

UNIVERSITY OF SOUTH ALABAMA
SCHOOL OF COMPUTER & INFORMATION SCIENCES

A MULTIPLE INSTANCE LEARNING STRATEGY FOR COMBATING
ADVERSARIAL GOOD WORD ATTACKS ON STATISTICAL SPAM FILTERS

BY

Zachary D. Jorgensen

A Thesis

Submitted to the Graduate Faculty of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Science

July 2008

Approved:

Date:

Chair of Thesis Committee: Dr. Yan Zhou

Member of Committee: Dr. Thomas F. Hain

Member of Committee: Dr. Thomas D. Johnsten

Member of Committee: Dr. Madhuri S. Mulekar

Dean of School of Computer & Information Sciences: Dr. Alec F. Yasinsac

Director of Graduate Studies: Dr. Roy J. Daigle

Dean of Graduate School: Dr. B. Keith Harrison

A MULTIPLE INSTANCE LEARNING STRATEGY FOR COMBATING
ADVERSARIAL GOOD WORD ATTACKS ON STATISTICAL SPAM FILTERS

A Thesis

Submitted to the Graduate Faculty of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Science

by

Zachary D. Jorgensen

B.S., University of South Alabama, 2006

July 2008

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge the people who helped make this thesis possible as well as some of those who played a role in getting me to this point in my life. God knows I could not have done it without them.

I would like to start by thanking my thesis mentor, Dr. Yan Zhou, for her constant support, patience, encouragement and valuable input throughout every stage of this research. It has truly been a pleasure working with her and getting to know her.

Next, I would like to thank the members of my thesis committee for their time and consideration. Their valuable feedback and suggestions on this work through its various stages has been a great help. I would also like to acknowledge my fellow graduate student and friend, Meador Inge, who helped with several aspects of this work.

Without the support of my family, I surely would not be where I am today. I am eternally grateful to have such a wonderful group of people to call my family. Their constant love and support has been such a blessing in my life.

Above all, I thank God for making me who I am and blessing my life with so many wonderful people and opportunities. Without Him, I am merely dust in the wind.

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT.....	vii
CHAPTER 1 INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Problem Statement.....	2
CHAPTER 2 BACKGROUND AND RELATED WORK.....	4
2.1 Single Instance Supervised Learning.....	4
2.2 Multiple Instance Learning.....	6
2.3 Related Work.....	7
CHAPTER 3 METHODOLOGY	13
3.1 Problem Formulation.....	13
3.2 Multiple Instance Bag Creation.....	14
3.2.1 Split-H.....	14
3.2.2 Split-T.....	15
3.2.3 Split-P.....	16
3.2.4 Split-S.....	17
3.3 Multiple Instance Logistic Regression.....	18
3.4 Research Questions.....	20
3.5 Research Hypotheses.....	20
CHAPTER 4 EXPERIMENTAL SETUP	21

4.1	Evaluation Metrics	21
4.2	Experimental Data	23
4.3	Feature Selection and Weighting.....	27
4.4	Good Word List Creation	28
4.5	Threshold Values for Split-Term.....	29
CHAPTER 5 EXPERIMENTAL RESULTS		31
5.1	Experiment 1: Attacking the Test Set.....	31
5.2	Experiment 2: Training on Attacked Spam Messages.....	42
5.3	Potential Attacks on the Splitting Methods	52
CHAPTER 6 CONCLUSIONS AND FUTURE WORK		55
REFERENCES.....		57
APPENDIX: Samples of Good Word Attack.....		60
BIOGRAPHICAL SKETCH		62

LIST OF TABLES

	Page
Table 1 Percentages of the terms divided by the split-T threshold.....	30
Table 2 Change in AUC as the quantity of injected good words is increased.....	37
Table 3 Attacking split-H by fragmenting spam with injected good words.....	52

LIST OF FIGURES

	Page
Figure 1 Preprocessing Example	25
Figure 2 Percentage of emails in each data set that are spam.	26
Figure 3 Effect of number of retained features on f-measure.	28
Figure 4 Change in average recall as quantity of injected good words increases.....	33
Figure 5 Average ROC curves for specific quantities of good words injected.	34
Figure 6 ROC curves for 0 words (a) and 500 good words (b) injected.....	38
Figure 7 Average precision (a) and recall (b) when injecting local good word list.....	42
Figure 8 Change in average recall with 10 good words injected into the training set.....	44
Figure 9 Change in average recall with 20 good words injected into the training set.....	45
Figure 10 Change in average recall with 30 good words injected into the training set....	46
Figure 11 Change in average recall with 40 good words injected into the training set...47	47
Figure 12 Change in average recall with 50 good words injected into the training set....	48
Figure 13 Injecting 10 good words into training set and 0 (a) and 500 (b) into test set. ...	49
Figure 14 Injecting 50 good words into training set and 0 (a) and 500 (b) into test set. ...	50
Figure 15 Attacking training and test sets with entire contents of local good word list...51	51
Figure 16 Comparing effects of the split-H attack and the experiment 1 style attack.....	54
Appendix	
Figure 17 An actual attacked spam email.....	60
Figure 18 Another attacked spam email	61

ABSTRACT

Jorgensen, Zachary D., M.S., University of South Alabama, July 2008. A Multiple Instance Learning Strategy for Combating Adversarial Good Word Attacks on Statistical Spam Filters. Chair of Committee: Dr. Yan Zhou.

Statistical spam filters are known to be vulnerable to adversarial attacks. One of the more common adversarial attacks, known as the good word attack, thwarts spam filters by appending to spam messages sets of “good” words, which are words that are common in legitimate email but rare in spam. This thesis describes a multiple instance learning strategy for filtering email spam in the context of good word attacks. Unlike its single instance counterpart, a multiple instance learning model treats each example of the target concept as a set, or bag, of instances. A bag is labeled positive if at least one of its instances is positive, and negative if all instances in the bag are negative. In the task of spam filtering, each email message is treated as a bag of instances (sets of words). A message is classified as spam if at least one instance in the corresponding bag is spam, and as legitimate if every instance in the bag is legitimate. We show that a classifier using the proposed multiple instance counterattack strategy is more robust to good word attacks than its single instance counterpart and other single instance learners commonly employed in the spam filtering domain.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Modern society has come to depend heavily on the Internet for many daily activities. Email, in particular, has become a major method of communication for many organizations and individuals around the world. In fact, the increasing popularity of email-ready mobile phones, PDA's, and Blackberry devices, as well as the increasing existence of public WiFi hotspots, has made email a ubiquitous medium for communication. Although email has proven to be an extremely convenient and useful tool, it has also proven to be highly susceptible to misuse. The problem of unsolicited bulk email, commonly referred to as spam, has become a significant hindrance to the effectiveness of email. In fact, the Messaging Anti-Abuse Working Group estimated that, from June 2006, spam accounted for as much as 80% of world-wide email [19].

Among the various techniques that have been proposed to combat the spam problem, spam filtering has been in the forefront. Emails can be classified as either legitimate, also known as ham, or spam. Spam filtering involves automated analysis of incoming emails to determine their classification. Emails determined by the filter to be ham are passed into the user's inbox, while those determined to be spam are either deleted or redirected to another location for further analysis by the user.

The majority of current spam filters make use of algorithms based on supervised learning techniques developed in the field of machine learning. Such algorithms use statistical information about the content of emails, e.g., word frequencies, to automatically classify them. A collection of pre-classified emails—a training set—is used to build a statistical model by which new emails can be classified.

As spam filtering techniques advance, spammers are using increasingly sophisticated techniques to circumvent this barrier. As a result, the task of spam filtering has become an example of adversarial learning, where an adversary—the spammer—is actively trying to defeat the spam filter. If we are to control the growing spam problem, we must take into account the adversarial nature of the problem.

One of the most common techniques used by spammers to evade detection by a filter is known as the good word attack [17]. This attack involves appending to spam messages sets of “good words” that are common in ham but rare in spam. The added good words effectively compensate for the spammy portion of the message and allow it to pass through the spam filter (see Appendix for samples of actual attacked emails). The focus of this research is to develop a possible defense strategy to combat this type of adversarial attack.

1.2 Problem Statement

This work proposes an effective strategy for combating adversarial good word attacks on statistical spam filters, since little research exists in the literature on this problem.

Existing spam filters are generally based on single instance supervised learning techniques, and treat each email message as a whole, making such filters susceptible to good word attacks. We present a strategy that considers each email to be a collection of separate parts. We classify an email as spam if it contains at least one part that is spam and as legitimate if it contains only legitimate parts. In this way, a spam message that has had legitimate parts added to it will still be classified as spam. In this thesis we implement this strategy using multiple instance learning as a framework and show that the result is a spam filter which is more robust to adversarial good word attacks than filters based on single instance supervised learning algorithms.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter presents definitions and background information related to this research. Section 2.1 introduces a type of machine learning known as single instance supervised learning and describes its role in current spam filters. Section 2.2 introduces the concept of multiple instance learning, upon which our proposed strategy is based. Finally, section 2.3 presents some of the relevant recent work on spam filtering.

2.1 Single Instance Supervised Learning

Supervised learning [28] is a machine learning technique commonly used for classification problems. It is often associated with data mining, in which the goal is to extract useful patterns from data. The goal of supervised learning is to learn the target concept from a collection of training examples—referred to as *instances*—where each training example is represented as a vector of *attributes*, and a class. The learned concept can be represented in various forms (for example, as a set of classification rules, or as a decision tree) and can be used to assign a class label to previously unseen examples, for which the class is clearly unknown. The type (generally nominal, numeric, or boolean) and number of attributes used to represent the instances depends on the classification problem.

Spam filtering can be expressed naturally as a single instance supervised learning classification problem. An email is an *instance* described by an *attribute vector*, which indicates the presence or absence of specific words. Training set emails also have a known *class* (which can be either ‘spam’ or ‘legitimate’) and must be classified by hand to assure they are correct. The learning algorithm uses the training set to build a binary classification function that can be used to classify unknown emails [32].

One of the most common single instance supervised learning algorithms used in statistical spam filters is naïve Bayes [25]. Spam filters based on the naïve Bayes classifier make use of Baye’s theorem to determine the probability that a given email message is spam. If the calculated probability is above a certain threshold, then the email is considered spam. The probability that an email, represented as an instance $X = \{x_1, x_2, \dots, x_n\}$ where x_1, x_2, \dots, x_n are the attribute values of the instance, is spam is calculated as:

$$\Pr(C = spam | X) = \frac{\Pr(X = \{x_1, x_2, \dots, x_n\} | C = spam) \times \Pr(C = spam)}{\Pr(X = \{x_1, x_2, \dots, x_n\})}$$

where $\Pr(X = \{x_1, x_2, \dots, x_n\} | C = spam)$ is the probability that the given attribute values appear in a spam email, $\Pr(C = spam)$ is the probability of any message being spam, and $\Pr(X = \{x_1, x_2, \dots, x_n\})$ is the probability of the given attribute values appearing in any email message. C in the formula above refers to the class label of the message which can be either *spam* or *legitimate*. The probability that a particular attribute (word) will appear in a spam email message is calculated during the training process, and is stored in a database. The probabilities used in the above formula are obtained from that database.

