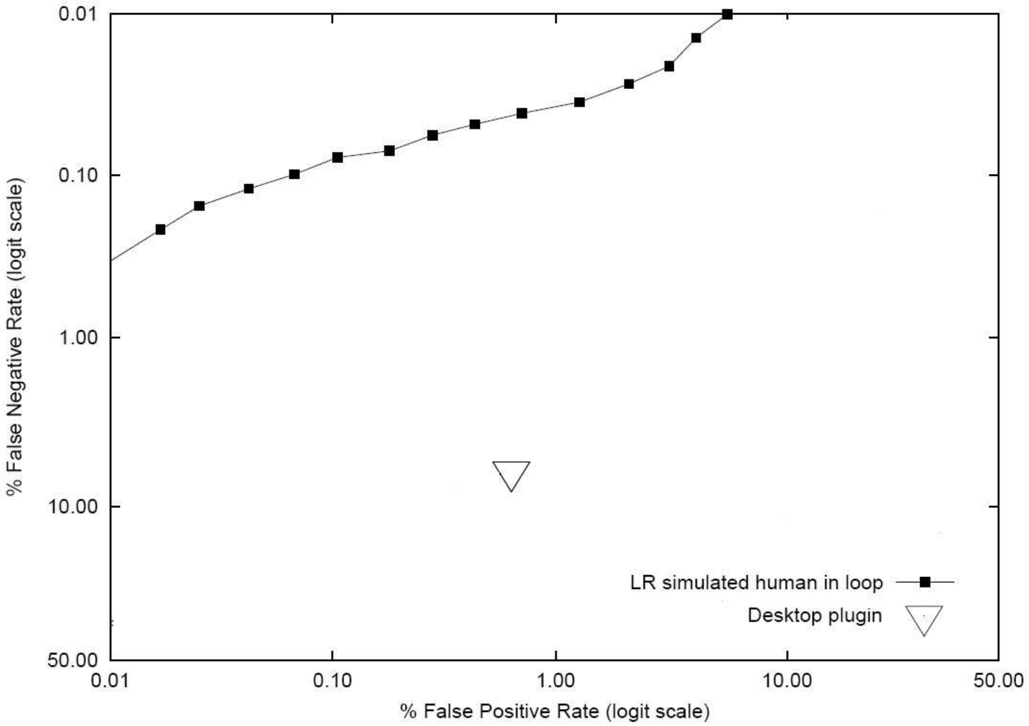


Figure 5.1: The Large Triangle Represents the Performance of a Cloudmark with the top Line Representing the Performance of a Personal Filter on the CEAS 2008 Dataset

To emphasize on the effect of ideal user feedback, Figure 5.1 shows the results of a lab experiment on the CEAS corpus [cea, 2008a], with Cloudmark filter as the baseline. The large triangle represents the performance of a Cloudmark as a commercial collaborative filter plugin applied to the messages in the course of the CEAS 2008 Live Spam Challenge. The top line represents the performance of a personal filter, trained using active on-line active learning on 0.8% of the messages. This line shows that by ideal feedback - i.e. messages (at least the ones that the filter asks for) are labeled, and labeled accurately - the performance of the personal filter is dramatically better than the performance of the Cloudmark.

We should emphasize that personal spam filters yield remarkably high accuracy when they are trained on the ground truth labels of emails. Typically, the user interface solicits feedback only for misclassified messages, or for a small subset of hard-to-classify messages. Therefore, the assumption of ideal feedback is not realistic. Also, the same filter might perform very poorly with noisy feedback. To tackle these problems, this chapter investigates the effect of user feedback on the performance of spam filtering, and supports the following hypothesis:

1. Personalized spam filtering is much more accurate than industry-leading global spam filters;

2. A personalized spam filter, with no user training at all, can perform much better than a global filter, when it is trained by the output of a global filter. We refer to this approach as *Autonomous spam filtering* and investigate it in more detail in section 5.2.

In summary, we show that the noise-tolerant autonomous filters would outperform commercial and open source desktop plugins and a leading commercial web-based spam filter, when trained on the output of those filters.

5.2 Autonomous Personal Filters

As we discussed in the previous section, performance of a personal spam filter relies heavily on the amount and accuracy of user feedback. Obviously, it is unreasonable to ask the user to label each message, or even most messages. The user can be solicited for feedback only for misclassified messages, or for a small subset of hard-to-classify messages. Therefore, having a personal spam filter which does not rely on the user for feedback can be very beneficial.

The most realistic scenario to replace ideal feedback in personal spam filters is having the filter learn in an on-line fashion, classifying each message, and then using it as a training example. It must be mentioned that correcting spam messages that have been classified as ham, is usually done by the user; however, by increasing the number of spam messages, it is more difficult to find a ham message among several spam messages. Moreover, users make mistakes. In fact, we know that among the messages actually labeled by users, error rates of 5% or more are typical [Graham-Cumming, 2006a, Yin et al., 2007].

On the other hand, global spam filters use information from a variety of sources to classify messages, including hard-coded rules, blacklists, and feedback from experts or a community of users. Spam traps are also an important source for commercial global filters. While these sources are independent of any particular user, it has been shown that global filters exhibit higher error rates than personal filters of the order of 5% for the industry-leading filters [Cormack and Mojdeh, 2009a].

This suggests that the global filter's error rate is comparable to the feedback error rate by a typical user of personal filters. Therefore, we propose to use a global filter as a *surrogate user*, providing simulated “feedback” labels to a noise-tolerant personal filter. This implies that if the errors appear as noise to the personal filter, the personal filter will improve on the global filter’s classification. If the errors do not appear as noise, the personal filter will learn to reproduce the errors, yielding no improvement. We call such an approach an *autonomous personal filter* because it is tailored to the users email, but requires no human input.

To confirm our hypothesis, we consider an extensive set of experiments, including a lab experiment in which a combination of personal and global spam filters on two data sets are evaluated, using both a real-time simulation and real-time user study. We also investigate this hypothesis by performing a user study in which we exploit our proposed privacy preserving methodology (Chapter 6) to perform a user study (Chapter 7). In detail, we conduct the following experiments:

1. The first experiment uses the CEAS 2008 email stream [cea, 2008a] to compare a commercial desktop plugin filter, a user-trained personal filter, and several autonomous personal filters.

2. The second experiment exploits a personal email stream of a single user over eight months used to compare the built-in filter of a commercial Webmail system to the same user-trained filter, and several autonomous personal filters.

3. In the third experiment we recruit four independent users of a well known webmail service (Gmail), and ask them to adjudicate the relative effectiveness of the built-

in filter and the best-performing autonomous filter. The methodology used in this user study and the results are described later in Chapters 6 and 7.

5.3 Experiment's Background

Throughout this chapter, we deal with different datasets, and global and personal spam filters. Before going into details of the experiments, we briefly explain the spam filters and datasets used, some of which have been more elaborately described in Section 2.3.

5.3.1 Datasets

CEAS 2008 Live Spam Challenge and Lab Corpus:
The real-time experiment collected and delivered a sequence of email messages over three days to participating filters [cea, 2008c]. The laboratory experiment simulated the real-time experiment after the fact, using the same sequence of messages. The results for each are therefore directly comparable.

During the real-time experiment, an industry-leading desktop plugin spam filter was configured and deployed. Using IMAP, the test messages in real-time were fetched, and the plugin moved those it classified as spam to a junk folder. The results were captured and included in the official CEAS evaluation. Due to technical issues with the setup, about 10% of the messages were never delivered to the plugin, so these messages have been removed for the purpose of comparison. The net result is a sequence of 123,100 messages – 23,844 ham and 99,256 spam – with labels indicating the filter result and also the gold standard as adjudicated by CEAS.

MrX-5:

This is a private corpus and consists of 266,424 messages (6908 ham and 259,516 spam) delivered to a particular user from July 2008 through March 2009. The same user's email has been used for MrX, MrX-2 and MrX-3 datasets reported elsewhere [Cormack, 2008, Cormack and Lynam, 2007].

5.3.2 Spam Filters

Global Filters:

Commercial web spam filter (Gmail): The method used in this filter is undisclosed; we expect it relies heavily on network authentication, blacklists, and the like. It supports multiple authentication systems to be certain that the emails are sent from whom it says it is from. The filter also supplies a user interface to mark email as spam or not, and is advertised as learning from this feedback. It is claimed that with the spam prevalence of 70% in 2007, users reported a rate of less than 1% as spam in their inbox (false positive) [gma, 2007]. However, this interface was not used in our experiments. The resulting error rates (4.2% false positives, 0.77% false negatives) are within the assumed range of 5

Commercial desktop plugin (Cloudmark): This filter uses collaborative filtering and duplicate detection. It maintains a fingerprint of each reported spam message and a near-duplicate detection method determines whether each message has been previously reported as spam. Filter errors arise from two sources: errors in near-duplicate detection and errors or omissions in the database of previously reported spam. Neither of these sources of error is obviously correlated with the content-based features employed by a personal filter. The error rates (0.6% false positives, 6.3% false negatives) are within the assumed range [Sharma and O'Donnell, 2005].

SpamAssassin: The SpamAssassin filter (SA) uses a wide variety of information sources, including blacklists and near-duplicate detection, and also content-based patterns. SpamAssassin includes a "Bayesian" filter, which was not used. SpamAssassin's error rates of 0.25% fpr, 17% fnr are higher than those of the other filters, but still amenable for noise-tolerant methods.

Personal Filters:

We used the best performing spam filter in the CEAS 2008 online active learning experiments, i.e. Logistic Regression (LR) [Sculley, 2007b]. The filter was trained in an active learning scenario. For the CEAS dataset, the filter was allowed to request feedback on a maximum of 1000 messages. For MrX-5, the filter first delivered messages classified as ham to the inbox, and then delivered messages whose classifications were uncertain to a different folder. The user perused this folder occasionally, identifying messages as ham or spam. 3843 messages, (1.4%), were placed in this folder.

Autonomous Personal Filters:

Discriminative filters: We tested two noise tolerant discriminative filters [Cormack and Kolcz, 2009, Sculley and Cormack, 2008]: (i) the Logistic Regression filter, with the learning rate reduced to 0.0005; and (ii) a relaxed on-line support vector machine (ROSVM) [Sculley and Wachman, 2007a]. This filter ranked first together with LR at TREC 2007 Spam Track [Cormack, 2007a]. The only change we made was to decrease the regularization parameter C from 100 to 0.5, a modification known to increase noise tolerance.

Generative filters: We also tested two noise tolerant generative filters [Cormack and Kolcz, 2009, Sculley and Cormack, 2008] (i) Dynamic Markov Compression (DMC); and (ii) our own adaptation of Graham's [Graham, 2004] and Robinson's [Robinson, 2003] Naive Bayes (NB) methods. DMC ranked third in TREC 2007 Spam Track [Cormack, 2007a]. The details of our Naive Bayes method have not previously been reported, and are therefore presented here. We implemented and tuned it some time ago in an effort to discover the essence of Graham's and Robinson's methods. Our NB implementation is, in fact an ensemble formed by summing the results of two NB filters using different tokenization. The first member of the ensemble, NB-4G, uses overlapping 4-grams from the first 3500 bytes of the message, including headers. The second member, NB-W, uses the first 3000 "words", where a word is defined as any sequence of alphabetic and numeric characters. For each message, the filter computes the log-odds-ratio for each token t, based on the number of times t appeared in feedback messages labeled as spam ($t \in spam$) or in feedback messages labeled as ham ($t \in ham$):

$$LogOddsRatio(t) = \log(\frac{|\{t \in spam\}|}{|\{t \notin spam\}|}.\frac{|\{t \notin ham\}|}{|\{t \in ham\}|}) \tag{5.1}$$

The score for NB-4G is the mean of the largest and smallest k values of $LogOddsRatio$, where $k = 30$. Note that the k largest values are typically positive, indicating spam, while the k smallest values are typically negative, indicating ham. The score for NB-W is derived by the same method with $k = 15$. The overall score for our NB is the sum of these two scores.

Fusion of Autonomous and Global Filters:

Given prior results showing the effectiveness of spam filter fusion, we predicted that the fusion of global and autonomous personal filter results might well exceed the performance of either. In order to combine a score from the global filter with the score from the autonomous personal filter, we have to derive a score from the global filter classification. For the Cloudmark and Gmail built-in spam filters, messages classified as spam were given a score of 1, while messages classified as ham were given a score of 0. For SpamAssassin, we added the score reported by SpamAssassin to the score from the autonomous personal filter.

5.3.3 Evaluation Measures

The desktop plugin (Cloudmark) returns just a categorical result for each message - that is if the message is classified as spam or ham. The effectiveness of this filter is determined by single (fpr, fnr). The personal filters that we examine all return a score s - as opposed to the categorical result - for each email. The Receiver Operating Characteristic (ROC) curve is used to evaluate the performance of these filters.

When comparing an ROC curve of a soft classifier with (fpr, fnr) of a hard classifier, a curve that lies above (fpr, fnr) indicates superior effectiveness. *false negative improvement factor*, as described in Section 2.3.4, is used to compare two such filters.

5.4 Lab Experiment

5.4.1 Real-Time Simulation

We used the CEAS 2008 Live Spam Challenge Laboratory Corpus [cea, 2008c]. The results of the best-performing personal filter (LR) in the CEAS on-line active learning laboratory experiment were also captured. Each filter was allowed to solicit user feedback for, at most, 1000 of the messages (about 0.8%). Active learning directly emulates the user interface scenario in which user feedback is solicited from time to time for "hard to call" messages. Although our primary goal was to evaluate autonomous filters, we include the result of this non-autonomous filter to establish its superiority, and to provide a baseline for comparison.

After the fact, we simulated the operation of several autonomous filters on the same stream of messages. For each message in turn, the autonomous filter was used to calculate a "spamminess score" and then the filter was trained as if the results from the commercial filter were truth. That is, if the commercial filter classified the message as spam, we used the message as a positive training example for the autonomous filter; and if the commercial filter classified it as ham, we used it as a negative training example.

5.4.2 Real-Time User Study

Our second email stream, which we dub MrX-5, consists of 266,424 messages – 6908 ham and 259,516 spam – delivered to a particular user from July 2008 through March

2009. While this dataset is private, the authors have archived it, and undertake to test other researchers' filters with it, on request. The same user's email has been used for the MrX, MrX-2 and MrX-3 datasets reported elsewhere [Cormack, 2007b]. The messages in the stream were filtered contemporaneously using LR in exactly the same active learning configuration as CEAS, but with real, rather than simulated, user feedback. That is, the actual recipient was asked to provide feedback on a small fraction of the messages.

In addition, the messages (prior to filtering) were forwarded to Gmail as an industry-leading web mail system. While Gmail has a (somewhat cumbersome) interface for user feedback, it was not used. A list of messages delivered to the inbox was captured from the system, and used to create labels indicating the Gmail filter result.

The same email stream was filtered using the University of Waterloo IT department's SpamAssassin server. The result, which was a numerical "spamminess" score for each message, was captured. This classification is used to train LR, DMC, and NB. The results of all the studies are discussed next.

5.4.3 Results

Figure 5.2 shows CEAS 2008 corpus results, with a Cloudmark filter as the baseline. The large triangle represents the performance of a commercial collaborative filter plugin applied to the messages in the course of the CEAS 2008 Live Spam Challenge. The top line represents the performance of a personal filter, trained using on-line active learning on 0.8% of the messages. The other lines represent the performance of autonomous personal filters, trained exclusively on the output from the baseline filter. Among the autonomous filters, the best performance is achieved by summing the results of Naive Bayes and baseline filters.

As seen in Figure 5.2 and Table 5.1, the baseline filter achieves $fpr = 0.63\%$ and $fnr = 6.64\%$. While these error rates are perhaps higher than advertised, they are competitive with the best global filter results we have observed for the MrX corpora. The personal filter, with 1000 feedback messages, achieves (with suitable choice of t) the same $fpr_t = fnr_t = 0.04$, an improvement factor of $fnif = 150$. We used paired t-test to analyze the significance of the results with the significance level of 0.05.

For our primary hypothesis, the filters of interest are those labeled "autonomous". The ROC curves for all autonomous filters except LR are clearly superior to the baseline, though Naive Bayes (NB) and the ensemble of Naive Bayes and the baseline (NB+Desktop) are substantially better. For all autonomous filters $fnif > 1$ ($p \approx 0.000$). However, it would be misleading to conclude that the LR result ($fnif = 1.1$) represents a substantive improvement over the baseline. In fact, the LR curve very nearly contains the baseline point, indicating that it learned the nature of the baseline filter errors. SVM ($fnif = 1.6$) and DMC ($fnif = 2.1$) show substantive improvement over the baseline, but this improvement is eclipsed by that of NB ($fnif = 22$). The ensemble method NB+Desktop somewhat improves NB ($fnif = 26$). The ensemble method DMC+Desktop in Figure 5.1 dramatically improves DMC ($fnif = 10$), but not to nearly the effectiveness of NB.

Our next experiment involved a long term user study, using the Gmail embedded spam filter and the active learning LR filters in parallel. Figure 5.3 shows MrX-5 Corpus results, with the Gmail filter as the baseline. The large triangle represents the performance of Gmail embedded spam filter when forwarded a stream of personal email in real time. The top line represents the performance of a personal filter, trained by the actual message recipient using on-line active learning on 1.4% of the messages. The other lines represent the performance of autonomous personal filters, trained exclusively on the output from the baseline filter. Among the autonomous filters, the best performance is achieved by summing the results of DMC and baseline filters.

As seen in Figure 5.3 and Table 5.2, the Gmail filter achieves $fpr = 4.2\%$ and $fnr = 0.8\%$. We were surprised by the high false positive rate, and manually-verified, for a one month period, that every false positive (some 30 messages) was in fact delivered to the spam folder. The personal filter, which was trained by the user in the normal course of reading email, achieved $fnr_t = 0.04 (fnif = 20)$. DMC ($fnif = 11$) and $NB (fnif = 5.3)$ both improved on the baseline, but in this case DMC was superior. The ensemble method improves on both DMC ($fnif = 15$) and NB ($fnif = 6.9$). The LR curve once again falls near, but in this case slightly below the baseline ($fnif = 0.8$). We were unable to run SVM due to the size of the dataset.

The third experiment also used the MrX-5 corpus, but with SpamAssassin as the baseline. As seen in Figure 5.4 and Table 5.3, SpamAssassin achieves $fpr = 0.25\%$ and $fnr = 17\%$. The same personal filter results from our second experiment yield $fnr = 0.65\%$ ($fnif = 26$) relative to the SpamAssassin baseline. Performance of the autonomous filters is worse than for the previous two experiments, with all individual filters showing $fnif < 1$. But the ensemble methods both improve on the baseline:

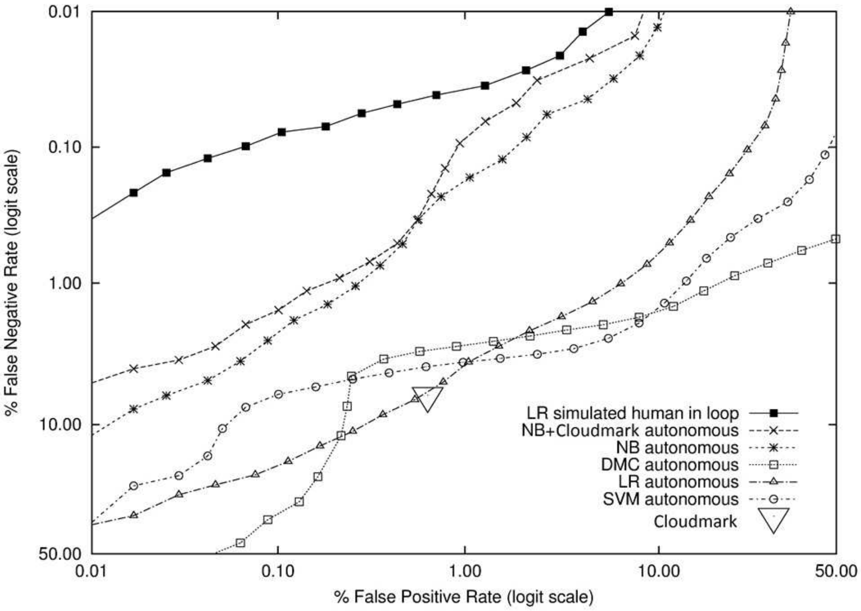


Figure 5.2: CEAS 2008 Corpus Results, with Cloudmark as the Baseline.

Table 5.1: False Negative Improvement Factor for Personal Filters Relative to Cloudmark. All Factors > 1 are Significant ($p \ll 0.01$).

Filter	$\%fpr$	$\%fnr$	$fnif$
LR simulated human	0.63	0.04	149.8
NB+Cloudmark	0.63	0.25	25.6
NB autonomous	0.63	0.29	21.9
DMC+Cloudmark	0.63	0.63	10.1
DMC autonomous	0.63	3.07	2.1
SVM autonomous	0.63	4.00	1.6
LR autonomous	0.63	5.98	1.1
Cloudmark	0.63	6.64	1.0

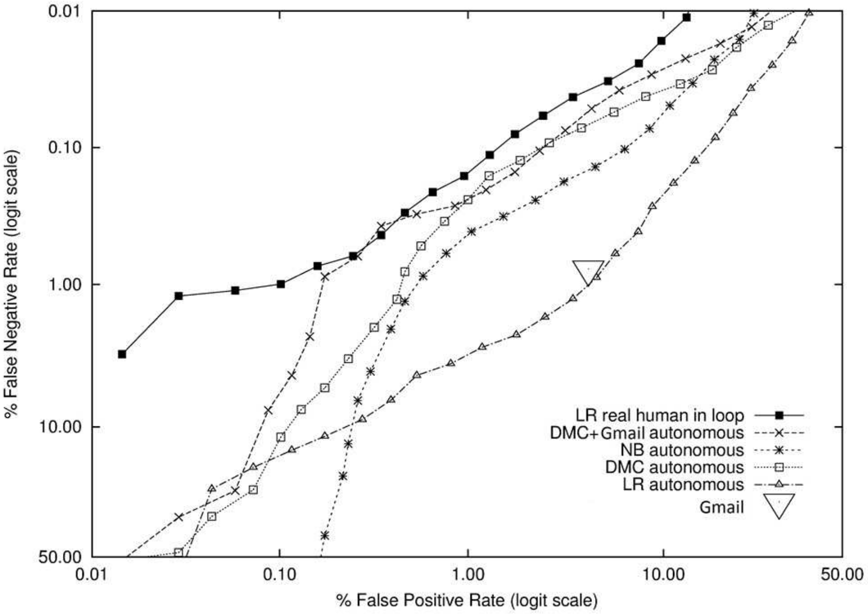


Figure 5.3: MrX-5 Corpus Results, with Gmail as the Baseline.

Table 5.2: False Negative Improvement Factor for Personal Filters Relative to Gmail Built-in Spam Filter, 8-month User Study. All Factors > 1 are Significant ($p \ll 0.01$).

Filter	$\%fpr$	$\%fnr$	$fnif$
LR human in loop	4.24	0.04	20.2
DMC+Gmail	4.24	0.05	14.6
DMC autonomous	4.24	0.07	11.1
NB+Gmail	4.24	0.11	6.9
NB autonomous	4.24	0.14	5.3
Gmail	4.24	0.81	1.0
LR autonomous	4.24 1	.03	0.8

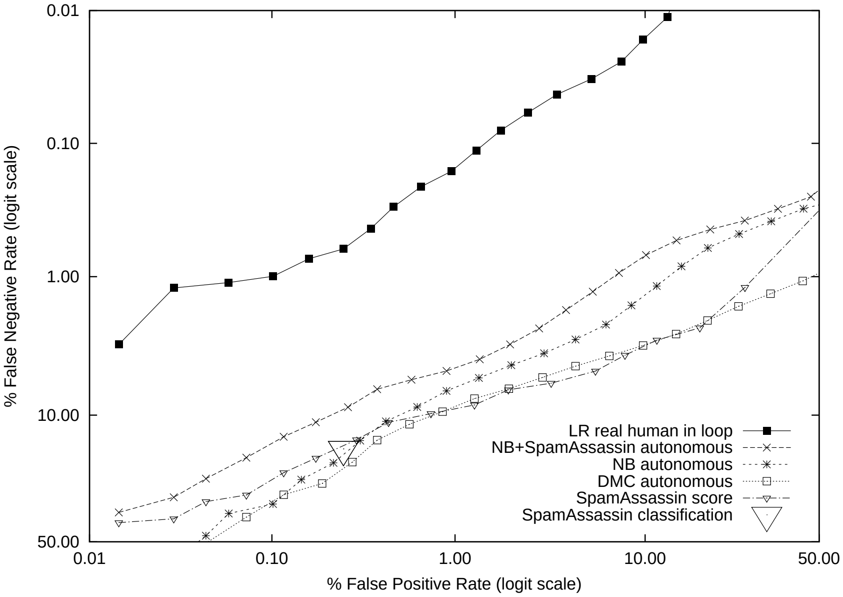


Figure 5.4: MrX-5 Corpus Results, with SpamAssassin as the Baseline.

Table 5.3: False Negative Improvement Factor for Personal Filters Relative to SpamAssassin. All Factors > 1 are Significant ($p \approx 0.000$).

Filter	$\%fpr$	$\%fnr$	$fnif$
LR simulated human	0.25	0.65	25.8
NB+SpamAssassin	0.25	9.13	1.8
DMC+SpamAssassin	0.25	12.59	1.3
SpamAssassin	0.25	16.70	1.0
NB autonomous	0.25	18.86	0.9
DMC autonomous	0.25	23.20	0.7
LR autonomous	0.25	31.36	0.5

NB+SpamAssassin ($fnif = 1.8$), DMC+SpamAssassin ($fnif = 1.3$). These improvements are significant $p \approx 0.000$ and substantive, though not as dramatic as for the previous experiments

5.5 Discussion and Findings

The experiments all support our hypothesis that autonomous personal filters can improve the global filters. Between generative and discriminative filters, generative filters, such as Naive Bayes and DMC, perform better. DMC, in particular, shows high performance.

The experiments overwhelmingly support the hypothesis that a user-trained filter can exceed the performance of a global filter. We note that the number of messages subject to user feedback is about 1% , far fewer than the number of false positives yielded by Cloudmark or the Gmail built-in spam filter. We argue that labeling a few messages is much less burdensome than fetching good email that has gone astray or, for that matter, coping with spam that is delivered to the inbox.