2.2 Multiple Instance Learning

Multiple instance (MI) learning [9] differs from single instance supervised learning in that an example is represented by a set, or *bag*, of instances rather than as just a single instance. The bag is assigned a class label (either ‘positive’ or ‘negative’) based on the instances it contains; however, the instances within the bag are never explicitly labeled. Classic MI learning assumes that a bag is positive if at least one instance in the bag is positive, and negative if all instances are negative (the precise meanings of ‘positive’ and ‘negative’ depend on the application). Therefore, the goal of multiple instance learning is to learn a classification function that accurately maps a given bag to a class. Formally, let $B = \{B_1, \dots, B_i, \dots, B_m\}$ be a set of bags where $B_i = \{X_{1_i}, X_{2_i}, \dots, X_{j_i}\}$ is the i^{th} bag and $X_{1_i}, X_{2_i}, \dots, X_{j_i}$ are the j instances contained in bag B_i . Note that the value of j may be different for each bag B_i ; classic MI learning makes no assumption about the number of instances in each bag. If B is a training set, then every $B_i \in B$ also has an accurate label $c_i \in C = \{positive, negative\}$ assigned to it. The training process, using B as input, yields a binary classification function $f(B_i): B \rightarrow C$ that maps a bag of unknown class to its proper class.

Our spam filtering strategy adopts the classical MI assumption, which states that a bag is positive if at least one of its instances is positive, and negative if all instances are negative. We treat each email as a bag of instances. Thus, an email is classified as spam if at least one instance in the corresponding bag is spam, and as legitimate if all the instances in it are legitimate. The idea is that by splitting an email into multiple instances, a multiple instance learner will be able to recognize the spam part of the message even if

the message has been injected with good words. In this thesis, we show that a multiple instance learner, combined with an appropriate technique for splitting emails into multiple instance bags, is more robust to good word attacks than its single instance counterpart and other single instance learners that are commonly used in the spam filtering domain.

In Chapter 3, we formalize spam filtering as a multiple instance classification problem in the context of adversarial attacks.

2.3 Related Work

Our work is primarily motivated by recent research on adversarial learning [8], [16], [13]. Dalvi et al. consider classification as a game between classifiers and adversaries in problem domains where adversarial attacks are expected [8]. They model the computation of the adversary's optimal strategy as a constrained optimization problem and approximate its solution based on dynamic programming. Subsequently, an optimal classifier is produced against the optimal adversarial strategy. Their experimental results demonstrate that their game-theoretic approach outperforms traditional classifiers in the spam filtering domain. However, in their adversarial classification framework, they assume both the classifier and the adversary have perfect knowledge of each other, which is unrealistic in practice.

Instead of assuming the adversary has perfect knowledge of the classifier, Lowd and Meek formalize the task of adversarial learning as the process of reverse engineering the classifier [16]. In their Adversarial Classifier Reverse Engineering (ACRE) framework, the adversary aims to identify difficult spam instances—the ones that are

hard to detect by the classifier—through membership queries. The goal is to find a set of negative instances with minimum adversarial cost within a polynomial number of membership queries.

Newsome et al. [20] emphasize the point that training data used to build classifiers for spam filtering and the similar problem of internet worm detection is, to a large extent, controlled by an adversary. They describe and demonstrate several attacks on the generators of such classifiers in which the adversary is able to significantly impair the learning of accurate classifiers by manipulating the training data, even while still providing correct labels for the training instances. The attacks involve inserting features, in a specific manner, into one or both classes of the training data and are specifically designed to cause a significant increase in false positives or false negatives for the resulting classifier. They conclude that the generation of accurate classifiers for adversarial environments should take into account the fact that training data is controlled by an adversarial source.

Barreno et al. [2] explore possible adversarial attacks on machine learning algorithms from multiple perspectives. They present a taxonomy of different types of attacks on machine learning systems. An attack is causative if it targets the training data, and is exploratory if it aims to discover information through, for example, offline analysis. An attack is targeted if it focuses on a small set of points, and is indiscriminate if it targets a general class of points. An integrity attack leads to false negatives, and an availability attack aims to cause (machine learning) system dysfunction by generating many false negatives and false positives. They also discuss several potential defenses

against those attacks, and give a lower bound on the adversary's effort in attacking a naïve learning algorithm.

A practical example of adversarial learning is the technique known as the good word attack. Lowd and Meek [17] present and evaluate several variations of the good word attack. The attack can be carried out in one of two ways: actively or passively. The *active* version of the attack uses feedback from the targeted spam filter to determine which words are deemed good by the filter. Two methods are described for executing these attacks. *First-N Words* involves removing words from a spam email one at a time, appending the removed word to a ham email and submitting it to the filter for classification. This process repeats until the previously-ham email is classified as spam. Dictionary words are then added to this “barely spam” email, one at a time and replacing the previously added word, until the email is classified as ham again. A word that causes the barely spam email to be classified as ham is added to a list. This process repeats until a list of n good words is obtained. *Best-N Words*, works like the former method but in addition to finding a list of good words, it also finds a list of bad words in a similar fashion. Each bad word is used to partition the list of good words into a set of words with smaller weight than the bad word and a set with greater weight. After completing the partitioning process with all of the bad words, the method halts with a list of the best n good words. The active attacks were found to be very effective, measured against current statistical spam filters. Moreover, the execution of such attacks can be quite costly to the spammer, since they require user-level access to the targeted spam filter as well as a large number of queries to the filter to build the lists.

The other type of good word attack discussed in [17] is known as the *passive* good word attack. This attack does not require feedback from the spam filter and instead involves guessing which words the target spam filter may consider to be good. There are three common ways of passively choosing good words. *Dictionary attacks* involve selecting random words from a large collection of words, such as a dictionary. In testing, this method did not prove to be effective; in fact, it actually increased the chances that the email would be correctly classified as spam. *Frequent word attacks* involve the selection of words that occur most often in legitimate sources of content, such as news articles. This method was more effective than the dictionary attack, but it still required as many as 1,000 words to be added to the original message. Finally, *frequency ratio attacks* involve the selection of words that occur very often in legitimate messages but very rarely in spam messages. Tests showed that this technique was quite effective, resulting in a typical spam message being passed off as legitimate by adding as few as 150 words.

Webb, Chitti, and Pu [27] also examine the effectiveness of good word attacks on statistical spam filters. They present a “large-scale evaluation” of the effectiveness of the attack on four spam filters: Naïve Bayes, Support Vector Machine (SVM), LogitBoost, and SpamProbe. Their experiments were performed on a large email corpus consisting of around a million spam and ham messages, which they formed by combining several public and private corpora. Such a large and diverse corpus more closely simulates the environment of a server-level spam filter than a client-level filter. The experimental results show that, on normal email, i.e., email that has not been modified with good words, each of the filters is able to attain accuracy as high as 98%. When testing on “camouflaged messages”, however, the accuracies of the filters drop to between 50% and

75%. In their experiments, spam emails were camouflaged by combining them with portions of legitimate messages. They experimented with camouflaged messages containing twice as much spam content as legitimate content, and vice versa. They also proposed and demonstrated a possible solution to the attack. By training on a collection of emails consisting of half normal and half camouflaged messages, and treating all camouflaged messages as spam, they were able to improve the accuracy of the filters when classifying camouflaged messages.

Our counterattack strategy against good word attacks is inspired by work in the field of multiple instance (MI) learning. The concept of MI learning was initially proposed by Dietterich et al. [9] for predicting drug activities. The challenge of identifying a drug molecule that binds strongly to a target protein is that a drug molecule can have multiple conformations, or shapes. A molecule is positive if at least one of its conformations binds tightly to the target, and negative if none of its conformations bind well to the target. The problem was tackled with an MI model that aims to learn axis-parallel rectangles (APR). Later, learning APR in the multiple instance setting was further studied and proved to be NP-complete by several other researchers in the PAC-learning framework [1], [15], [3].

Several probabilistic models: Diverse Density (DD) [18] and its variation EM-DD [34], and multiple instance logistic regression (MILR) [23], employ a maximum likelihood estimation to solve problems in the MI domain. The original DD algorithm searches for the target concept by finding an area in the feature space with maximum diverse density, that is, an area with a high density of positive points and a low density of negative points. The diverse density at a point in the feature space is defined to “measure

probabilistically how many different positive bags have instances near that point, and how far the negative instances are from that point.” EM-DD combines EM with the DD algorithm to reduce the multiple instance learning problem to a single-instance setting. The algorithm uses EM to estimate the instance in each bag which is most likely to be the one responsible for the label of the bag. The MILR algorithm presented by Ray and Craven [23] is designed to learn linear models in a multiple instance setting. Logistic regression is used to model the posterior probability of the label of each instance in a bag, and the bag level posterior probability is estimated by using softmax to combine the posterior probabilities over the instances of the bag. Similar approaches with different combining functions are presented by Xu and Frank [30].

Many single-instance learning algorithms have been adapted to solve the multiple instance learning problem. For example, Wang and Zucker [26] propose the lazy MI learning algorithms, namely Bayesian-kNN and citation-kNN, which solve the multiple instance learning problem by using the Hausdorff distance to measure the distance between two bags of points in the feature space. Chevaleyre and Zucker [6] propose the multi-instance decision tree ID3-MI and decision rule learner RIPPER-MI by defining a new multiple instance entropy function and a multiple instance coverage function. Other algorithms that have been adapted to multiple instance learning include the neural network MI-NN [22], DD-SVM [5], MI-SVM and mi-SVM [1], multi-instance kernels [11], MI-Ensemble [35], and MI-Boosting [30].

CHAPTER 3

METHODOLOGY

3.1 Problem Formulation

Consider a single instance supervised learning problem with a set of training data $D = \{ \langle X_1, Y_1 \rangle, \dots, \langle X_i, Y_i \rangle, \dots, \langle X_m, Y_m \rangle \}$, where X_i is an instance represented as a single feature vector and $Y_i = C(X_i)$ is the target value of X_i , where C is the target binary classification function. Normally, the task is to learn C given D . The learning task becomes more difficult when there are adversaries that may alter some instance X_i so that $X_i \rightarrow X'_i$ and cause $Y_i \rightarrow Y'_i$, where $Y_i \neq Y'_i$. Let ΔX_i be the difference between X_i and X'_i , that is, $X'_i = X_i + \Delta X_i$. In the case of spam filtering, an adversary can modify spam emails by injecting them with good words. So, ΔX_i represents a set of good words added to a spam message by the spammer. There are two cases that need to be studied separately:

1. the filter is trained on normal emails, that is, emails that have not been injected with good words, and tested on emails which have been injected with good words;
2. both the training and testing sets contain emails injected with good words.

In the first case, the classifier is trained on a clean training set. Predictions made for the altered test instances are highly unreliable. In the second case, the classifier may

capture some adversarial patterns as long as the adversaries consistently follow a particular pattern.

In both cases, the problem becomes trivial if we know exactly how the instances are altered; we could recover the original data and solve the problem as if no instances were altered by the adversary. In reality, knowing exactly how the instances are altered is impossible. Instead, we seek to approximately separate X_i and ΔX_i and treat them as separate instances in a bag. We can then apply multiple instance learning to learn a hypothesis defined over a set of bags. The next section introduces four techniques we have devised for splitting emails into bags of instances.

3.2 Multiple Instance Bag Creation

We propose four different approaches for creating multiple instance bags from emails. We call them split-half (split-H), split-term (split-T), split-projection (split-P), and split subtraction (split-S). We will now discuss each of these splitting methods, in turn.

3.2.1 Split-H

In our first splitting method, *split-half* (split-H), we split every email right down the middle into approximately equal halves. Formally, let $B = \{B_1, \dots, B_i, \dots, B_m\}$ be a set of bags (emails), where $B_i = \{X_{i1}, X_{i2}\}$ is the i^{th} bag, X_{i1} and X_{i2} are the two instances in the i^{th} bag created from the upper half and the lower half of the email respectively. This splitting approach is reasonable in practice because spammers usually append a section of

good words to either the beginning or the end of an email to ensure the legibility of the spam message. As will be discussed later in Section 5.3, this splitting method, because it relies on the positions of words in an email, could potentially be circumvented by the spammer. The next three splitting methods do not rely on the positions of the words and thus do not suffer from that vulnerability.

3.2.2 Split-T

The second splitting method, *split-term* (split-T), partitions a message into three groups of words (terms) depending on whether the word is an indicator of spam, an indicator of ham, or neutral, i.e., $B_i = \{X_{is}, X_{in}, X_{ih}\}$, where X_{is} is the spam-likely instance, X_{in} is the neutral instance, and X_{ih} is the ham-likely instance in bag B_i . The instance to which each word is assigned is based on a weight generated for it during preprocessing. These weights are calculated using word frequencies obtained from the spam and legitimate messages in the training corpus. More specifically, the weight of a term W is given as follows:

$$weight(W) = \frac{p(W | D_s)}{p(W | D_s) + p(W | D_h)},$$

where D_s and D_h are the spam and ham emails in the training set respectively. When splitting an email into instances we used two threshold values, $thresh_s$ and $thresh_\ell$, to determine which instance (spam-likely, ham-likely, or neutral) each word in the email should be assigned to, given its weight. We considered any word with a weight greater than $thresh_s$ to be spammy, any word with a weight less than $thresh_\ell$ to be legitimate, and any word with a weight in between to be neutral. In our experiments, reasonable

threshold values were determined by using cross validation on training emails. Given each training set, $thresh_s$ was selected such that some fraction of the terms chosen during attribute selection (discussed in Section 4.3) would have a weight less than or equal to it. $thresh_\ell$ was selected so that some other fraction of the terms would have a weight greater than or equal to it. The actual threshold values used in our experiments are given later in Section 4.5.

3.2.3 Split-P

The third splitting method, *split-projection* (split-P), transforms each message into a bag of two instances by projecting the message vector onto the spam and ham prototype vectors. The prototype vectors are computed using all the spam and ham messages in the training set. If we view the spam and ham messages in the training set as two clusters, then the prototypes are essentially the centroid of the two clusters. More specifically, let C_s be the set of emails that are spam and C_ℓ be the set of emails that are legitimate. The prototypes are computed using Rocchio's algorithm [24] as follows:

$$P_s = \beta \cdot 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i} - \gamma \cdot 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}$$

$$P_\ell = \beta \cdot 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i} - \gamma \cdot 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i}$$

where C_{s_i} is the i^{th} spam message in C_s and C_{ℓ_i} is the i^{th} ham message in C_ℓ , β is a fixed constant suggested to be 16 and γ is a fixed constant suggested to be 4. Given a

message M , two new instances, M_s and M_ℓ , are formed by projecting M onto P_s and P_ℓ :

$$M_s = \frac{M \times P_s}{|P_s|^2} P_s$$

$$M_\ell = \frac{M \cdot P_\ell}{|P_\ell|^2} P_\ell$$

The rationale of this splitting approach rests on the assumption that a message is close to the spam prototype in terms of cosine similarity if it is indeed spam, and a ham message is close to the ham prototype.

3.2.4 Split-S

The last splitting method, *split-subtraction* (split-S), like the former, uses prototype (centroid) vectors. In this method, however, the ham and spam prototypes are calculated by averaging the corresponding attribute values of all of the ham and spam emails, respectively:

$$P_s = 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i}$$

$$P_\ell = 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}$$

where C_s is a set of spam and C_{s_i} is the i^{th} spam message in C_s ; C_ℓ is a set of ham, and C_{ℓ_i} is the i^{th} ham message in C_ℓ . A message can then be transformed from a single instance attribute vector M into a bag of two instances by subtracting corresponding attribute values in the single instance vector from the ham prototype and the spam

prototype, yielding a legitimate instance $M_\ell = M - P_\ell$ and a spam instance $M_s = M - P_s$, respectively [33].

Now that we have devised several techniques for creating multiple instance bags from email messages, we can transform the standard supervised learning problem of spam filtering into a multiple instance learning problem under the standard MI assumption. For this thesis, we adopt the multiple instance logistic regression (MILR) model to train a spam filter that is more robust to adversarial good word attacks than traditional spam filters based on single instance models. We chose to use the MILR classifier over other MI classifiers mainly because its single instance counter-part, logistic regression (LR), which has been shown to be very effective in the spam filtering domain [31], appeared to be the best among the single instance classifiers considered in our experiments. The next section outlines multiple instance logistic regression.

3.3 Multiple Instance Logistic Regression

Given a set of training bags $B = \{ \langle B_1, Y_1 \rangle, \dots, \langle B_i, Y_i \rangle, \dots, \langle B_m, Y_m \rangle \}$, let $\Pr(Y_i = 1 | B_i)$ be the probability that the i^{th} bag is positive, and $\Pr(Y_i = 0 | B_i)$ be the probability that it is negative. The bag-level binomial log-likelihood function is:

$$L = \sum_{i=1}^m [Y_i \log \Pr(Y_i = 1 | B_i) + (1 - Y_i) \log \Pr(Y_i = 0 | B_i)]$$

In a single instance setting where logistic regression is used, given an example X_i , we model the expected value of the dichotomous outcome of X_i with a sigmoidal response function, i.e., $\Pr(Y_i = 1 | X_i) = \exp(p \cdot X_i + b) / (1 + \exp(p \cdot X_i + b))$, then estimate the

parameters p and b that maximize the log-likelihood function. In a multiple instance setting, we do not have direct measure of bag-level probabilities in the log-likelihood function. However, since individual instances in the bags can also be considered as binary response data, we estimate the instance-level class probabilities $Pr(Y_{ij} = 1 | X_{ij})$ with a sigmoidal response function as follows:

$$Pr(Y_{ij} = 1 | X_{ij}) = \frac{\exp(p \cdot X_{ij} + b)}{1 + \exp(p \cdot X_{ij} + b)},$$

where X_{ij} is the j^{th} instance in the i^{th} bag, and p and b are the parameters that need to be estimated. Thus $Pr(Y_i = 0 | B_i)$ with instance-level class probabilities can be computed as follows:

$$Pr(Y_{ij} = 0 | X_{ij}) = \frac{1}{1 + \exp(p \cdot X_{ij} + b)}.$$

Now we can compute the probability that a bag is negative as:

$$\begin{aligned} Pr(Y_i = 0 | B_i) &= \prod_{j=1}^n Pr(Y_{ij} = 0 | X_{ij}) \\ &= \exp\left(-\sum_{j=1}^n (\log(1 + \exp(p \cdot X_{ij} + b)))\right) \end{aligned}$$

where n is the number of instances in the i^{th} bag. Note that this probability estimate encodes the multiple instance assumption, i.e., a bag is negative if and only if every instance in the bag is negative, and thus the probability estimate

$$Pr(Y_i = 1 | B_i) = 1 - Pr(Y_i = 0 | B_i)$$

encodes that a bag is positive if at least one instance in the bag is positive. In our case, given a set of emails for training, X_{ij} is a vector of the frequency count (or other

variations such as a *tf-idf* weight) of unique terms in each email. We can apply maximum likelihood estimation (MLE) to maximize the bag-level log-likelihood function, and estimate the parameters p and b that maximize the probability of observing the bags in B .

3.4 Research Questions

This thesis is primarily concerned with answering the following research questions:

- Q1:** Does using the proposed multiple instance learning strategy to classify email reduce the effectiveness of an adversarial good word attack on a statistical spam filter trained on normal, i.e., unaltered, email?
- Q2:** Does using the proposed multiple instance learning strategy to classify email reduce the effectiveness of an adversarial good word attack on a statistical spam filter trained on modified, i.e., attacked, email?
- Q3:** Is the effectiveness of the proposed strategy dependent on the specific technique used to split emails into multiple instance bags?

3.5 Research Hypotheses

In this thesis, we verify the following hypotheses:

- H1:** Using the proposed multiple instance learning strategy to classify email will reduce the effectiveness of an adversarial good word attack on a statistical spam filter trained on normal email.
- H2:** Using the proposed multiple instance learning strategy to classify email will reduce the effectiveness of an adversarial good word attack on a statistical spam filter, when training on email modified with good words.
- H3:** The effectiveness of the proposed strategy is heavily dependent on the specific technique used to split emails into multiple instance bags.

CHAPTER 4

EXPERIMENTAL SETUP

We evaluated our multiple instance learning counterattack strategy on emails from the 2006 TREC Public Spam Corpus [7]. Good word attacks were simulated by generating a list of good words from the corpus and injecting them into spam messages in the training and/or test data sets. We compared our counterattack strategy, using the multiple instance logistic regression model and the four splitting methods introduced earlier, to its single instance learning counterpart—logistic regression (LR)—and to the support vector machine (SVM) and the multinomial naïve Bayes (MNB) classifiers. These single instance classifiers are among those most commonly employed in existing statistical spam filters.

4.1 Evaluation Metrics

One important characteristic of spam filtering is that incorrectly classifying a legitimate email is far more costly to the user than incorrectly classifying a spam email. If a spam message makes its way into a user’s inbox, it can be annoying; but if an important legitimate message is accidentally misclassified as spam and the user never gets it, there can be serious consequences. On the other hand, having a spam filter that never misclassifies a legitimate message but misses large amounts of spam defeats the

purpose of having the filter in the first place. This asymmetrical misclassification cost must be considered when evaluating the effectiveness of spam filters.