As predicted, we found that discriminative filters like LR and SVM do not work well as autonomous filters, even with noise-tolerant parameter settings. This result is not surprising, as these filters should be able to isolate features that are highly correlated with the global filter's errors, and therefore effectively reproduce the filters incorrect behaviour. Overall, we believe there is substantial latent demand for personal spam filtering, as it is shown to perform very well when trained on user feedback. Users' existing impressions of the accuracy of existing filters, or, the benefit vs. burden of personal filtering, do not incorporate knowledge of the actual error rate, the improvement that might be gleaned from personal filtering, or the ease with which such filtering may be effective, given an ergonomically designed interface. Such an interface would make it easy to find and mark misclassified messages, and show an immediate tangible effect, perhaps by highlighting messages whose classification is revised as a result of training.

Chapter 6

A Toolkit for Privacy-Preserving Spam Filter Evaluation

This chapter describes the toolkit we developed in order to perform privacy-preserving user studies for the evaluation of spam filters. The toolkit is an extension to the Thunderbird Mail Client and supports a standardized evaluation of multiple spam filters on private mail streams. The main advantage of the extension is that researchers need not view or handle the subject users' messages. In addition, subject users need not be familiar with spam filter evaluation methodology. We use this extension in the next chapter to evaluate our proposed autonomous spam filter on real users' email.

All that is required of the user is to install the plugin as a standard extension and to run it on his or her mailbox. The plugin evaluates a spam filter, assuming the user's existing classification is accurate, and sends only the summary results to the researcher, after allowing the user to verify exactly what will be sent. This plugin addresses an outstanding challenge in spam filtering evaluation which is using a broad base of realistic data while satisfying personal and legislative privacy requirements. This chapter ends with preliminary results utilizing the tool to evaluate some filters previously evaluated at TREC [Cormack, 2007a].

This extension and the user study, described later in this chapter, are approved for clearance through the Office of Research Ethics at the University of Waterloo.[1] The intent of the ethics review process is to ensure that research involving human participants at Waterloo is consistent with University of Waterloo's guidelines. This process offers a level of assurance to the research participants, the researchers and the University that the procedures proposed are consistent with the research ethics guidelines, and that the participants will be involved in a consent process which is fully informed and voluntary. More information about this process can be found in Appendix A.

6.1 privacy-preserving Issues in Evaluation of Spam Filters

Preserving the privacy of personal and corporate data has been the subject of increasing attention in both the public and private sectors. These privacy considerations are often at odds with the conduct of large-scale realistic spam filtering efforts. Users are usually

[1] http://iris.uwaterloo.ca/ethics/

very protective about their messages - even the spam - making it almost impossible to carry out user studies. To date, quantitative spam filter evaluation has been conducted using only data sets that may not be representative of the email for which spam filters are actually deployed.

Previous efforts have used public data which may not be representative of personal or corporate email; have captured data which may be insufficiently private; and have used obfuscation techniques which may compromise the integrity of the data and may also be insufficiently private. Ling Spam [Androutsopoulos et al., 2000], for example, is a synthetic corpus consisting of mailing list messages, in which much information is removed from the messages. The PU corpora [Androutsopoulos et al.,] are derived from real email, but obfuscated in a manner that strongly compromises filter performance. Furthermore, in the related field of information retrieval, obfuscation such as that applied to the PU corpora has been found inadequate when preserving privacy [net, 2007]. For example, in 2006, AOL released the data containing search keywords on their website for the duration of 3 months. Though there was no personal identifier in the released dataset, the identities of several searchers were eventually identified [aol, 2006]. The SpamAssassin Corpus [spamassassin.org, 2005] consists of donated messages from heterogeneous sources. It is not easy to acquire a large corpus of such messages. However, they are unrepresentative by virtue of the fact that they are selectively donated.

TREC [TREC, 2005] – perhaps the most realistic comparative evaluation to date – uses both public and private corpora. The public corpora are derived from public sources and hand-crafted to approximate realistic data [Cormack and Lynam, 2005a].

The private corpora consist of data acquired from actual email users; these users allowed the TREC researchers to archive their email for the purpose of evaluation. These private datasets would more correctly be called semi-private.

In summary, the corpora used in spam filtering evaluation suffers from some or all of the following shortcomings:

- They contain private data and they were 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 messages were sampled from different sources.
- Lossy transformations such as tokenization, feature selection, and header stripping were performed.
- Message contents were obfuscated to preserve privacy.

Even though testing with these corpora is valuable as it allows testing different filters and methods under identical circumstances, these corpora 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. On the other hand, it is desirable to extend the evaluation with a realistic and timely sample of real email such that the user privacy is preserved.

Our goal is to provide a better environment to facilitate user studies for email spam filtering, while preserving the users' privacy. To the best of our knowledge, there is very little literature on designing a privacy-preserving user study for the evaluation of spam filters. A user study from the Human Computer Interaction perspective describes work towards helping users better understand the often complex decision making of spam filtering [Lueg and Martin, 2007]. An investigation of a number of popular web-based email services suggests that the filtering process is typically implemented as a black box allowing very little user involvement. The conclusion of the study is that making information about filtering assessments available to users. However, there is no result showing how this feedback can impact and perhaps improve the performance of filtering. Another user study focuses on whitelisting and suggests that whitelisting can be very effective [Erickson et al., 2008]. However, there is no notion on how the user's privacy is considered in either study.

To achieve access to a wider range of representative data we believe it is necessary to use the email of subject users without allowing researchers to read it or otherwise deduce its content. We have designed WATEF, a Thunderbird extension that a volunteer subject can easily install and run to test a filter on his or her email. In this environment, the filter can interact with the user, sender and the community, while preserving the privacy of the volunteer. The results of this run are exactly those defined by the TREC interface: simply a text file with one line per message indicating the true (gold standard) classification, the filter's classification, and the filter's *spamminess score.* To enable researcher to compare different runs of different filters, for each email, the toolkit adds the MD5 hash value. This The volunteer subject is given an opportunity to review the file and to approve (or disapprove) of its being transmitted to the researcher.

In addition, the tool supports the following experimental design: the researcher would solicit subjects. Filters implementing the TREC interface would be encapsulated as WATEF extensions. The users would be instructed to install and run one or more, resulting in the summary files being returned to the researcher by email. These summary files would then be evaluated using the TREC spam filter evaluation toolkit [TREC, 2005].

6.2 The WATEF Extension

WATEF is an extension for Mozilla Thunderbird. Thunderbird is a free, open source, cross-platform e-mail and news client developed by the Mozilla Foundation [mozilla.org, 2004]. An extension, also known as add-on, is a great way to extend the functionality of Mozilla Thunderbird by enhancing the Mozilla Foundation projects [ext, 2011]. Extensions are easily installed and allow users to modify and personalize Mozilla's environment. JavaScript is the primary language of Mozilla browsers.

The WATEF extension works as a wrapper around any spam filter written to conform to the TREC Spam Track format. As explained in section 2.3, the TREC Spam Track uses a standard testing framework that presents a set of chronologically ordered email messages to the spam filter for classification. In the filtering task, the messages are presented one at a time to the filter, which yields a binary judgment (spam or ham). The filter also yields a *spamminess* score, intended to reflect the likelihood that the classified messages are spam [Cormack and Lynam, 2005b].

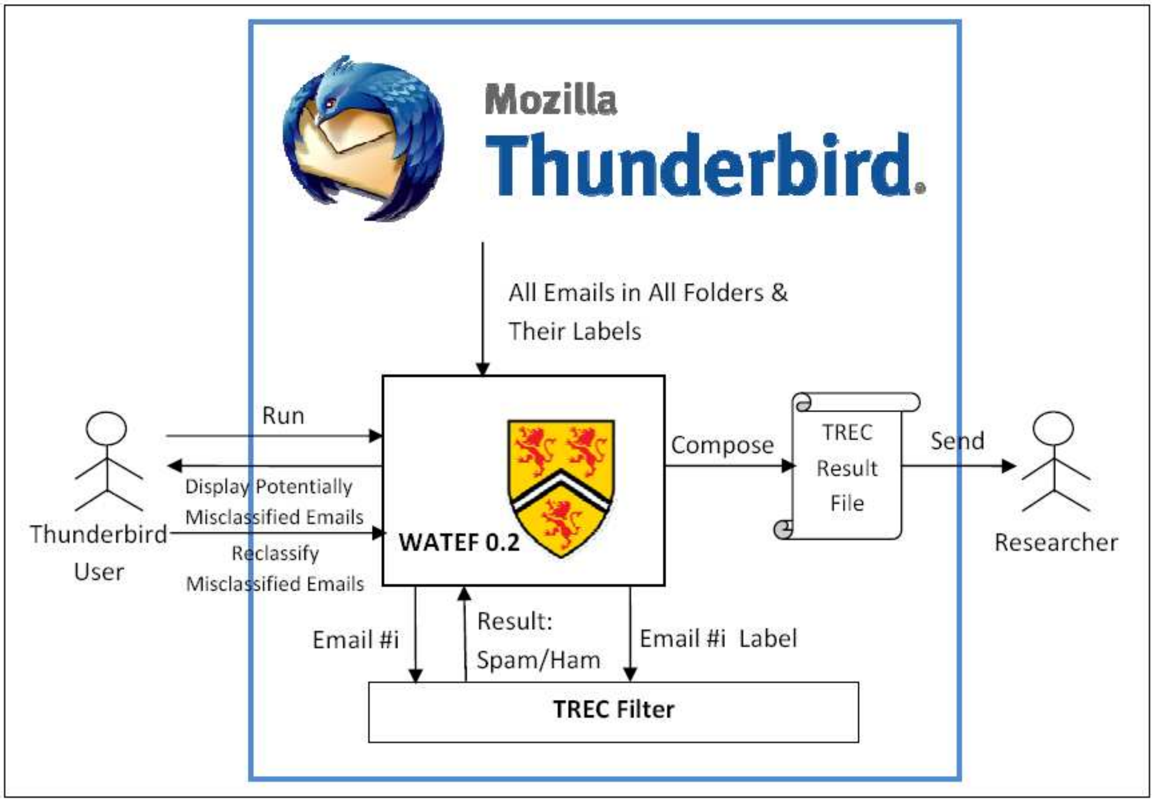


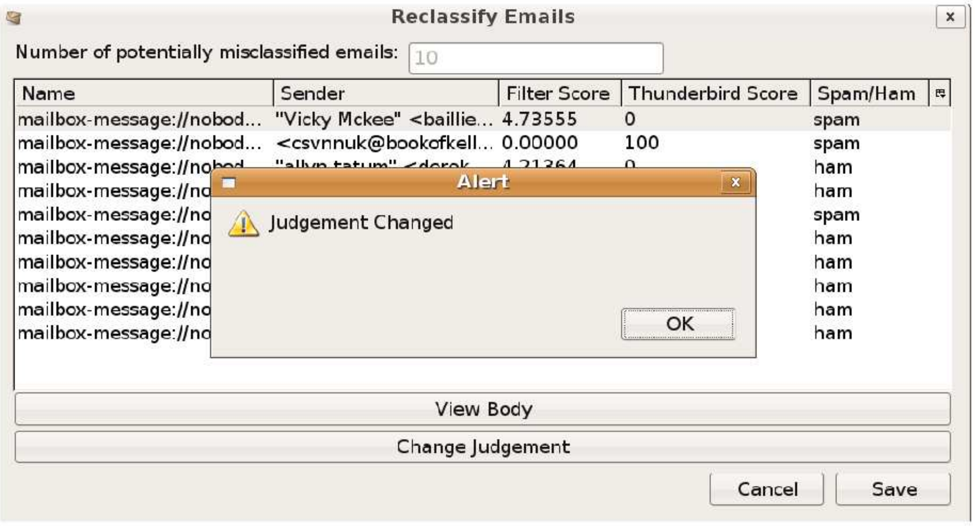
Figure 6.1: Extension Design

The design of WATEF is shown in Figure 6.1. On one side, WATEF can include any TREC-formatted filter, and on the other side, it accesses emails within Thunderbird. When run, WATEF first compiles and initializes the TREC spam filter. Next, it accesses all the folders and emails of Thunderbird through pre-defined components in Mozilla, and passes them one by one, in chronological order, to the filter.

WATEF also gets the judgment of Thunderbird on the emails; this judgment is basically how Thunderbird has labeled the email at the time of running the extension. Thunderbird assigns a *junk score* to each email, which is usually the score the built-in spam filter has assigned. For the first round of running the filter, we treat Thunderbird's judgments as the ground truth labels for the emails. This judgment is revealed to the filter after the classification of the filter is returned to WATEF to simulate the immediate feedback task from the TREC Spam Track. When all the emails are evaluated by the TREC filter, WATEF collects all the results and makes a TREC formatted result file. The result file is emailed to a configurable email address; e.g., the researcher's email address.

Once the filter is run, the volunteer subject is asked to adjudicate cases of disagreement between the filter being tested and Thunderbird's classifications. The adjudicated result is taken to be ground truth. To identify cases of disagreement for adjudication, it is necessary to convert the filter's output from a score to a categorical decision. So, prior to adjudication, the filter's threshold is adjusted to achieve equal apparent ham and spam error rates. We equalize the fraction of ham and the fraction of spam misclassified by the the filter ($fnr = fpr$), considering Thunderbird's classifications to be ground truth.