Throughout the rest of this thesis, we will refer to several evaluation metrics when speaking about the effectiveness of a given spam filter. *Precision* is defined as the fraction of messages correctly classified as spam. A high precision indicates that few legitimate messages are misclassified as spam, i.e., false positives. *Recall* is the fraction of spam messages (out of all the spam messages encountered by the filter) that were identified by the filter as spam. A high recall indicates that few spam emails are misclassified, i.e., false negatives. Ideally, spam filters should have both high precision and high recall, but as a practical matter, high precision has a greater priority. *F-measure* is the weighted harmonic mean of precision and recall and allows us to compare spam filters based on a single value. Formulas for the calculation of these three metrics are given below:

$$Recall = \frac{TP}{FN + TP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where *TP* is the number of true positives, *FP* is the number of false positives, and *FN* is the number of false negatives.

In addition to these three measures, we also present some of our experimental results in the form of *Receiver Operating Characteristics*, or *ROC*, graphs. ROC graphs are two-dimensional graphs with true positive rate on the Y axis and false positive rate on

the X axis. These graphs show the tradeoffs between true positives and false positives and are commonly used to visualize the performance of statistical classifiers [10]. In general, one classifier performs better than another if it produces a ROC curve that is to the northwest of the curve produced by the other on the graph. The total area under a ROC curve (*AUC*) is also commonly used as a metric to compare classifiers. The *AUC* of a spam classifier can be interpreted as the probability that the classifier will rate a randomly chosen spam email as spammier than a randomly chosen legitimate email. Thus a classifier with a higher *AUC* generally performs better than one with a lower *AUC*.

4.2 Experimental Data

Our experimental data consists of 36,674 spam and legitimate email messages from the 2006 TREC Public Spam Corpus. All of the emails in the corpus come in their raw form, with headers, html and other features fully intact. As such, they must be put through a preprocessing stage to reduce them to a form that is suitable for use by the spam filters. Therefore, we preprocessed the entire corpus using the following standard preprocessing techniques:

- Partial stripping of email headers—the *to*, *from*, *cc*, *subject*, and *received* headers were retained, while the remaining headers were discarded.
- Stripping of HTML tags—HTML tags, when present, were fully stripped from each email message.
- Discarding of attachments—email attachments, when present, were discarded.
- Stemming of all words—the stemming process essentially strips the suffix from each word, e.g., the words “mail”, “mailer”, and “mailing” would all be reduced to “mail”. This is done to reduce the number of terms considered by the filter to a manageable number and greatly improves the time and space efficiency of the

filtering process. We chose to use the popular Porter Stemming Algorithm [21] for the stemming process.

- Removing stop words—stop words are words that are extremely frequent in the English language. Our list of stop words contains a little over 300 words. We can safely remove these words because they are so common that they generally appear in spam messages just as often as they do in legitimate messages. Again, we do this to reduce the number of terms that the filter must consider.

Messages that contained an empty body after preprocessing were discarded.

Tokenization was done by splitting on nonalphanumeric characters. We did not take any measures to counter obfuscated words in the spam messages, as that is out of the scope of this thesis. Given that there are a large number of possible ways to disguise a word, most content-based spam filters are not able to deobfuscate the text of a message efficiently [4]. Recently, an efficient complementary filter [14] has been demonstrated to be able to effectively deobfuscate text with high accuracy. In practice, this type of technique could be used during preprocessing. Figure 1 shows a sample spam email message before and after preprocessing.

```
From svzxsx@wanadoo.fr Wed Sep 25 10:29:10 2002
Return-Path: <svzxsx@wanadoo.fr>
Delivered-To: zzzz@localhost.spamassassin.taint.org
Received: from localhost (jalapeno [127.0.0.1])
```

```
.
      [OMITTED FOR BREVITY]
.
```

```
To: <body@katomic.com>
From: "Mandy Barton" <svzxsx@wanadoo.fr>
Subject: are you in the mood XGHTMTGCC
Date: Tue, 24 Sep 2002 18:59:51 -1600
```

```
<HTML>
<HEAD>
</HEAD>
<BODY>
VIAGRA, XENICAL, VIOXX, ZYBAN, PROPECIA &reg;<BR>
We only offer the real VIAGRA, XENICAL, VIOXX, ZYBAN, PROPECIA &reg;<br>
No Herbal Supplements at a much reduced rate found anyplace on<br>
the Internet.all orders shipped discreetly and swiftly to your door<BR>
<A HREF=3D"http://www.great-meds.com/main2.php?rx=3D18148">Enter here for =
pricing and info</A>
<br>
<br>
<br>
<br>
<br>
<br>
<A HREF=3D"http://greenzer.com/remove.php"><FONT SIZE=3D"1" COLOR=3D"#FFFF=
CC">Enter this link to abandon future offer's?</FONT></A>
</BODY>
</HTML>
```

a) email before preprocessing

```
viagra xenic vioxx zyban propecia @
offer real viagra xenic vioxx zyban propecia @
herbal supplement reduc rate anyplac
internet order ship discreetli swiftli door
enter
price info

enter link abandon futur offer
```

b) email after preprocessing

Figure 1 Preprocessing Example

For our experiments we sorted the emails in the corpus chronologically by receiving date and evenly divided them into 11 subsets $\{D_1, \dots, D_{11}\}$. In other words, the messages in subset n were received chronologically before the messages in subset $n+1$. Experiments were run in an on-line fashion, i.e., we trained on subset n and tested on subset $n+1$. Each subset contains approximately 3300 messages. The percentage of spam messages in each subset varies as in the operational setting (see Figure 2). We used the Multiple Instance Learning Tool Kit (MILK) [29] implementation of MILR and the Weka 3.4.7 [28] implementations of LR, SVM and multinomial naïve Bayes, in our experiments.

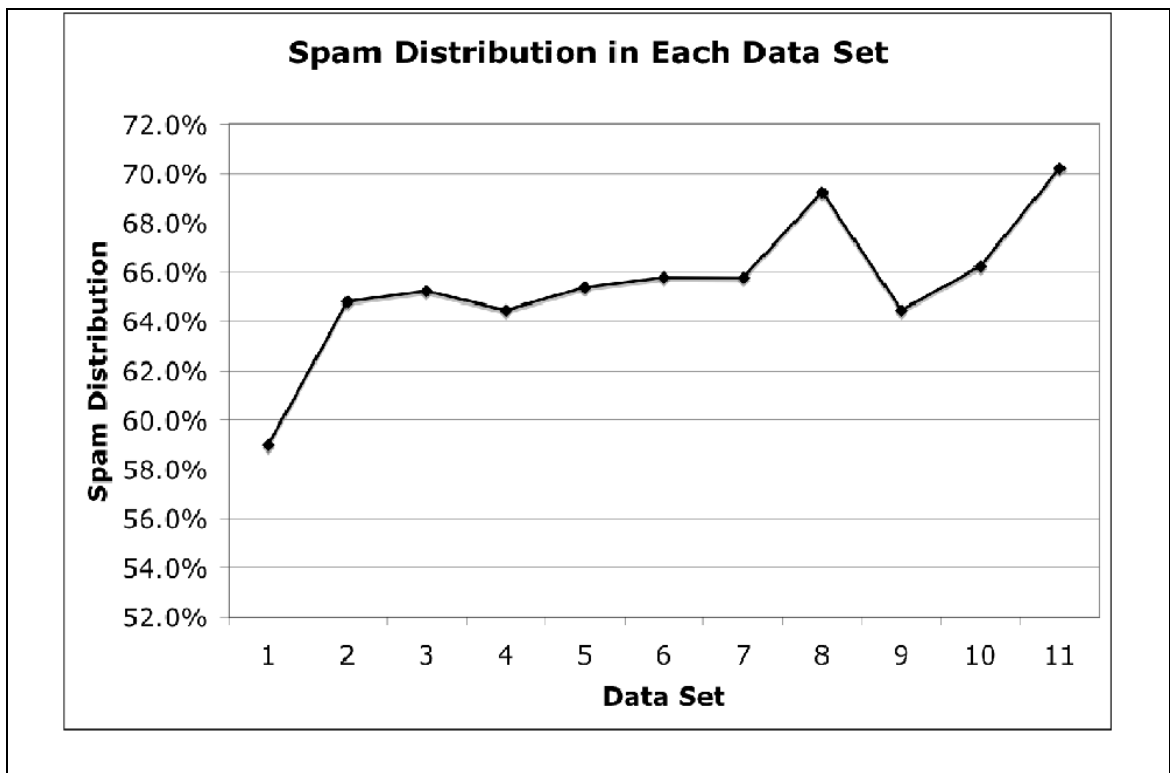


Figure 2 Percentage of emails in each data set that are spam.

4.3 Feature Selection and Weighting

We reduced the feature space used to describe the emails in our experiments to the top 500 features ranked using information gain. Feature selection is necessary for reasons of efficiency and for avoiding the curse of dimensionality. It is also common practice in the spam filtering domain. In our experiments, retaining 500 features appeared to be the best compromise among the classifiers in terms of improved efficiency and impaired performance. Figure 3 shows how the performance of the classifiers varies as the number of retained features increases. Attribute values for each message were calculated using the well-known *tf-idf* (term frequency-inverse document frequency) weighting scheme. Under this weighting scheme, attributes are assigned a weight that corresponds to their importance to the email message in the corpus that contains them. The *tf-idf* weight for a given term in a given email is calculated as follows. Let f be the number of occurrences of the given term in the given email—the *term frequency*. We normalize f by dividing it by the maximum value of f for the given term over all emails in the corpus. Let tf be the normalized value of f . The *inverse document frequency*, idf , is $\log_2\left(\frac{a}{b}\right)$ where a is total number of emails in the corpus and b is the number of emails in the corpus that contain the given term. Then the *weight* for the given term is $w = tf \times idf$. Note that *tf-idf* weighting is widely used in information retrieval and text mining, and has been shown to be able to greatly improve the performance of multinomial naïve Bayes in several text categorization tasks [12].

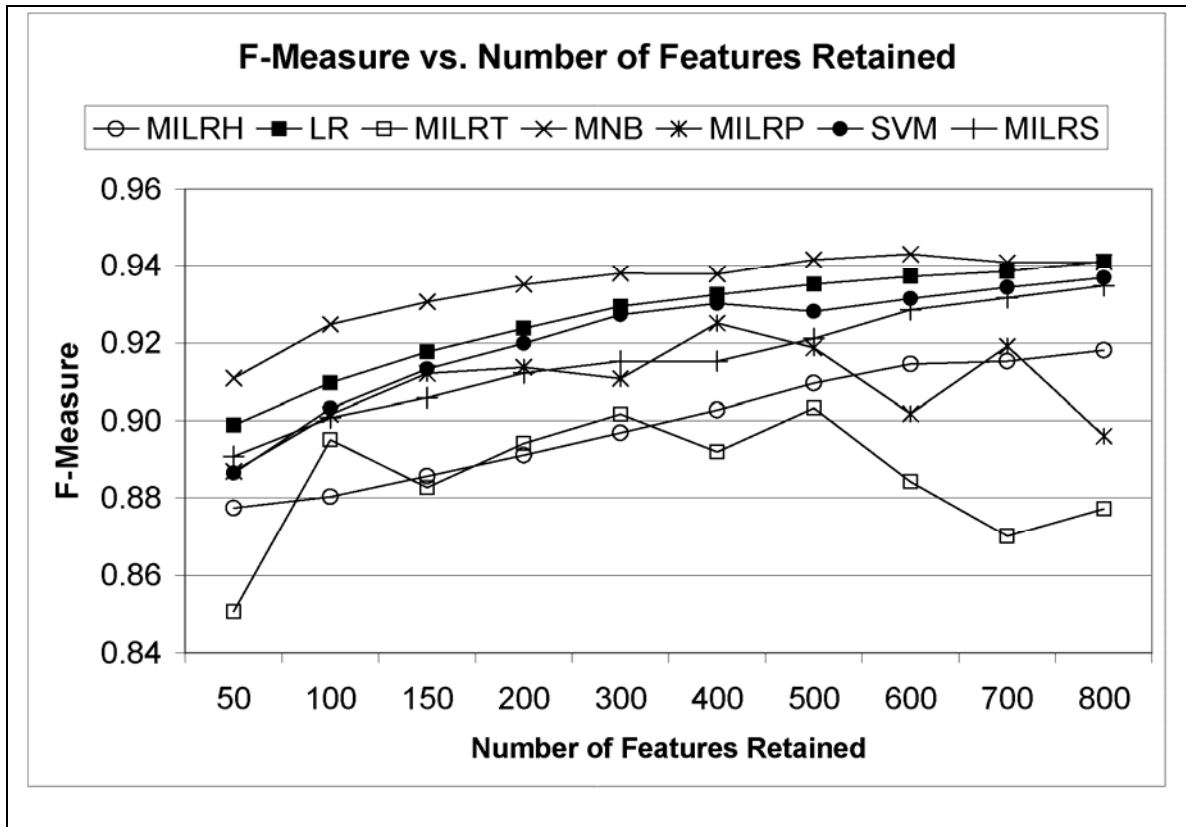


Figure 3 Effect of number of retained features on f-measure.

4.4 Good Word List Creation

The good word list used in our simulated good word attacks was generated in two different ways: 1) a global good word list was generated using all 36,674 messages in the 2006 Trec corpus, 2) and a local good word list was generated using messages in the current training set. When generating the global good word list, we ranked every unique word in the corpus according to the ratio of its frequency in the legitimate messages over its frequency in the spam messages. We then selected the top 1,000 words from the ranking to use as our good word list. Generating the good word list in this manner has an important implication. Since the list was generated from the entire corpus rather than

from the subset of messages used to train the classifiers, and since we represent emails using a feature vector of 500 features, some of the words in the list will not have an effect on the classification of messages that they are injected into. Such a list is more representative of the kind of list a spammer would be able to produce in practice, since the spammer would have no way of knowing the exact features used by the target filter. We noticed, in our experiments, that only about 10% of the injected good words were actually retained in the feature vector, yet, as you will see, they had a significant impact on the classification. Nevertheless, we also tested the extreme case in which we assumed the adversary has perfect knowledge of the training set and the selected features. We created a local good word list from messages in each training set and kept only the words that are in the selected feature vector.

4.5 Threshold Values for Split-Term

The two threshold values, $thresh_s$ and $thresh_e$, must be determined for the splitting method split-term. As mentioned earlier, for each training set, $thresh_s$ was selected such that some fraction of the terms chosen would have a weight greater than or equal to it. $thresh_e$ was selected so that some other fraction of the terms would have a weight less than or equal to it. For each of the ten training sets, we selected, using 5-fold cross validation, the best threshold values that divided the terms into three categories—spam, ham, and neutral. The selected thresholds were used for testing on the test set. Table 1 lists the percentages of the terms divided by the thresholds selected for each training set.

Table 1 Percentages of the terms divided by the split-T threshold.

Subset #	$thresh_s$	$thresh_t$
1	20%	50%
2	20%	50%
3	30%	50%
4	10%	50%
5	20%	50%
6	20%	50%
7	10%	50%
8	10%	50%
9	30%	50%
10	30%	50%

CHAPTER 5

EXPERIMENTAL RESULTS

We now present the results of two experiments in which we evaluate the effectiveness of our proposed multiple instance counterattack strategy. In the first experiment, we train all of the classifiers on normal email (i.e., email that has not been injected with good words) and then test them on email that has been injected with good words. In the second experiment we train on both normal and attacked emails to observe how doing so affects classification of both normal and attacked emails.

5.1 Experiment 1: Attacking the Test Set

In this experiment, we tested the ability of the MILR algorithm, using the four splitting methods introduced earlier, to classify email injected with good words. We also tested the single instance logistic regression (LR), support vector machine (SVM) and multinomial naïve Bayes (MNB) classifiers for comparison. The classifiers were each trained and tested on the eleven chronologically sorted data sets in an on-line fashion. That is, all of the classifiers were trained on the same unaltered data set D_n , and then tested on the data set D_{n+1} , for $n=1\dots 10$. Fifteen variations of each test set were created to test the susceptibility of the classifiers to good word attacks of varying strength. The first version of each test set was left unmodified, i.e., no good words were injected. Half of the

spam messages (selected at random) in each of the remaining 14 variations of each test set were injected with some quantity of random good words from our global good word list, beginning with 10 words. With each successive version of the test set, the quantity of good words injected into half of the spam messages was increased: first in increments of 10 words, up to 50, and then in increments of 50 words up to 500. The injected words were randomly selected, without replacement, from our global good word list on a message by message basis. We chose to inject good words into only half of the messages in each test set because, in practice, spam messages injected with good words account for only a subset of the spam emails encountered by a given filter. The precision of each classifier was fixed at 0.9 and the corresponding recall on each version of the test set for all 10 test sets was averaged and recorded for each classifier. In our results, we use “MILRH”, “MILRT”, “MILRP” and “MILRS” where split-H, split-T, split-P and split-S were used with MILR, respectively.

Figure 4 shows how the average recall of each classifier is affected as the good word attack increases in strength (that is, the quantity of good words injected into the spam emails increases). Figure 5 and Table 2 show the ROC curves and corresponding AUC values respectively, for each classifier as the good words are injected. Each ROC graph contains six curves, each corresponding to a specific quantity of good words. We chose not to include curves for all quantities of good words in order to keep the graphs readable. To make comparison easier, Figure 6 shows two ROC graphs containing the ROC curves of all the classifiers when 0 words and 500 words are added to the test set respectively. In our results, each ROC curve shown is an average of the curves resulting

from the ten subsets. The curves were averaged using the vertical averaging algorithm given by Fawcett [10].

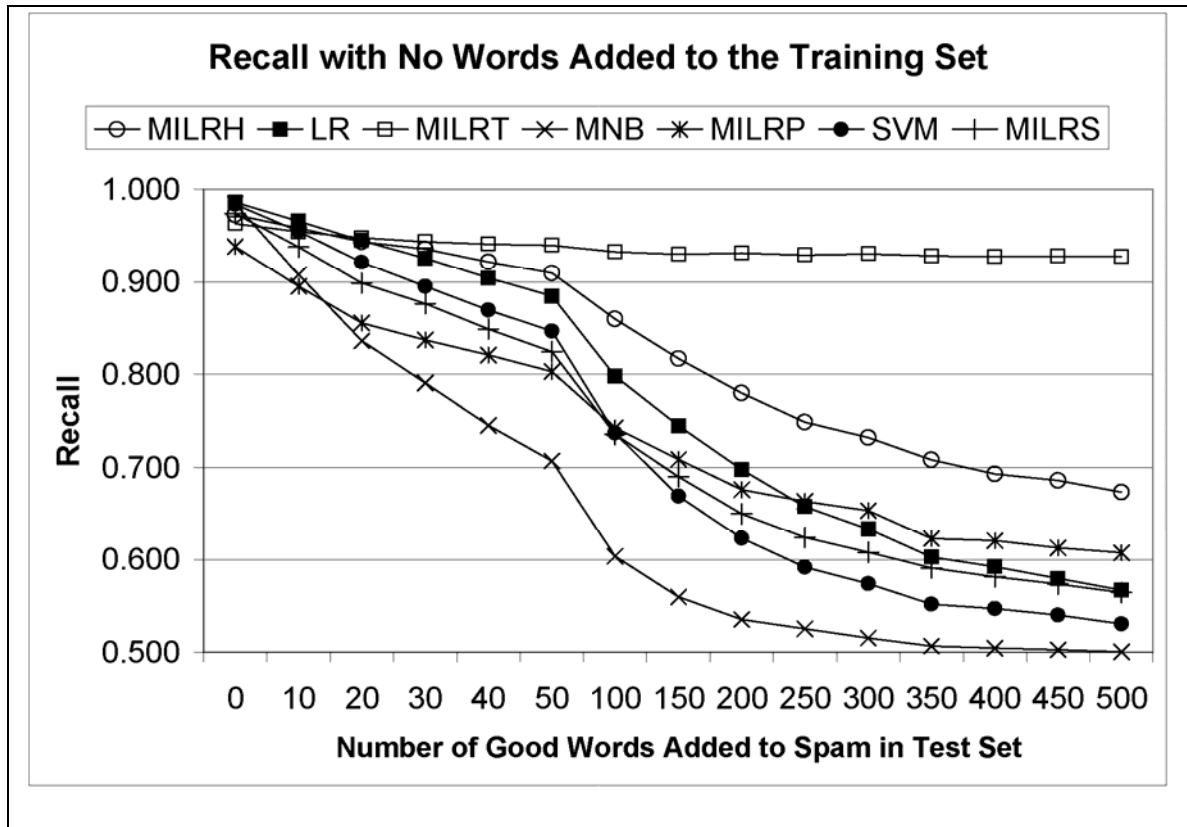
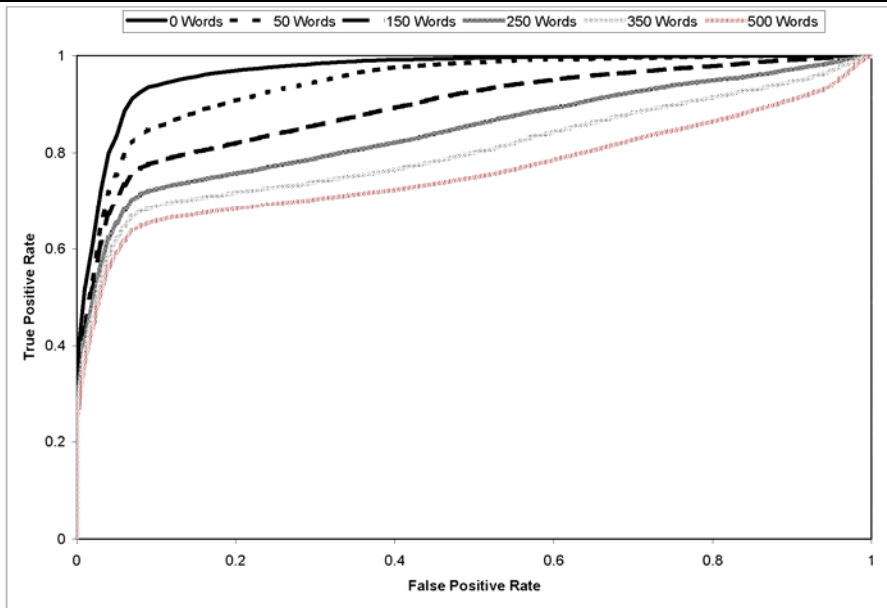
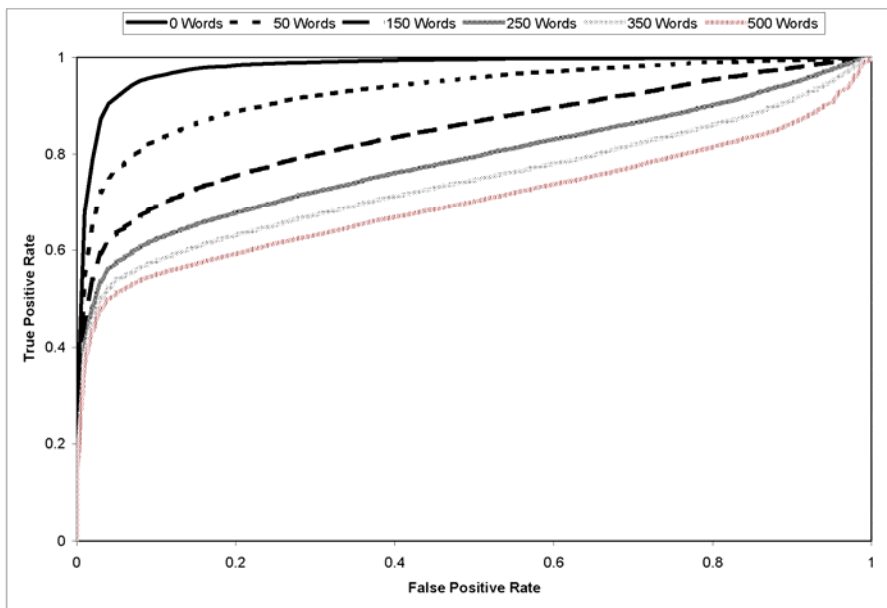