Figure 6.2 shows how the extension deals with what is believed to be misclassified emails by Thunderbird, asking the subject user to correct them. Basically, the user is shown the sender of the email, the filter's score, Thunderbird's score and Thunderbird's classification (spam or ham). The user can view the body of the email by clicking on it. If the email appears to have incorrect judgments by Thunderbird, the user can change then; this change can be applied to multiple emails at once. The new correct



Reclassify Emails

Number of potentially misclassified emails: 10

Name	Sender	Filter Score	Thunderbird Score	Spam/Ham
mailbox-message://nobod...	"Vicky Mckee" <baillie...	4.73555	0	spam
mailbox-message://nobod...	<csvnnuk@bookofkell...	0.00000	100	spam
mailbox-message://nobod...	"allyn tatum" <derek...	4.21364	0	ham
mailbox-message://no				ham
mailbox-message://no				spam
mailbox-message://no				ham
mailbox-message://no				ham
mailbox-message://no				ham
mailbox-message://no				ham
mailbox-message://no				ham

Alert

Judgement Changed

Figure 6.2: Subject User Gets the Chance to Reclassify Thunderbird's Misclassifications

judgments are then saved within Thunderbird and the filter is run again, this time with the corrected judgments.

Available preference options in the extension are shown in Figure 6.3. The subject user can modify the email address that the final result file is sent to. Also, the number of emails the user wants the filter to classify can be modified. To prevent the filter from being biased towards Thunderbird's labels, the extension, by default, removes the data that has been added to the email by Thunderbird and the subject user can change this option. Finally, the subject user can upload any TREC formatted filter and change the default spam filter.

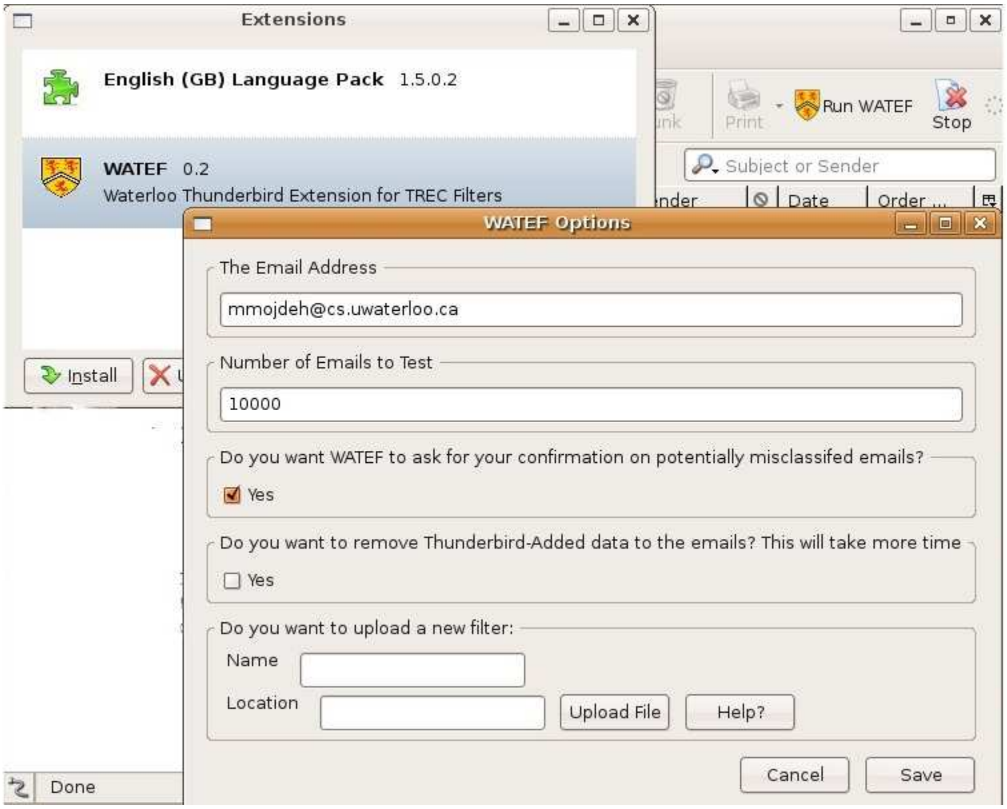
6.2.1 Addressing Privacy Issues

To preserve privacy, the WATEF extension takes the following steps:

- To be able to evaluate a new filter on the same data that previous researchers have used to evaluate their filters, the extension uses MD5 [MD5, 2008] to hash each email. The signature is then added to the TREC result file for each email. MD5 has a 128-bit hash value and is widely used for cryptographic security and data integration.

- The email address that the final email is sent to can be modified in the preference option of WATEF extension. The default value for this address is the author's email address.

- The subject user can easily open the result file before sending the email to make



English (GB) Language Pack 1.5.0.2

WATEF 0.2
Waterloo Thunderbird Extension for TREC Filters

The Email Address
mmojdeh@cs.uwaterloo.ca

Number of Emails to Test
10000

Do you want WATEF to ask for your confirmation on potentially misclassifed emails?
☑ Yes

Do you want to remove Thunderbird-Added data to the emails? This will take more time
□ Yes

Do you want to upload a new filter:
Name
Location

Figure 6.3: Options Provided to the Subject User

sure no private data is sent out. This result file has, for each email, the spamminess score, the filter's classification, and Thunderbird's label, without indicating any sensitive date about the email itself.

- The source code of the extension - in Javascript and XUL - can easily be browsed and investigated.

We have to mention that we are focusing on the fact that the toolkit preserves the privacy of subject users. Since the toolkit is originally designed to evaluate TREC filters, we tend to trust the filter for not violating user's privacy.

6.3 Experiments and Results

In this section, we describe the preliminary results of running the extension on the author's emails. We used three different filters to evaluate the extension: Logistic Regression (LR) [Goodman and tau Yih, 2006], Dynamic Markov Compression (DMC) [Bratko et al., 2006b], LR and DMC Fusion [Cormack, 2007b]. In LR and DMC Fusion, we used on-line logistic regression to fuse the results of the LR and DMC methods giving a very fast algorithm that had the best results at TREC07 and better than any previous filter used at TREC. All methods use overlapping character 4-gram on the first 3500 byte of the raw text of the emails.

The result files were tested using the evaluation script in the TREC Spam Kit. The measure used is the Receiver Operating Characteristic (ROC) Curves, and (1-AUC)(%) as the area above ROC curve, indicating the probability that a random spam message will receive a lower spamminess score than a random ham message [Cormack, 2007a]. Table 6.1 shows the (1-AUC)(%) of the three filters applied on the current author's set of emails in Thunderbird. The number of emails at the time of running the filters was $4,800$. The Table also shows the results of the filters on the TREC'07 Public Corpus as published by the conference [Cormack, 2007a]. Note that LR and DMC Fusion and LR ranked the first two in the TREC'07 in terms of the (1-AUC)(%).

Table 6.1: Results: (1-AUC)(%)

Method	**Thunderbird**	**TREC07 Pub.**
LR	2.4368	0.0057
DMC	3.6107	0.0077
LR & DMC Fusion	2.5877	0.0055

Table 6.2: Results after Correcting Misclassifications

Method	**(1-AUC)(%)**
LR	0.2079
DMC	0.3207
LR & DMC Fusion	0.1222
ROSVM	0.5309

As seen in Table 6.1, the three filters do not perform at their best when run within Thunderbird. Recall that the lower the (1-AUC)(%), the better the filter's performance. The first explanation for this problem is that we are taking the judgments of Thunderbird on each email as the ground truth of the email's classification, while this is not always true.

To show that the relatively high (1-AUC)(%) in Table 6.1 are due to mistakes in the ground truth judgments, we went through all the emails with different filter classification and Thunderbird judgements, and corrected the misjudgments. Table 6.2 shows the results of filters applied on the new modified email judgments. For this Table we have also tested Relaxed Online SVM (ROSVM) filter first introduced by Sculley et al. [Sculley and Wachman, 2007a], which was the best performing filter in the TREC07 after LR and DMC. More details about this filter can be found in Section 2.2.4.

6.4 Difficulties in Developing the User Study

Developing a Thunderbird Extension

We chose Mozilla Thunderbird as it was a relatively popular and free email client with available API for development of an add-on. According to Litmus, in 2010, Thunderbird had a market share of 2.4% among all email clients including Gmail and Yahoo [cli, 2010].

However, developing an extension for Thunderbird had its own challenges. First and most important of all, was the fact that, to the best of our knowledge, there was very little well-formatted documentation on extension development for Mozilla (both

Firefox and Thunderbird). There exists a fair number of add-ons for Thunderbird, but most of them were developed to perform rather simple tasks within Thunderbird environment. Our extension, however, had to access and run an outside program — the filter.

Also, the Thunderbird environment does not appear to be backward compatible for add-ons. We developed the extension for Thunderbird version 2.0.0.24, but Thunderbird versions higher than 3.0 fail to run the extension; the current available version is 3.1.1.17. This added to our difficulty of recruiting participants for our user study, as many users did not have the acceptable version of Thunderbird on their machines, and were not willing to install a new one. We had to ask most users to connect to one of our own machines which had Thunderbird already installed. Many users refuse to run Thunderbird and download their email on any of our machines.

In addition, Thunderbird seems to be slow and sometimes crashes for no apparent reason. This cost us a rather long time to develop and debug the code. Debugging the code, was in fact, another hassle since the only test cases we had in hand were our own email accounts. At times, we noticed specific bugs when a user was participating in the study. We had to cancel the experiment and try to fix the code using our own email sets but the code was running perfectly on own email accounts, making it very difficult to debug.

User Study on Email

Although we had acquired the ethical clearance approval from the Office of Research Ethics at the University of Waterloo, we found recruiting participants for our user study extremely difficult. It seemed to us that users were generally very protective about their emails. This was even true for spam emails, as at some point during our study, we were only looking for spam, and many users still denied any access to their spam emails.

6.5 Summary

This chapter introduced a new approach to evaluate spam filters in more genuine settings. The main objective was to provide a reliable and convenient way to assess filters written in TREC format on a real user mailbox, without asking them to donate their email. We described an extension (or add-on) for Mozilla Thunderbird email client. It can inculde some TREC-formatted spam filter which ran on the user's client machine, and composes a file for TREC Spam Kit. This file does not contain any user-sensitive information. We also showed the results of some preliminary experiments using the extension. To the best of our knowledge this is the first privacy-preserving approach to perform user studies on spam filters. The difficulties involved in development of the toolkit and performing the user study are also discussed.

Chapter 7

User Study

This chapter discusses the user study we performed to evaluate the hypothesis proposed in Chapter 5. The hypothesis suggested that we can build an autonomous personal spam filter that does not require user feedback for training purposes. Following the privacy-preserving user study approach, proposed in Chapter 6, we test this hypothesis on emails belonging to a handful users, and show that the performance of a commercial global filter is improved when a personal learning-based spam filter is trained on the classification of the global filter as the ground truth labels of the emails.

This chapter, first presents the design of the user study, which involved recruiting users of a commercial web mail filter, and running a personal filter on their emails. Users were asked to adjudicate the relative effectiveness of the built-in filter and the best-performing autonomous filter, identified by the previous experiments explained earlier in Chapter 5. The results of the experiment with statistical analysis on users' emails are discussed next.

7.1 User Study Design

For the purpose of the user study we chose Gmail as an industry-leading web mail provider. Gmail was used in a real-time experiment in Section 5.4 where a stream of emails belonging to a single user (MrX-5) was forwarded to it, and a number of personal filters were trained on the result of Gmail afterwards; the results suggested that the autonomous spam filter with Dynamic Markov Compression (DMC), trained exclusively on the classification of Gmail, achieved the best. Because of being the best-performing filter, we chose DMC for this experiment. More description about DMC can be found in Section 2.2.4. We did not consider it feasible to have the user repeat the experiment for other filters, due to crossover effects and the subject effort involved.

Since the user study focuses on the Gmail built-in spam filter, we had to look for subject users with an active email account on the Gmail system. Meanwhile, these users should not have been manually correcting any misclassification of Gmail spam filter, meaning they should not have reported a false positive message as spam, or checked "Filter Messages Like This" on Gmail toolbar. Also they should not have been manually moving emails from regular folders to the Spam folder and vice versa. They also should not have deleted any of their spam emails; knowing that users are more likely to keep their legitimate messages than their spam. Notice that all these limitations made the recruiting process extremely difficult. This is in addition to all the difficulty we had in order to find users who agreed to participate in a user study on their email. Although we had the study cleared through the Office of the Research Ethics at the University of Waterloo, we had a difficult time finding volunteer subjects. It seemed to us that users are very protective about their email and they are not even willing to share their spam emails.

In order to get access to user's mailboxes on Gmail, we followed our proposed privacy-preserving approach described earlier in Section 6, in which users' emails are not revealed to the researcher. We used the Thunderbird extension to perform the study. Each subject was instructed to first install the extension on Thunderbird, to create an IMAP Thunderbird account for Gmail, and to access messages using this account. By disabling Thunderbird's junk mail control, and manually labeling all the emails in the "Spam" folder, which is a copy of Gmail's Spam folder, as Junk, the Thunderbird account will have an exact matching of messages and their labels as Gmail. So, from now on, we may use Thunderbird and Gmail interchangeably. Note that, the usage of the extension for Thunderbird is purely to get access to users' Gmail account while preserving their privacy, and we are not considered about Thunderbird's built-in spam filter at all; which is disabled during the procedure. The detailed set of guidelines provided to users for this study can be found in Appendix A.3.

Next, we asked the participants to run the extension. The extension, as discussed before, can include any TREC-formatted filter on one side, and on the other, it accesses all the messages in Thunderbird through pre-defined components in Mozilla. After all messages are classified by the filter, an email, with the TREC-formatted result file as an attachment, is sent to the researcher. In this case, user's emails are those in their Gmail accounts and the TREC-formatted filter is DMC. The extension then ran the

DMC autonomous filter on the subject's email messages, training messages from the inbox as ham, and from the Spam folder as spam. The results are then sent to the researcher in the format of the TREC result file.