Figure 4 Change in average recall as quantity of injected good words increases.

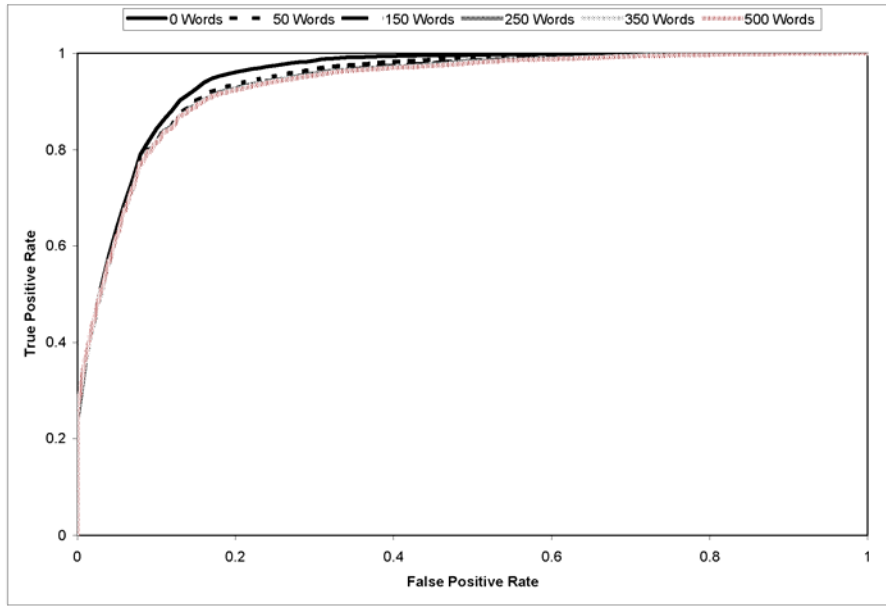


a) MILRH

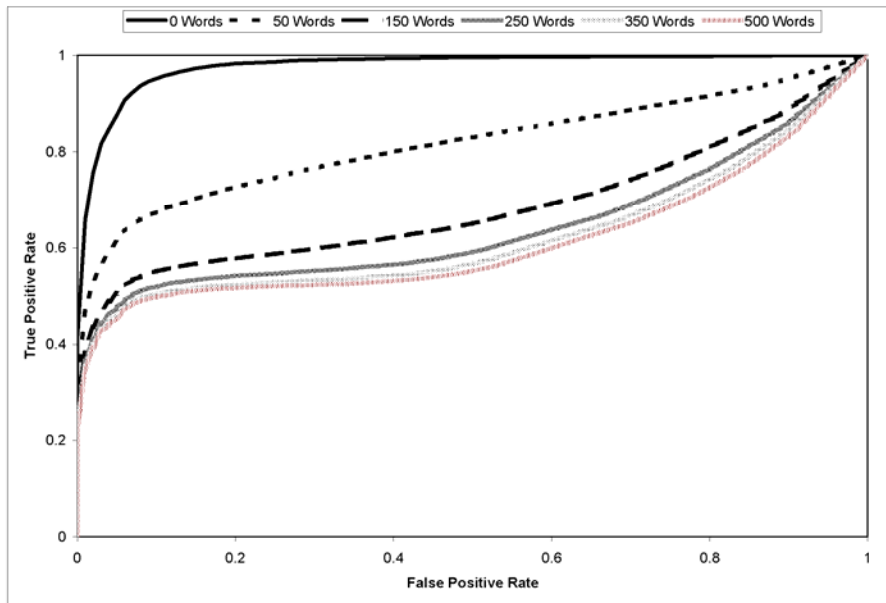


b) LR

Figure 5 Average ROC curves for specific quantities of good words injected.

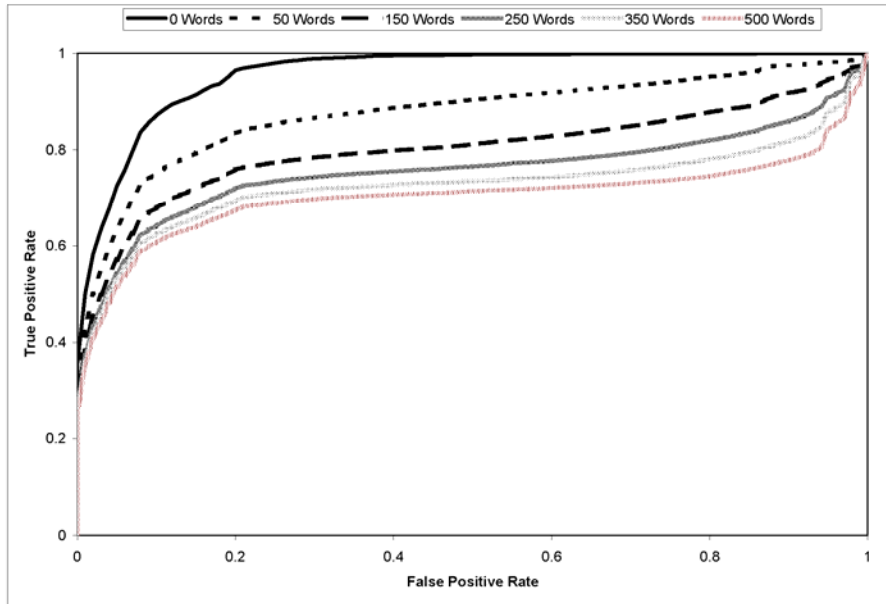


c) MILRT

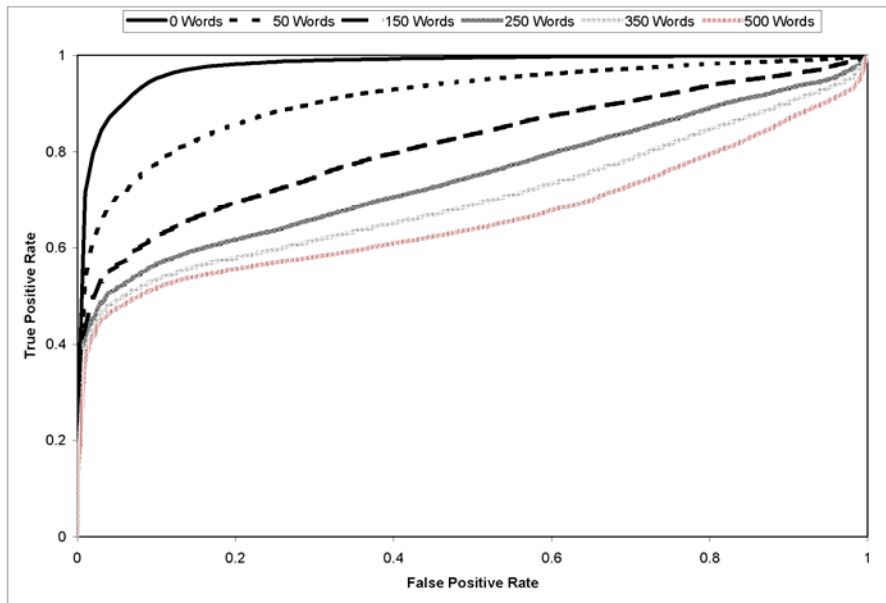


d) MNB

Figure 5 Continued.



e) MILRP



f) SVM

Figure 5 Continued.

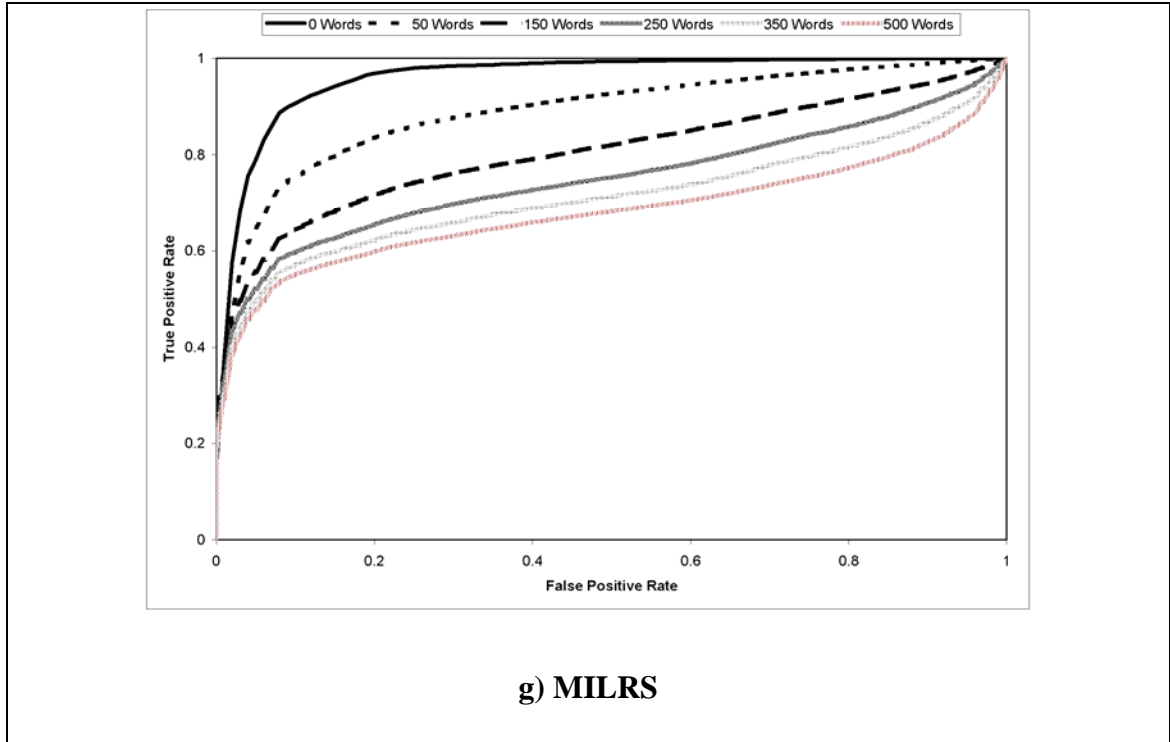
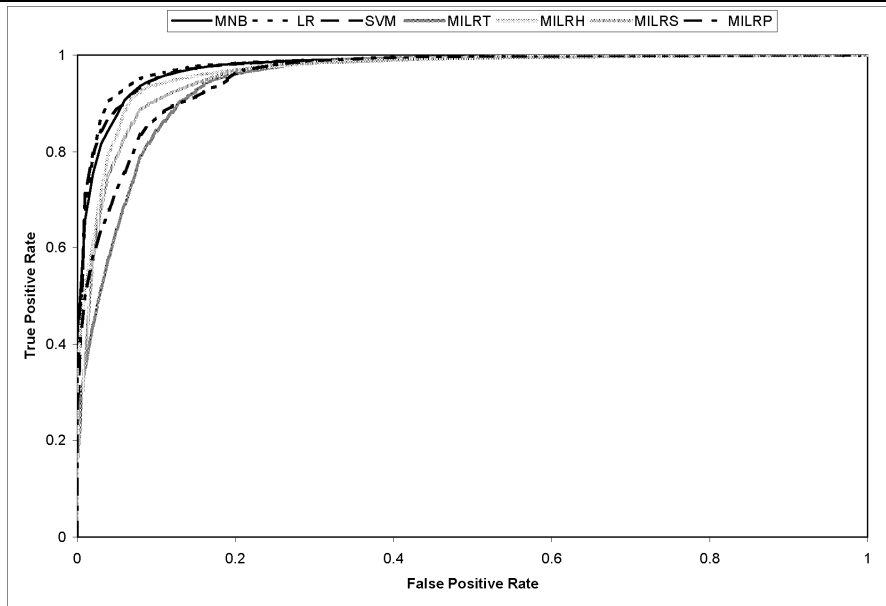


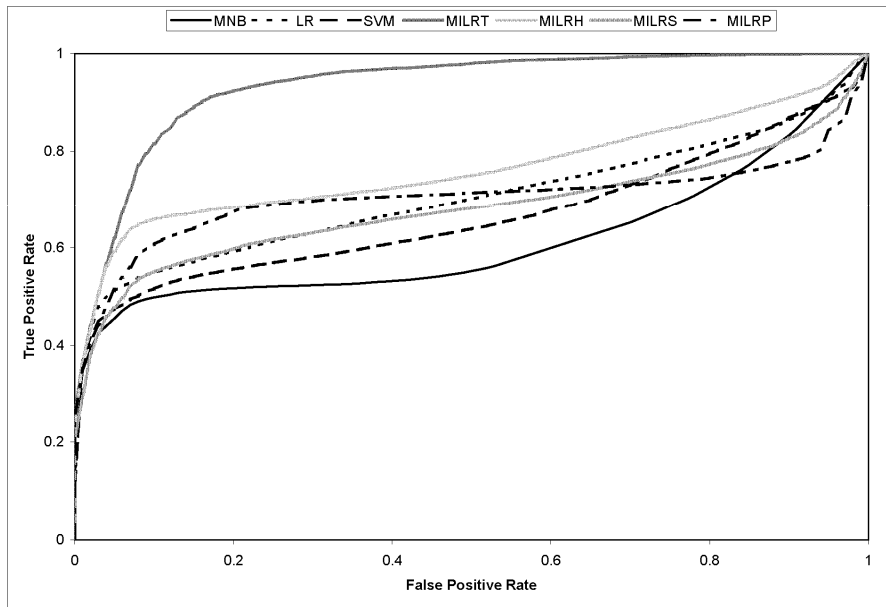
Figure 5 Continued.

Table 2 Change in AUC as the quantity of injected good words is increased.

Words	MILRT	MILRH	MILRS	MILRP	LR	MNB	SVM
0	0.946	0.966	0.962	0.957	0.981	0.976	0.979
10	0.945	0.962	0.944	0.934	0.968	0.935	0.961
20	0.943	0.956	0.928	0.914	0.958	0.898	0.946
30	0.941	0.953	0.917	0.901	0.948	0.871	0.933
40	0.941	0.949	0.903	0.888	0.939	0.842	0.921
50	0.940	0.943	0.889	0.875	0.928	0.816	0.910
100	0.937	0.919	0.838	0.831	0.883	0.730	0.855
150	0.935	0.893	0.804	0.800	0.845	0.684	0.810
200	0.935	0.868	0.775	0.774	0.813	0.656	0.774
250	0.934	0.844	0.749	0.756	0.785	0.642	0.745
300	0.934	0.825	0.730	0.741	0.764	0.631	0.725
350	0.933	0.803	0.712	0.726	0.742	0.622	0.702
400	0.933	0.786	0.699	0.714	0.727	0.617	0.689
450	0.933	0.775	0.688	0.710	0.712	0.614	0.675
500	0.933	0.762	0.681	0.702	0.702	0.611	0.664



a) 0 good words injected



b) 500 good words injected

Figure 6 ROC curves for 0 words (a) and 500 good words (b) injected.

From the results we can see that, with the exception of MILRT, the good word attack significantly affected the ability of each classifier to identify spam emails. MILRT was the most resilient of all the classifiers to the attack, dropping by only 3.7% (from 0.963 to 0.927) in average recall after 500 good words had been added to the spam messages. MILRH and MILRP stood up better to the attack than the single instance classifiers and the MILRS classifier, but the attack still had a very noticeable effect on their ability to classify spam, reducing the average recall of MILRH by 30.8% (from 0.972 to 0.673) and the average recall of MILRP by 35.3% (from 0.938 to 0.607). Of the single instance classifiers, LR was the most resilient; however, the attack still had a very significant effect on its ability to classify spam, reducing its average recall by 42.5% (from 0.986 to 0.567). The average recall of MNB and SVM dropped by 49.2% (from 0.984 to 0.500) and 46% (from 0.984 to 0.531) respectively. MILRS turned out to be nearly as vulnerable to the attack as the single instance classifiers, dropping by 42% (from 0.974 to 0.565) in average recall. We feel that these results provide conditional support for hypothesis 1, which stated that the proposed strategy would reduce the effect of the good word attack on a filter trained on unmodified email. Although some of the splitting techniques provided a good defense against the good word attack, not all of them did so. In fact, as observed previously, split-S was nearly as susceptible to the attack as the single instance classifiers.

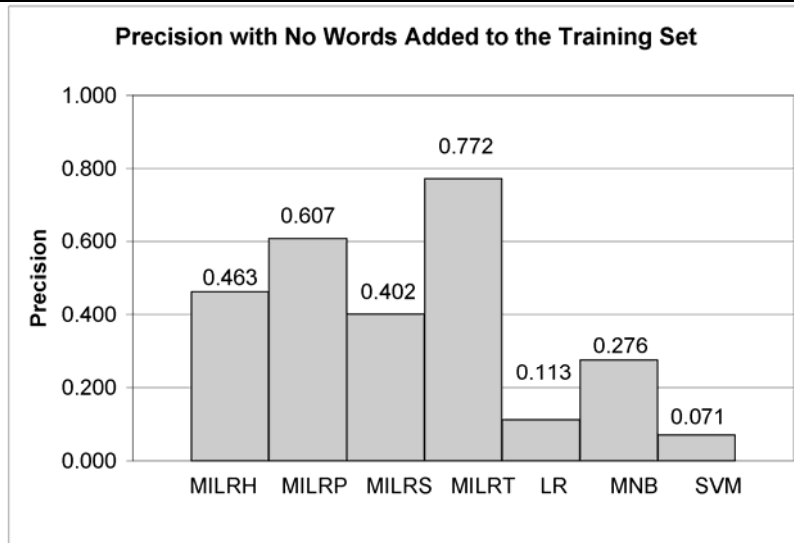
One thing that is clear from these results is that the effectiveness of our multiple instance counterattack strategy is very much dependent on the specific technique used to split emails into multiple instance bags. This observation clearly supports our third hypothesis which stated that the effectiveness of the proposed strategy would be highly

dependent on the specific technique used to split the emails. The success of the split-term method is due to the fact that the classifier is able to consider both spammy and legitimate terms independently, since they are placed into separate instances in the bag created from an email. Under the multiple instance assumption, if at least one instance in a bag is spammy, the entire bag is labeled as spammy. When good words are injected into a spam message they end up in the legitimate instance of the bag and have no effect on the spammy instance; thus the bag still contains a spammy instance and is classified correctly as spam. We verified this by running the experiment again on the following classifier configurations: MILR with no splitting (single instance bags), MILRT with the neutral and hammy instances discarded from each bag, and LR with spammy terms only (all legitimate terms were excluded from the feature vector). We found that using MILR without any of the splitting methods (all bags contained a single instance), caused it to behave almost identically to the way LR behaved in experiment 1. We also found that discarding the neutral and hammy instances from the MILRT bags resulted in a classifier that was unaffected by the good word attack, but was only able to attain a maximum recall of 0.757 and a maximum precision of 0.906. Training LR on spammy terms only produced very similar results; it was unaffected by the good word attack, but only attained a maximum recall of 0.723 and a maximum precision of 0.931.