For this user study, we managed to solicit seven users to participate. Since Gmail automatically deletes messages it classifies as spam after one month, we could only investigate messages within the last one month of performing the experiment.

7.1.1 Evaluation Measures

Since Gmail returns only a categorial result for each message, the effectiveness of the filter is determined by a pair (fpr, fnr). For evaluation of DMC, ROC curve is used, which is described elaborately in Section 2.3.4. To compare the performance of Gmail and DMC, as used in the previous experiments in Chapter 5, we use *false negative improvement factor*. For elaborate explanation of $fnif$ refer to Section 2.3.4.

7.2 Results

In this experiment, each user's email had been classified in real-time by the Gmail system and, using the extension, was scored by the autonomous DMC filter, trained exclusively on the output of Gmail. Chi-squared test [Fisher, 1935] is used to evaluate how statistically significance the results are.

Table 7.1 shows the statistics for the user study. Second and third columns show the number of ground truth ham and spam messages in each user's mailbox. Note that the correct label of each message was identified by asking the user to judge all the messages that had different Gmail and DMC classifications, plus some randomly selected messages, as necessary to yield 300 messages. Since DMC returns a score for each message, to get the class of each message for this purpose, the threshold was adjusted such that apparent ham and spam misclassification were equal; that is the threshold that yields $fpr = fnr$. For this threshold, a messages was assigned a class based on its DMC score being above or below the threshold. This classification is compared against Gmail's label and, if different, the message is presented to the user for adjudication through the interface explained earlier in Figure 6.2.

As Table 7.1 shows, the seven participants in the user study had an average number of 634 messages in their mailboxes during the past one month at the time of the experiment. Average number of ham was 242, and average number of spam was 394. Gmail had an average fp and fn of 9 and 48, respectively. More interestingly, Gmail had an average $fpr\% = 4.68$ and average $fnr\% = 12$. Many e-mail systems, which probably includes Gmail, filter blatant spam at the server and never deliver such messages to the user's junk mail folder. The low spam rates seen here may be due to this effect.

The results of our retrospective user study are shown in Table 7.2. All results indicate improvement since $fnif$ values are all greater than 1.00. The results for User 2, User 3 and User 6 are individually significant ($p < 0.05$). With all the 7 users showing improvement, the combined results are also significant ($p < 0.01$). The average $fnif$ over all users is 2.97.

Figures 7.1 to 7.4 show the ROC curves for the retrospective application of the autonomous DMC Filter. In each graph, the point represents Gmail's filter result, while the curve represents the DMC filter result. For all seven users, the ROC curve is superior to the point and so autonomous DMC has outperformed Gmail.

7.3 Conclusion

This chapter examined the hypothesis that it is possible to build an autonomous personal spam filter that does not rely on user feedback, in a retrospective user study. The user study followed our proposed privacy-preserving approach that, to the best of our knowledge, is the first in this field. In this study, researchers had no access to the contents of users emails.

Seven users participated in the user study. The user study supported the hypothesis, and showed that Dynamic Markov Compression, when trained exclusively on the results of Gmail, outperforms the performance of Gmail. Results are shown in terms of *false negative improvement factor*, which for all seven users, showed improvement.

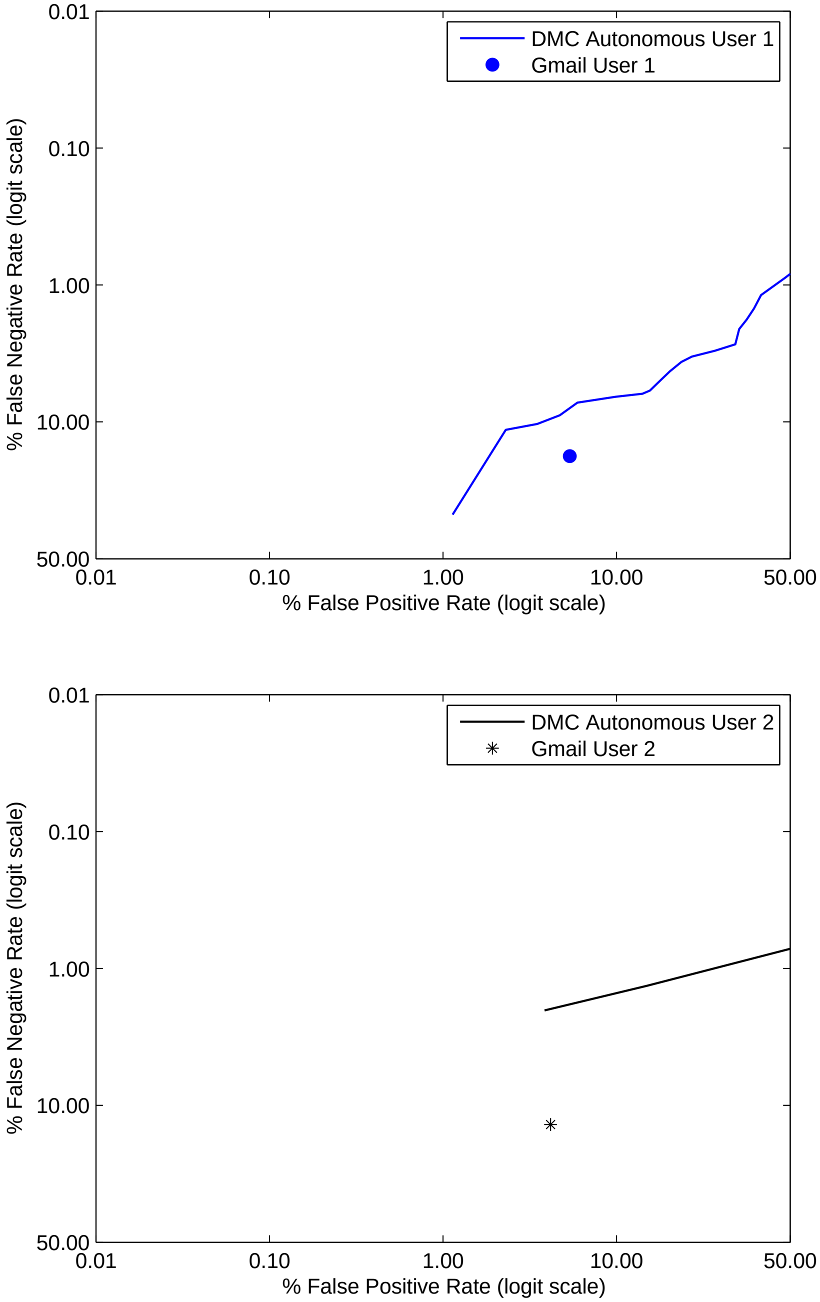


Figure 7.1: Retrospective User Study ROC Curves for User 1 and User 2, Curves Represent Results of the Autonomous DMC Filter. Points represent Gmail's Filter Result.

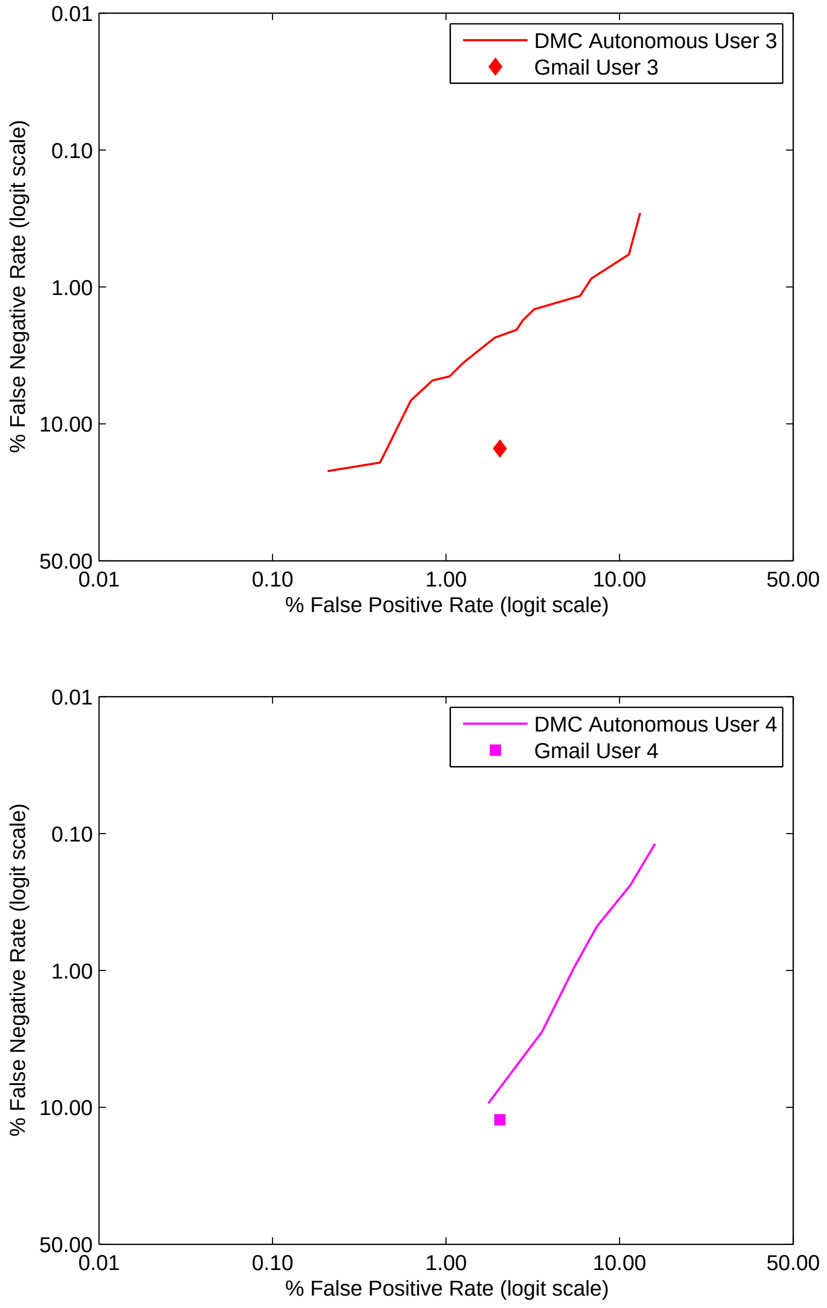
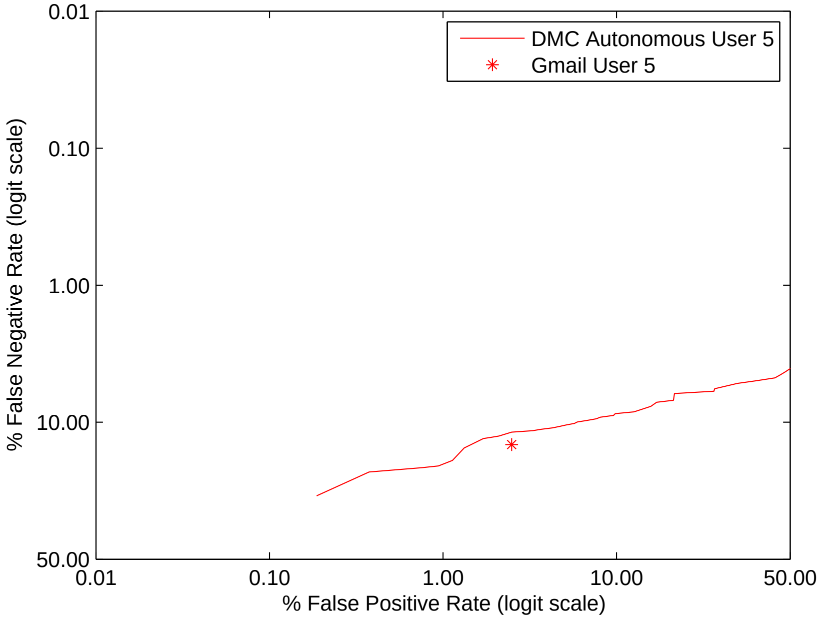


Figure 7.2: Retrospective User Study ROC Curves for User 3 and User 4. Curves Represent Results of the Autonomous DMC Filter. Points Represent Gmail's Filter Result.



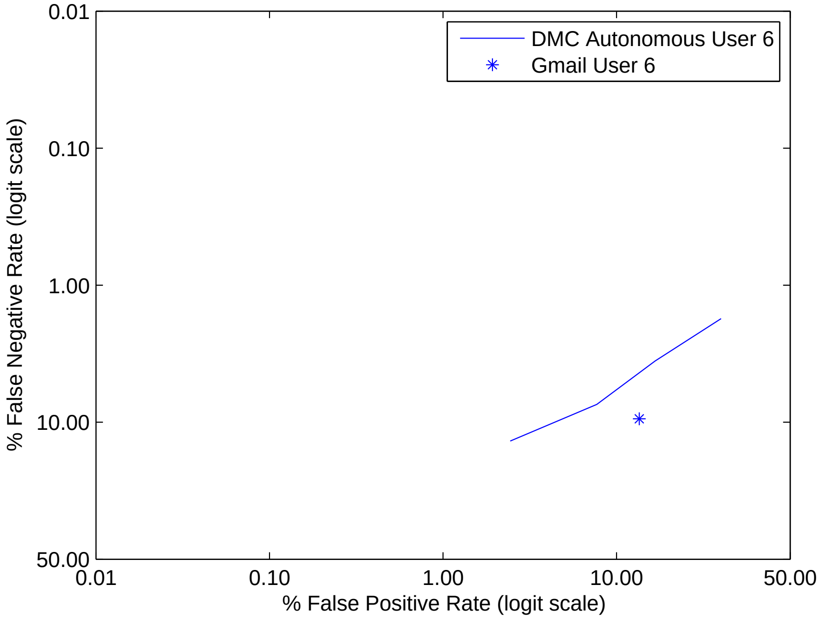


Figure 7.3: Retrospective User Study ROC Curves for User 5 and User 6. Curves Represent Results of the Autonomous DMC Filter. Points Represent Gmail's Filter Result.

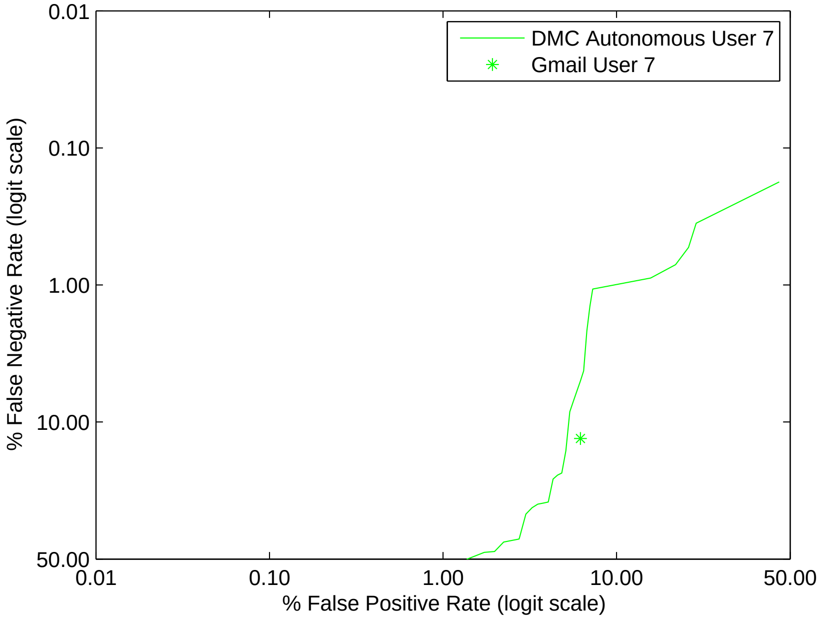


Figure 7.4: Retrospective User Study ROC Curve for User 7. Curve Represent Result of the Autonomous DMC Filter. Point Represents Gmail's Filter Result.

Table 7.1: Gmail Statistics for 7 Users over the 30-day Interval.

User #	Ham	Spam	Total	fp	fn	%fpr	%fnr
User 1	98	341	430	5	51	5.10	14.96
User 2	54	151	205	2	18	3.70	11.92
User 3	482	348	830	10	46	2.07	13.22
User 4	58	843	901	1	92	1.72	10.91
User 5	536	464	1000	13	59	2.43	12.72
User 6	42	58	100	5	5	11.90	8.62
User 7	412	566	978	24	66	5.83	11.66
Average	240	394	634	9	48	4.68	12.00

Table 7.2: False Negative Improvement Factor for DMC Autonomous Personal Filter Relative to Gmail, 1-month Retrospective Study.

User	Filter	$\%fpr$	$\%fnr$	$fnif$
User 1	Gmail	5.10	14.95	1.0
User 1	DMC autonomous	5.10	6.80	2.19
User 2	Gmail	3.70	11.92	1.0
User 2	DMC autonomous	3.70	1.97	6.05
User 3	Gmail	2.07	13.12	1.0
User 3	DMC autonomous	2.07	2.36	5.56
User 4	Gmail	1.72	10.91	1.0
User 4	DMC autonomous	1.72	8.36	1.30
User 5	Gmail	2.43	12.72	1.0
User 5	DMC autonomous	2.43	10.56	1.20
User 6	Gmail	11.90	8.62	1.0
User 6	DMC autonomous	11.90	4.17	2.06
User 7	Gmail	5.83	11.66	1.0
User 7	DMC autonomous	5.83	4.77	2.44
Average	Gmail	4.68	12.00	1.0
Average	DMC autonomous	4.68	5.57	2.97

Chapter 8

Conclusion and Future Work

In this thesis, we discussed the problem of personal spam filtering with minimal user interaction. It has previously been shown that the performance of personal filters, when trained on the ground truth labels of the messages, outperform other global filters. However, personal spam filters rely heavily on users to provide correct labels to emails for training purposes. On the contrary, email clients do not provide users with easy-to-use spam filtering features, and hence, discourage them from helping out spam filters. Users are not motivated to report spam emails in the Inbox, and they barely search the spam folder for legitimate messages. So, it would be unreasonable for personal spam filters to ask for a user's help in correcting misclassified emails. An email client, newly installed on a machine, is another good example of a filter which has to perform well, without having adequate training material from the user.

There have been other attempts to solve this problem and reduce the users' feedback and involvement with the filtering techniques, including learning from user's outgoing email, the Grey folder presented in SpamGuru, discussed in Section 2.2.2, SpamAssassin's auto learning, and collaborative spam filtering.

SpamGuru [Segal et al., 2004] tries to reduce user's involvement by using a server-side spam filter. In this filter, there is a user friendly process called voting, in which users can label messages as spam or ham. A second feature is the addition of a folder called "borderline", which is designed to contain a small set of messages which the anti-spam server is relatively uncertain about. Since this folder also contains borderline ham messages, the user is more inclined to scan it for possible false positives. In summary, this filter aims to provide a more user friendly process to encourage user's participation and also aims to reduce user's feedback by using a server-side filter and providing the new borderline filter. However, user feedback is an important part of it process.

In SpamAssassin [spamassassin.org, 2005], the users' participation is replaced a semi-supervised learning mechanism, called "auto-learning" (see section 2.2.2) which automatically re-trains the Naive Bayes classifier using some unlabeled emails collected during filter operation. The idea behind collaborative spam filtering is that each message is sent to a number of recipients. A certain message could have been received and classified or judged by another user. Collaborative spam filtering is the process of capturing, recording, and querying those early judgments. Therefore, in general, the user's participation is reduced in these techniques by sharing other users' judgment. However, the false positive and false negative rates of this approach depend on both human and technical factors which limit the timeliness, completeness, and accuracy of these three steps

Among other attempts to minimize user interaction with spam filter is when filters try to learn from user's outgoing email [Blanzieri and Bryl, 2008]. SpamGuru also learns from outgoing messages, however, this only partly contributes to spam filtering.

We believe it is necessary to build a personal filter that would not require user cooperation. To have such a filter, we tried different approaches of semi-supervised learning methods in an attempt to have cross-user training. We found, however, that when training and test sets belong to different users, semi-supervised methods do not really help. Next, we took a different perspective, trying to have a spam filter with an acceptable performance when there was only a handful of training examples available. We adapted a graph-based, semi-supervised learning method that would yield a high performance with the presence of very small training data.

We built an autonomous spam filter that does not require user feedback. The filter is made up of a personal filter that is trained exclusively on the result of a global filter. Investigating different types of spam filtering approaches, we discovered that generative filters, as opposed to discriminative filters, perform the best. For the experiments, we tested our hypothesis on a lab and a real-time experiment. We also performed a user study in which we recruited users with active accounts on an industry leading web-based spam filter. We showed the performance of the web-based filter is improved when a personal filter is trained exclusively on its results.

Next, we designed a mechanism allows researchers to conduct a user study on spam filters while preserving user privacy. This mechanism addresses an outstanding challenge in spam filtering evaluation which is using a broad base of realistic data while satisfying personal and legislative privacy requirements. To do that, we developed an extension for the Thunderbird email client that would allow researchers to evaluate any TREC-formatted spam filter on a user's mailbox, without asking the user to donate his/her emails. The main advantage of the extension is that researchers need not view or handle the subject users' messages. In addition, subject users need not be familiar with spam filter evaluation methodology.

We utilized the toolkit in a retrospective user study for evaluating different autonomous spam filters. The proposed privacy-preserving approach, to the best of our knowledge, is the first in this field. We performed a user study in which we recruited users with active accounts on Gmail as an industry leading web-based spam filter. We showed the performance of Gmail built-in filter is improved when a personal filter is trained exclusively on its results. We believe the same approach can also be used for other studies involving evaluation of filters.

8.1 Future Work

The low cost of digital communication has given rise to integration of telecommunication and Internet services into a single device, smart phone, which has computing and networking power of a Personal Computer (PC). In particular, one of the features of smart phones is to provide access to emails wirelessly on the device. This means any smart phone can be compromised the same way as a traditional PC by worms, viruses, or Trojan horses. Therefore, spam filtering has always been one of the main concerns for receiving emails on the device.

While receiving spam emails on PCs is counted as a source of waste for users' time and Internet resources, spam emails on smart phone are also considered as a waste of money since users pay to receive their emails. There are many spam filtering techniques on the server side not to send the spam emails to the device. However, many users prefer to have access to all their emails on the device and spam filtering is performed on their devices as there exists enough computing power on smart phones.

Many of the traditional spam filtering algorithms require a large size training set in order to work properly. This necessitates large amount of computational power, whereas smart phone's battery is limited. Therefore, having techniques that only rely on few training samples, yet with simple algorithms, is always desired. One of the important directions in our future work is to implement the same methodology for personal email spam filtering on wireless devices.

In designing a personal spam filter that works with very few training examples, we noticed in our experiments that by selecting only a set of unlabeled examples we may obtain better results than the case we select the whole set of unlabeled examples. This can be explained by the fact that when we consider the whole set of unlabeled examples, we incorporate a lot of doubtable-quality information. Our future work in this direction include investigating on the effect of unlabeled examples. It is desirable to see how the performance of the filter changes when only a small group of unlabeled examples is incorporated into the training phase and how large or small this group can be.

Another future direction of this work is design of a user study in which users are asked to use a developed IMAP client to access their emails. This way, spam filters could be evaluated in real-time. With an appropriate interface, this study can gather valuable information about user's interaction with email clients in real world. Using a proper user interface, many information can be captured from the user, such as messages the user replied to, the amount of time he spent viewing each message, the messages can get deleted, and even how long it takes the user to delete the message (maybe the shorter the time, the higher the probability that the message is spam). The relevancy of all this information to better filters, can yield to design a spam filter with a UI that is specifically designed with the user behaviour in mind.

Appendix A

User Study Review Procedure and Guidelines

A.1 Ethics Review Process

The intent of the ethics review process is to ensure that all research involving human participants at Waterloo is consistent with UW's Guidelines and the Tri-Council Policy Statement: Ethical Conduct for Research Involving Humans (1998) as well as ethics guidelines of professional associations. This process offers a level of assurance to the research participants, the researchers and the university that the procedures proposed are consistent with these research ethics guidelines, that the rights and welfare of the participants will be protected, and that the participants will be involved in a consent process which is fully informed and voluntary.

The review process is also intended to ensure adequate provisions for protection of individuals privacy as well as confidentiality of information they provide. In addition, the process makes certain that known and anticipated risks associated with the procedures will be adequately communicated to potential volunteers prior to participation, and that these risks are deemed to be outweighed by potential benefits from conducting the research. Procedures used to recruit participants are examined to ensure that participation is voluntary and free of explicit or implicit coercion and that participants are able to withdraw their consent at any time without fear of reprisal or loss of entitlement. The review process is intended to protect participants from risks; however, there are occasions or situations that can place the researcher at risk[1].

A.2 User Study Agreement Letter

Following, is the user agreement letter that subject users were asked to sign before participating in the user study.

Title of Project: Privacy Preserving Spam Filter Evaluation

This study is being conducted by Mona Mojdeh as part of my PhD thesis under the supervision of Professor Gordon V. Cormack, School of Computer Science at the University of Waterloo, Canada. We are conducting a research study about evaluating spam filters on personal mailboxes. We experiment on the standardized evaluation of multiple spam filters on private mail stream while maintaining user privacy. Previous efforts for spam filter evaluation have used public data which may not be representative,

[1] http://iris.uwaterloo.ca/ethics/human/ethicsReview/reviewProcess.htm

captured data which may be insufficiently private, and obfuscation techniques which compromise the integrity of the data and may also be insufficiently private.

To participate in this study, you should be using Gmail and using Thunderbird as your email client. If you decide to volunteer, you will be asked to download and install WATEF, the Thunderbird extension, run it, wait for the Compose window to popup, and then send the email. Your participation in the study should take no longer than thirty minutes. Participation in this study is voluntary. You can withdraw your participation at any time by not submitting your responses. There are no known or anticipated risks from participating in this study. The filter that we run on your mailbox, does not impact, in any way, your emails or your own spam filter. If you don't use Thunderbird, you can still participate by connecting to our machine. We will provide the detail guidelines.

It is important for you to know that any information that you provide will be confidential. All of the data will be summarized and no individual could be identified from these summarized results. Furthermore, the result file sent to us does not contain any private data, or any information that could potentially identify the content of your emails. You can make sure of this, by opening the attachment of the composed email before sending it to us. Meanwhile, the extension contains all the source codes in XML and Javascript as you may want to browse the source code.

If you wish to participate, please visit the study website at www.cs.uwaterloo.ca/m̃mojdeh/
WATEF/. From the main page, download WATEF extension. The website provides

enough instructions for installing and using the extension. The data collected from this study will be accessed only by the two researchers named above and will be maintained on a password-protected computer database in a restricted access area of the university. As well, the data will be electronically archived after completion of the study and maintained for two years after the research study has been completed and any submissions to journals have been completed. Should you have any questions about the study, please contact either Mona Mojdeh, mmojdeh@cs.uwaterloo.ca, DC 3548F, or Gordon V. Cormack, gvcormack@uwaterloo.ca, DC2502. Further, if you would like to receive a copy of the results of this study, please contact either researcher.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, feel free to contact Dr. Susan Sykes, Director, Office of Research Ethics, at 1-519-888-4567 ext. 36005 or by email at ssykes@uwaterloo.ca . Thank you for considering participation in this study.

A.3 User Study Guidelines

This section describes the guidelines we provided to users in order to accomplish the study. We have to mention that since we had difficulties getting users already running Thunderbird, we had to ask them first to connect to our computer with a fresh Thunderbird already installed.

Dear user,

Thank you for participating in this user study! Here are the steps you need to follow:

Connect to our computer:

1. Windows Users:

   You have to connect to our machine first. Use the VNC viewer application or ssh client. Please download VNC Viewer [vnc, 2011] from http://plg.uwaterloo.ca/ mmojdeh/watef and run it. This would allow you to connect to our computer.

   Information for connecting with VNC Client:

   Server: teff21.cs.uwaterloo.ca:1
   Password: I will let you know separately.

2. Linux Users

   You can use the following command to connect to our machine, in this case, let me know so I can give you username and password.

   ```
   ssh -Y watef@teff21.cs.uwaterloo.ca
   ```

Setup Your Account:

1. Thunderbird must be running on the machine. If not, type “sh thunderbird” in the command prompt

2. Right click on Local Folders > Properties > Add account
3. Select “email account” and OK
4. Enter your name and id@gmail.com
5. Type of incoming server: IMAP
6. Incoming server: IMAP.gmail.com
7. Press Next
8. Enter your Gmail account
9. Press Finish
10. Go to Server Settings > Security Settings: select SSL
11. Go to Junk Settings (under the account you just created) > disable “Enable adaptive mail control for this account”, Press OK

Setup Emails:

1. From the main toolbar press “Get Mail”
2. Type your password
3. Select “[Gmail]/All Mail” folder and wait untill all email headers are downloaded. At the bottom of the page it usually says how many emails are downloaded and what the total number of emails is.

4. Go to “[Gmail]/Spam” folder. If you do not see the spam folder under Gmail refresh the page by pressing F5, or just click on your account name.
5. If you still do not see your spam go to File > Subscribe and check the box for spam and press “Get Mail” again.
6. Make sure that all emails in this folder have the *fire* icon, meaning that they are Spam.
7. If not, select all emails in the spam folder and press the Junk button on the main toolbar (this might take some time). The *fire* icon will appear beside all emails.

The Experiment:

1. From the main menu, select Tools > Addons > WATEF 0.2 > Preferences
2. In the Name textbox type your first name > save > close the Preferences
3. From the main toolbar, press “Run Watef”. (This takes between 10 and 30 minutes depending on the number of emails you get every month)
4. The extension has two rounds, during each round, it goes through your emails and classifies them as spam or ham (non spam). After each round, it displays a list of emails for which it thinks that Gmail has wrong classification for Spam/Ham column shows the current class of the email.
5. For each email in the list, press “View Body”, read the text of the email and decide whether it is spam or not. If the current class is not correct, change it by pressing “Change Judgement”.

6. After you double checked all the emails in the list, press “Save Changes” and then OK.

Delete Your Account:

Right click on your account (for example, Mojdeh@gmail) > Properties > Remove Account and delete your account.

References

[gre, 2003] (2003). Greylisting: The next step in the spam control war. http://projects.puremagic.com/greylisting/. 22

[lin, 2003] (2003). Ling-Spam, PU and Enron Corpora. http://www.iit.demokritos.gr/skel/i-config/downloads/. 49

[aol, 2006] (2006). AOL Publishes, Withdraws User Search Data. http://www.securitypronews.com/insiderreports/insider/spn-49-20060807AOLPublishesWithdrawsUserSearchData.html. 123

[ecm, 2006] (2006). European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD. http://www.ecmlpkdd2006.org/challenge.html. 7, 28, 47, 59, 69, 73

[net, 2007] (2007). Researchers Reverse Netflix Anonymization. http://www.securityfocus.com/news/11497. 123

[gma, 2007] (2007). *Webmail Spam filter*. http://mail.google.com/mail/help/fightspam/spamexplained.html. 108

[cea, 2008a] (2008a). The CEAS 2008 Live Spam Challenge. http://www.ceas.cc/2008/challenge. 4, 46, 102, 103, 106

[DOE, 2008] (2008). Certified Senders Alliance. http://www.certified-senders.eu/csa_html/en/index_en.htm. 22

[cea, 2008b] (2008b). Conference on Email and Anti-Spam (CEAS). http://www.ceas.cc/2008/challenge/. 46

[cea, 2008c] (2008c). Conference on Email and Anti-Spam (CEAS2008) Corpus. plg.uwaterloo.ca/ gvcormac/ceascorpus/. 107, 112

[MD5, 2008] (2008). Md5. http://www.fourmilab.ch/md5/. 130

[svm, 2008] (2008). Svm light. http://svmlight.joachims.org/. 42, 48, 70, 92

[cli, 2010] (2010). Email Client Market Share. http://litmus.com/resources/email-client-stats. 134

[cle, 2011] (2011). CleanMail. http://www.cleanmail.com/. 17

[IBM, 2011] (2011). IBM Email Security. http://www-935.ibm.com/services/us/en/it-services/express-managed-e-mail-security.html. 16

[vnc, 2011] (2011). Real VNC. http://www.realvnc.com/index.html. 159

[bri, 2011] (2011). Symantec Brightmail Product Family. http://www.symantec.com/business/products/family.jsp?familyid=brightmail. 16

[ext, 2011] (2011). Thunderbird Extensions. http://www.mozilla.org/projects/ thunderbird/specs/extensions.html. 126

[mda, 2012] (2012). MDaemon Messaging Server. http://www.altn.com/products/mdaemon-email-server-windows/. 17

[rad, 2012] (2012). *Radicati Email Statistics Report 2012* . http://www.radicati.com. 2

[Albrecht et al., 2005] Albrecht, K., Burri, N., and Wattenhofer, R. (2005). Spamato - an extendable spam filter system. In *Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS 2005)*, California, USA. 36

[Alter et al., 2000] Alter, O., Brown, P., and Botstein, D. (2000). Singular value decomposition for genome-wide expression data processing and modeling. In *Proceedings of the National Academy of Sciences*, USA. 8, 83, 90

[Amini et al., 2010] Amini, M. R., Goutte, C., and Usunier, N. (2010). Combining coregularization and consensus-based self-training for multilingual text categorization. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 475–482. 72

[Androutsopoulos et al., 2000] Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., and Spyropoulos, C. D. (2000). An evaluation of naive bayesian anti-spam filtering. *the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML-2000).* 26, 123

[Androutsopoulos et al.,] Androutsopoulos, I., Paliouras, G., and Michelakis, E. Learning to filter unsolicited commercial e-mail. *Technical Report 2004/2, NCSR "Demokritos"*. 123

[Attenberg et al., 2009] Attenberg, J., Weinberger, K., Dasgupta, A., Smola, A., and Zinkevich, M. (2009). Collaborative email-spam filtering with the hashing-trick. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2009)*, Mountain View, CA. 36

[Bailey et al., 2008] Bailey, P., Craswell, N., Soboroff, I., Thomas, P., de Vries, A. P., and Yilmaz, E. (2008). Relevance assessment: are judges exchangeable and does it matter. In *Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, SIGIR '08, pages 667–674, New York, NY, USA. ACM. 18

[Beverly and Sollins, 2008] Beverly, R. and Sollins, K. (2008). Exploiting transport-level characteristics of spam. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 21

[Blanzieri and Bryl, 2008] Blanzieri, E. and Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29:63–92. 10.1007/s10462-009-9109-6. 151

[Bratko et al., 2006a] Bratko, A., Cormack, G. V., Filipic, B., Lynam, T. R., and Zupan, B. (2006a). Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7(Dec):2673–2698. 17, 30

[Bratko et al., 2006b] Bratko, A., Cormack, G. V., R, D., Filipic, B., Chan, P., Lynam, T. R., and Lynam, T. R. (2006b). Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7:2673–2698. 39, 71, 132

[Castillo et al., 2007] Castillo, C., Donato, D., Murdock, V., and Silvestri, F. (2007). Know your neighbors: Web spam detection using the web topology. In *30st ACM SIGIR Conference on Research and Development in Information Retrieval*, Netherlands. 86

[Chang et al., 2008] Chang, M., Yih, W., and McCann, R. (2008). Personalized spam filtering for gray mail. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 19

[Chapelle et al., 2002] Chapelle, O., Weston, J., and Schölkopf, B. (2002). Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 585–592. 84

[Cleary and Witten, 1984] Cleary, J. G. and Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402. 17

[Cormack, 2006a] Cormack, G. V. (2006a). Harnessing unlabeled examples through iterative application of Dynamic Markov Modeling. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) Discovery Challenge Workshop*. 28, 63, 82

[Cormack, 2006b] Cormack, G. V. (2006b). TREC 2006 Spam Track Overview. In *Fifteenth Text REtrieval Conference (TREC-2006)*, Gaithersburg, MD. NIST. 65, 102

[Cormack, 2007a] Cormack, G. V. (2007a). TREC 2007 Spam Track Overview. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD. NIST. 1, 37, 44, 46, 60, 64, 67, 68, 69, 77, 102, 110, 122, 133

[Cormack, 2007b] Cormack, G. V. (2007b). University of Waterloo participation in the TREC 2007 Spam Track. In *Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD. NIST. 113, 132

[Cormack, 2008] Cormack, G. V. (2008). *Email Spam Filtering: A systematic review*, volume 1. 21, 35, 108

[Cormack and Bratko, 2006] Cormack, G. V. and Bratko, A. (2006). Batch and online spam filter evaluation. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, Mountain View, CA. 75

[Cormack et al., 2007] Cormack, G. V., Hidalgo, J. M. G., and Sanz, E. P. (2007). Feature engineering for mobile (SMS) spam filtering. In *30th ACM SIGIR Conference on Research and Development on Information Retrieval*, Amsterdam. 14

[Cormack and Horspool, 1987] Cormack, G. V. and Horspool, R. N. S. (1987). Data compression using dynamic Markov modelling. *The Computer Journal*, 30(6):541–550. 24, 25, 28, 30, 38

[Cormack and Kolcz, 2009] Cormack, G. V. and Kolcz, A. (2009). Spam filter evaluation with imprecise ground truth. In *Proceedings of the 32nd ACM SIGIR Conference on Research and Development on Information Retrieval*, Boston. 19, 109, 110

[Cormack and Lynam, 2005a] Cormack, G. V. and Lynam, T. R. (2005a). Spam corpus creation for TREC. In *Conference on Email and Anti-Spam (CEAS 2005)*, California, USA. 45, 66, 123

[Cormack and Lynam, 2005b] Cormack, G. V. and Lynam, T. R. (2005b). TREC 2005 Spam Track overview. 50, 64, 66, 67, 68, 102, 127

[Cormack and Lynam, 2007] Cormack, G. V. and Lynam, T. R. (2007). On-line supervised spam filter evaluation. *ACM Transactions on Information Systems*, 25(3). 44, 102, 108

[Cormack and Martins da Cruz, 2009] Cormack, G. V. and Martins da Cruz, J.-M. (2009). On the relative age of spam and ham training samples for email filtering. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*, pages 744–745, New York, NY, USA. ACM. 58

[Cormack and Mojdeh, 2009a] Cormack, G. V. and Mojdeh, M. (2009a). Autonomous Personal Filtering Improves Global Spam Filter Performance. In *Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS09)*, Mountain View, CA. 9, 105

[Cormack and Mojdeh, 2009b] Cormack, G. V. and Mojdeh, M. (2009b). Machine learning for information retrieval: Trec 2009 web, relevance feedback and legal tracks.

In *Proceedings 18th Text REtrieval Conference (TREC 2009)*, Gaithersburg, MD. 14, 37

[Cormack et al., 2011] Cormack, G. V., Smucker, M. D., and Clarke, C. L. A. (2011). Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval.*, 14(5):441–465. 37

[Dietrich and Rossow, 2008] Dietrich, C. and Rossow, C. (2008). Empirical research on IP blacklisting. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 21

[Erickson et al., 2008] Erickson, D., Casado, M., and N.McKeown (2008). The effectiveness of whitelisting: a user-study. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 22, 125

[Esquivel et al., 2009] Esquivel, H., Mori, T., and Akella, A. (2009). Router-level spam filtering using TCP fingerprints:architecture and measurement-based evaluation. In *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS 2009)*, Mountain View, CA. 21

[Fisher, 1935] Fisher, R. A. (1935). *The Design of Experiments*. Oliver and Boyd. 140

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37:277–296. 24

[Gao et al., 2008] Gao, J., Fan, W., Jiang, J., and Han, J. (2008). Knowledge transfer via multiple model local structure mapping. In *Proceeding of the 14th ACM SIGKDD* *international conference on Knowledge discovery and data mining*, KDD '08, pages 283–291. 82

[Ghahramani, 2004] Ghahramani, Z. (2004). Unsupervised learning. In *Advanced Lectures in Machine Learning*, pages 72–112. Lecture Notes In Computer Science 3176. 29

[Golbeck and Hendler, 2004] Golbeck, J. and Hendler, J. (2004). Reputation network analysis for email filtering. In *CEAS 2004 – The Conference on Email and Anti-Spam*, Mountain View, CA. 14

[Goodman et al., 2007] Goodman, J., Cormack, G. V., and Heckerman, D. (2007). Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2):24–33. 2

[Goodman and tau Yih, 2006] Goodman, J. and tau Yih, W. (2006). Online discriminative spam filter training. In *The Third Conference on Email and Anti-Spam (CEAS 2006)*, Mountain View, CA. 17, 18, 30, 71, 132

[Graham, 2002] Graham, P. (2002). A plan for spam. http://www.paulgraham.com/spam.html. 50

[Graham, 2004] Graham, P. (2004). Better Bayesian filtering. http://www.paulgraham.com/better.html. 24, 110

[Graham-Cumming, 2005] Graham-Cumming, J. (2005). People and spam. In *The Spam Conference*. 24

[Graham-Cumming, 2006a] Graham-Cumming, J. (2006a). Does Bayesian poisoning exist? *Virus Bulletin*. 105

[Graham-Cumming, 2006b] Graham-Cumming, J. (2006b). SpamOrHam. *Virus Bulletin*. 18

[Hayati et al., 2010] Hayati, P., Potdar, V., and Talevski, A. (2010). Rule-based on-the-fly web spambot detection using action strings. In *Proceedings of the 7th Conference on Email and Anti-Spam (CEAS 2010)*, Redmond, Washington. 23

[He et al., 2007] He, J., Carbonell, J., and Liu, Y. (2007). Graph-based semi-supervised learning as a generative model. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2492–2497, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 82

[He and Gildea, 2007] He, S. and Gildea, D. (2007). Self-training and co-training for semantic role labeling:primary report. (technical report), Department of Computer Science, Rochester University, NY. 72

[Hovold, 2005] Hovold, J. (2005). Naive Bayes spam filtering using word-position-based attributes. In *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, Palo Alto, CA. 26

[Joachims, 1998] Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, pages 137–142. 24, 25

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. 28, 41

[Junejo et al., 2006] Junejo, K. N., Yousaf, M. M., and Karim, A. (2006). A two-pass statistical approach for automatic personalized spam filtering. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) Discovery Challenge Workshop*. 62

[Khardon and Wachman, 2007] Khardon, R. and Wachman, G. (2007). Noise tolerant variants of the perceptron algorithm. *Journal of Machine Learning Research*, 8:227–248. 25

[Kim et al., 2005] Kim, H., Howland, P., and Park, H. (2005). Dimension reduction in text classification with support vector machines. *Journal of Machine Learning Research*, 6:37–53. 92

[Kolcz, 2005] Kolcz, A. (2005). Local sparsity control for Naive Bayes with extreme misclassification costs. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 128–137. 15, 26

[Kolcz and Chowdhury, 2005] Kolcz, A. and Chowdhury, A. (2005). Improved Naive Bayes for extremely skewed misclassification costs. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 561–568. 26

[Kolcz and Chowdhury, 2007] Kolcz, A. and Chowdhury, A. (2007). Hardening fingerprints by context. In *CEAS 2007 – The Third Conference on Email and Anti-Spam*, Mountain View, CA. 35

[Kolcz et al., 2004] Kolcz, A., Chowdhury, A., and Alspector, J. (2004). The impact of feature selection on signature-driven spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA. 14, 15, 35

[Kolcz and Cormack, 2009] Kolcz, A. and Cormack, G. V. (2009). Genre-based decomposition of email class noise. In *Proceedings of the 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 427–436. 20

[Kondor and Lafferty, 2002] Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 315–322, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 85

[Li et al., 2009] Li, K., Zhong, Z., and Ramaswamy, L. (2009). Privacy-aware collaborative spam filtering. *IEEE Transactions on Parallel and Distributed Systems*, 20:725–739. 35

[Li et al., 2008] Li, Y., Guan, C., Li, H., and Chin, Z. (2008). A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system. volume 29, pages 1285 – 1294. 72

[Lueg and Martin, 2007] Lueg, C. and Martin, S. (2007). Users dealing with spam and spam filters: some observations and recommendations. In *Proceedings of the 7th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction: Design Centered HCI*, CHINZ '07, pages 67–72. 125

[Lynam, 2009] Lynam, T. R. (2009). *Spam Filter Improvement Through Measurement*. PhD thesis, University of Waterloo, ON, Canada. 23, 35

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*. 26

[McMillan, 2006] McMillan, R. (October, 2006). US Court threatens Spamhaus with shut down. *InfoWorld*. 47

[Metsis et al., 2006] Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). Naive Bayes – which naive Bayes? In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, Mountain View, CA. 26

[Mojdeh and Cormack, 2008a] Mojdeh, M. and Cormack, G. V. (2008a). A mail client plugin for privacy-preserving spam filter evaluation. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 10

[Mojdeh and Cormack, 2008b] Mojdeh, M. and Cormack, G. V. (2008b). Semi supervised spam filtering: Does it work? In *31st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, Singapore. 8, 61

[Mojdeh and Cormack, 2010] Mojdeh, M. and Cormack, G. V. (2010). Semi-supervised spam filtering using aggressive consistency learning. In *33th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 751–752. 8

[mozilla.org, 2004] mozilla.org (2004). Mozilla. http://www.mozilla.org. 126

[Ng et al., 2001] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856. 89

[Pereira and Gordon, 2006] Pereira, F. and Gordon, G. J. (2006). The support vector decomposition machine. In *ICML*, pages 689–696. 92

[Pfahringer, 2006] Pfahringer, B. (2006). A semi-supervised spam mail detector. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) Discovery Challenge Workshop.* 63, 83, 86, 88, 95, 96

[Pfahringer et al., 2007] Pfahringer, B., Leschi, C., and Reutemann, P. (2007). Scaling up semi-supervised learning: An efficient and effective llgc variant. In *PAKDD*, pages 236–247. 86, 87

[Pitsillidis et al., 2010] Pitsillidis, A., Levchenko, K., Kreibich, C., Kanich, C., Voelker, G. M., Paxson, V., Weaver, N., and Savage, S. (2010). Botnet judo: Fighting spam with itself. In *Network and IT Security Conference (NDSS).* 29

[Prakash and O'Donnell, 2005] Prakash, V. V. and O'Donnell, A. (2005). Fighting spam with reputation systems. *Queue*, 3:36–41. 33

[Qian et al., 2010] Qian, F., Pathak, A., Hu, Y., Morley, Z. M., and Xie, Y. (2010). A case for unsupervised-learning-based spam filtering. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '10, pages 367–368. 29

[Ramachandran et al., 2006] Ramachandran, A., Dagon, D., and Feamster, N. (2006). Can DNS-based blacklists keep up with bots? In *The Second Conference on Email and Anti-Spam (CEAS 2006)*. 21

[Ramachandran and Feamster, 2006] Ramachandran, A. and Feamster, N. (2006). Understanding the network-level behavior of spammers. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 291–302. 14, 21

[Raymond et al., 2004] Raymond, E. S., Relson, D., Andree, M., and Louis, G. (2004). Bogofilter. http://bogofilter.sourceforge.net/. 50

[Robinson, 2002] Robinson, G. (2002). Spam detection. http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html. 24

[Robinson, 2003] Robinson, G. (2003). A statistical approach to the spam problem. http://www.linuxjournal.com/article.php?sid=6467. 110

[Robinson, 2004] Robinson, G. (2004). Gary robinson's spam rants. http://radio.weblogs.com/0101454/categories/spam/. 24

[Sahami et al., 1998] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05. 17, 50

[Sarafijanovic et al., 2008] Sarafijanovic, S., Perez, S., and Boudec, J. L. (2008). Resolving fp-tp conflict in digest-based collaborativespam detection by use of negative selection algorithm. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*, Mountain View, CA. 36

[Schneider, 2003] Schneider, K. M. (2003). A comparison of event models for Naive Bayes anti-spam E-mail filtering. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. 26

[Sculley, 2007a] Sculley, D. (2007a). Online active learning methods for fast label-efficient spam filtering. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007)*, Mountain View, CA. 31

[Sculley, 2007b] Sculley, D. (2007b). Online active learning methods for fast label-efficient spam filtering. In *The Third Conference on Email and Anti-Spam (CEAS07)*, Mountain View, CA. 109

[Sculley and Brodley, 2006] Sculley, D. and Brodley, C. E. (2006). Compression and machine learning: A new perspective on feature space vectors. In *Data Compression Conference (DCC 06)*, pages 332–341, Snowbird. 15

[Sculley and Cormack, 2008] Sculley, D. and Cormack, G. V. (2008). Filtering email spam in the presence of noisy user feedback. In *Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008)*. 19, 109, 110

[Sculley and Wachman, 2007a] Sculley, D. and Wachman, G. M. (2007a). Relaxed online support vector machines for spam filtering. In *30th ACM SIGIR Conference on Research and Development on Information Retrieval*, Amsterdam. 18, 30, 68, 109, 134

[Sculley and Wachman, 2007b] Sculley, D. and Wachman, G. M. (2007b). Relaxed online SVMs in the TREC Spam Filtering Track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC-2007)*, Gaithersburg, MD. NIST. 45, 102

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47. 14

[Segal, 2005] Segal, R. (2005). IBM Spamguru on the trec 2005 spam track. In *Fourteenth Text REtrieval Conference (TREC-2005)*, Gaithersburg, MD. 33

[Segal, 2007] Segal, R. (2007). Combining global and personal anti-spam filtering. In *The Third Conference on Email and Anti-Spam (CEAS 2007)*, Mountain View, CA. 16

[Segal et al., 2004] Segal, R., Crawford, J., Kephart, J., and Leiba, B. (2004). SpamGuru: An enterprise anti-spam filtering system. In *Proc of the First Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA. 32, 150

[Segal et al., 2006] Segal, R., Markowitz, T., and Arnold, W. (2006). Fast uncertainty sampling for labeling large e-mail corpora. In *Proceedings CEAS 2006 – Third Conference on Email and Anti-Spam*, Mountain View, CA. 31

[Sharma and O'Donnell, 2005] Sharma, V. and O'Donnell, A. (November 2005). Fighting spam with reputation systems. *ACM Queue*. 109

[Sochor, 2010] Sochor, T. (2010). Greylisting method analysis in real smtp server environment case study. In Sobh, T., editor, *Innovations and Advances in Computer Sciences and Engineering*, pages 423–427. Springer Netherlands. 22

[Song et al., 2009] Song, Y., Kolcz, A., and Giles, C. L. (2009). Better naive bayes classification for high-precision spam detection. *Softw., Pract. Exper.*, 39(11):1003–1024. 26

[spamassassin.org, 2003] spamassassin.org (2003). *The Spamassassin Public Mail Corpus.* 49

[spamassassin.org, 2005] spamassassin.org (2005). *SpamAssassin.* 23, 102, 123, 150

[TREC, 2005] TREC (2005). The text retrieval conference. http://trec.nist.gov. 123, 126

[Truong et al., 2009] Truong, T. V., Amini, M.-R., and Gallinari, P. (2009). A self-training method for learning to rank with unlabeled data. In *The European Symposium on Artificial Neural Networks.* 72

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory.* Wiley-Interscience. 8, 24, 25, 28, 39, 69, 91

[Voorhees, 2007] Voorhees, E. (2007). Overview of the sixteenth Text REtrieval Conference. In *Proc of the 16th Text REtrieval Conference (TREC 2007)*, Gaithersburg, MD. NIST. 4

[Wang et al., 2008] Wang, B., Spencer, B., Ling, C. X., and Zhang, H. (2008). Semi-supervised self-training for sentence subjectivity classification. In *Proceedings of the Canadian Society for computational studies of intelligence, 21st Conference on Advances in Artificial Intelligence*, Canadian AI'08, pages 344–355, Berlin, Heidelberg. Springer-Verlag. 72

[Wei et al., 2010] Wei, C., Sprague, A., Warner, G., and Skjellum, A. (2010). Identifying new spam domains by hosting ips: Improving domain blacklisting. In *Proceedings of the 7th Conference on Email and Anti-Spam (CEAS 2010)*, Redmond, Washington. 21

[Whissell and Clarke, 2011] Whissell, J. S. and Clarke, C. L. A. (2011). Clustering for semi-supervised spam filtering. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 125–134, New York, NY, USA. ACM. 32

[Xu et al., 2009] Xu, J., Fumera, G., Zhou, F., and Zhou, Z. (2009). Training spamassassin with active semi-supervised learning. In *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS 2009)*, Mountain View, CA. 34

[Yin et al., 2007] Yin, W. T., McCann, R., and Kolcz, A. (2007). Improving spam filtering by detecting gray mail. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS 2007)*, Mountain View, CA. 19, 105

[Zheleva et al., 2008] Zheleva, E., Kolcz, A., and Getoor, L. (2008). Trusting spam reporters: A reporter-based reputation system for email filtering. *ACM Transaction Information Systems*, 27(1). 33

[Zhou et al., 2003] Zhou, D., Bousquet, O., Lal, T., Weston, J., and Scholkopf, B. (2003). Learning with local and global consistency. pages 321–328, New York. MIT Press. in 18th Annual Conference on Neural Information Processing Systems (NIPS 2003). 8, 29, 63, 83, 85, 86, 89, 95, 96

[Zhou et al., 2007] Zhou, D., Burges, C. J. C., and Tao, T. (2007). Transductive link spam detection. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, pages 21–28. 28

[Zhu, 2007] Zhu, X. (2007). *Semi-supervised learning literature survey*, volume TR 1530. University of Wisconsin. 28, 40, 41, 82

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *IN Internatiola Conference on Machine Learning*, pages 912–919. 82, 85, 86