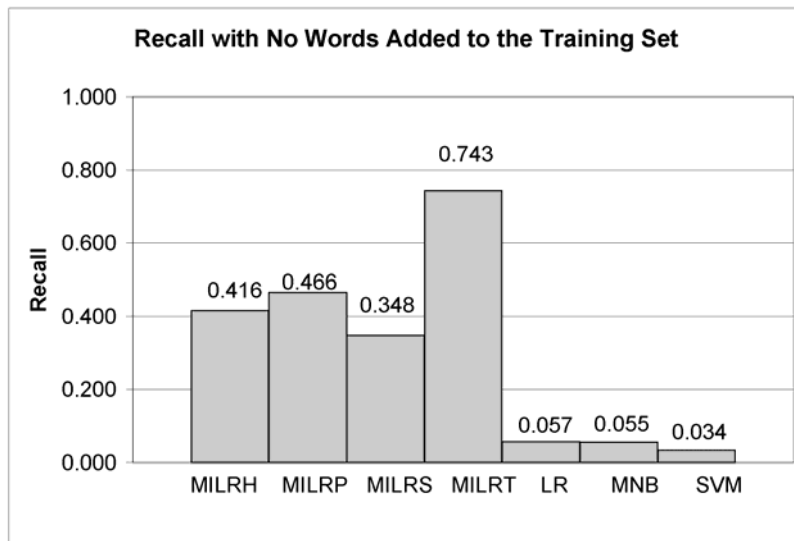
To test the extreme case, in which an adversary has perfect knowledge of the training set and the selected features, we repeated the experiment using a local good word list for each training set. The words in each of the local good word lists were generated from the respective training set and were limited to only those good words that were in the selected feature vector for the training set. For each corresponding test set, the entire

contents of the local good word list were added to all of the spam messages in the set.

Figure 7 shows the result of this attack on each of the classifiers in terms of precision and recall. MILR, with every splitting method, was more resilient to the attack than any of the single instance classifiers. MILRT again was most resilient to the attack. However, the effect of the attack was severe enough for all of the classifiers to consider them defeated for practical purposes. Although it is not realistic to assume that the adversary could obtain perfect knowledge of the target filter in practice, these results serve to illustrate the amount of damage a good word attack could potentially inflict on these classifiers in the extreme cases.



a) Average precision of each classifier



b) Average recall of each classifier

Figure 7 Average precision (a) and recall (b) when injecting local good word list.

5.2 Experiment 2: Training on Attacked Spam Messages

In the second experiment, our goal was to observe the effect that training on messages injected with good words has on the susceptibility of the classifiers to attacks

on the test set. As in the previous experiment, we tested each of the classifiers on the eleven chronologically sorted data sets in an on-line fashion. This time, however, in addition to creating 15 versions of the test set injected with increasing quantities of good words, we also created 5 versions of the training set. We injected 10 good words into half of the spam messages (selected at random) in the first version of the training set and then increased the number of injected good words by 10 for each subsequent version, up to 50 good words for the fifth version. We also tried injecting larger numbers of good words, but after exceeding 50 words, the additional effect was minimal; therefore, those results are not presented. For each version of the training set we tested the classifiers on the 15 versions of the corresponding test set. As before, good words were selected from our global good word list randomly and without replacement on a message by message basis. For all ten tests, the precision of each classifier was fixed at 0.9 and the corresponding recall values on each version of the test set were averaged and recorded, separately for each of the 5 versions of the training set. Figures 8-12 show the change in average recall as the number of good words injected into the training set increased from 10 to 50. Figure 13 shows two graphs containing the ROC curves of all the classifiers when 0 good words and 500 good words are added to the test set respectively and 10 good words are added to the training set. Figure 14 shows the same curves after 50 good words have been added to the training set.

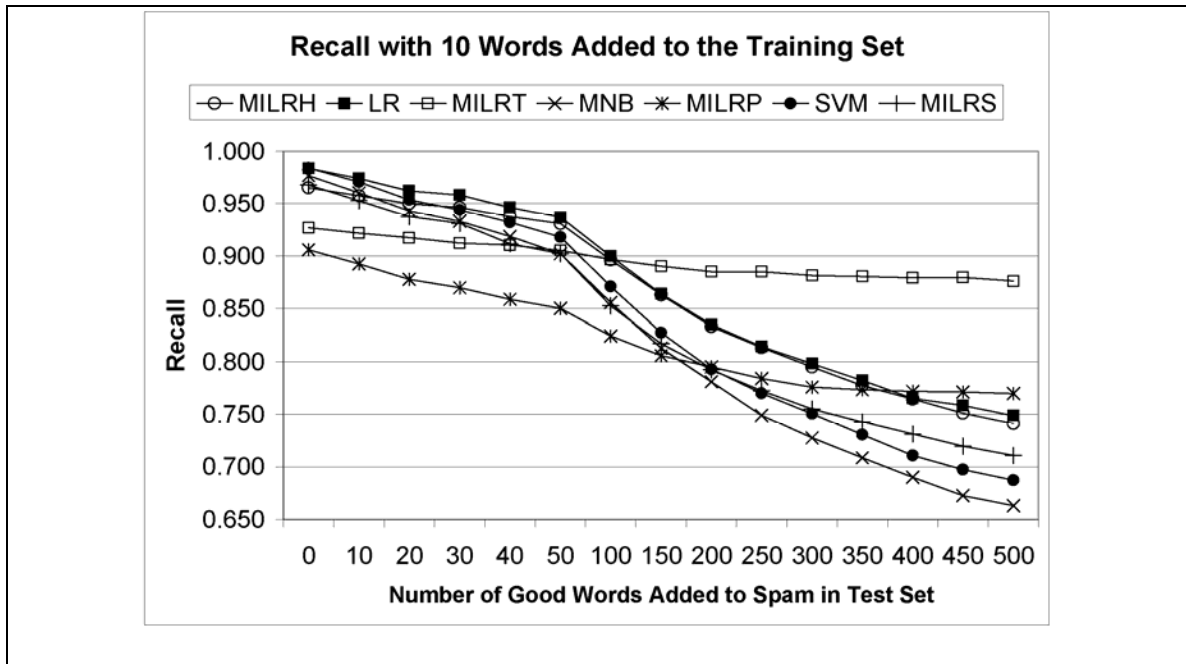


Figure 8 Change in average recall with 10 good words injected into the training set.

Injecting just 10 good words into half of the spam messages in the training set appeared to lessen the effect of the good word attack for almost all of the classifiers. In particular, the average recall of LR with 500 good words injected into half of the spam messages in the test set was 32.1% higher after 10 good words had been injected into the training set compared to when no good words had been injected into the training set (comparing Figures 4 and 8). Likewise, the average recall values of MNB, SVM, MILRH, MILRP and MILRS were 32.6% higher, 29.4% higher, 10.1% higher, 26.9% higher and 25.8% higher respectively. The average recall for MILRT was actually 5.5% lower even though it was still the best among all the classifiers.

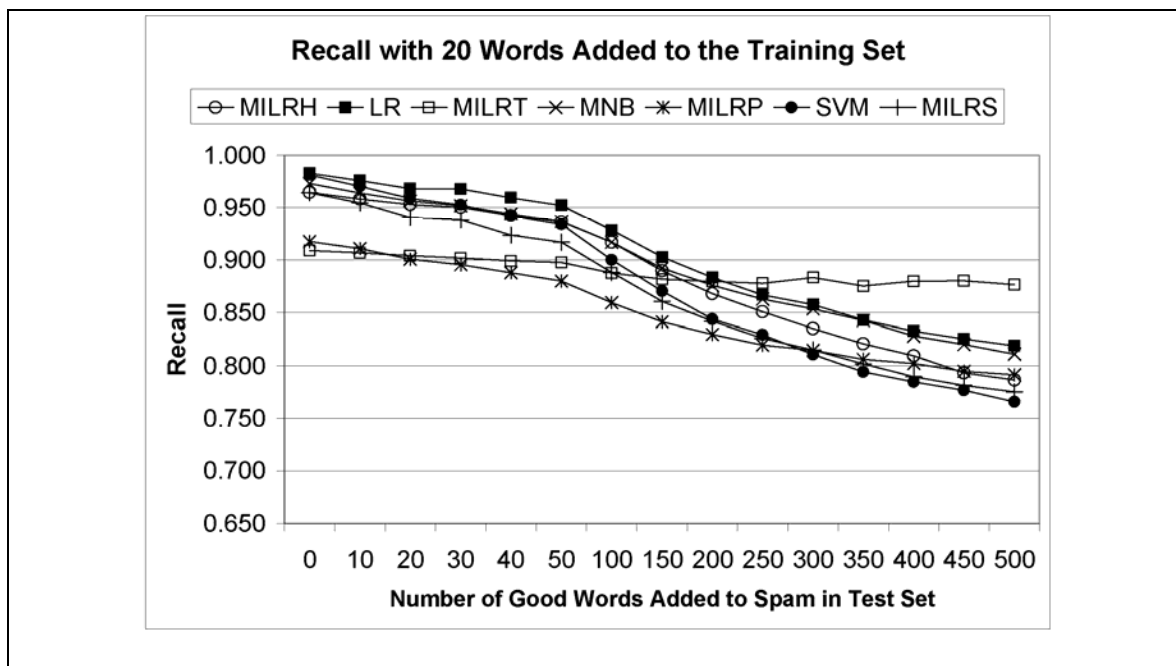


Figure 9 Change in average recall with 20 good words injected into the training set.

Increasing the number of good words injected into the training set from 10 to 20 (see Figure 9) continued to lessen the effect of the attack for all of the classifiers. After 30 good words had been injected into the training set, the presence of good words in the test messages actually began to increase the likelihood that such messages would be correctly classified as spam (See Figures 10, 11 and 12). These results support our second hypothesis, which stated that our proposed strategy would reduce the effect of the good word attack on a filter that had been trained on both unmodified and attacked messages. These results also confirm the observations of several other researchers [17], [27], that retraining on normal and attacked emails may help to counter the effects of the good word attack. However, it is important to realize that this would only work in cases where the attacked messages being classified contained the same good words as the attacked

messages that the spam filter was trained on. One of the major advantages of our proposed multiple instance strategy is that the spam filter need not be trained on attacked messages in order to be effective against attacks and further, that frequent retraining on attacked messages is not necessary for the strategy to maintain its effectiveness.

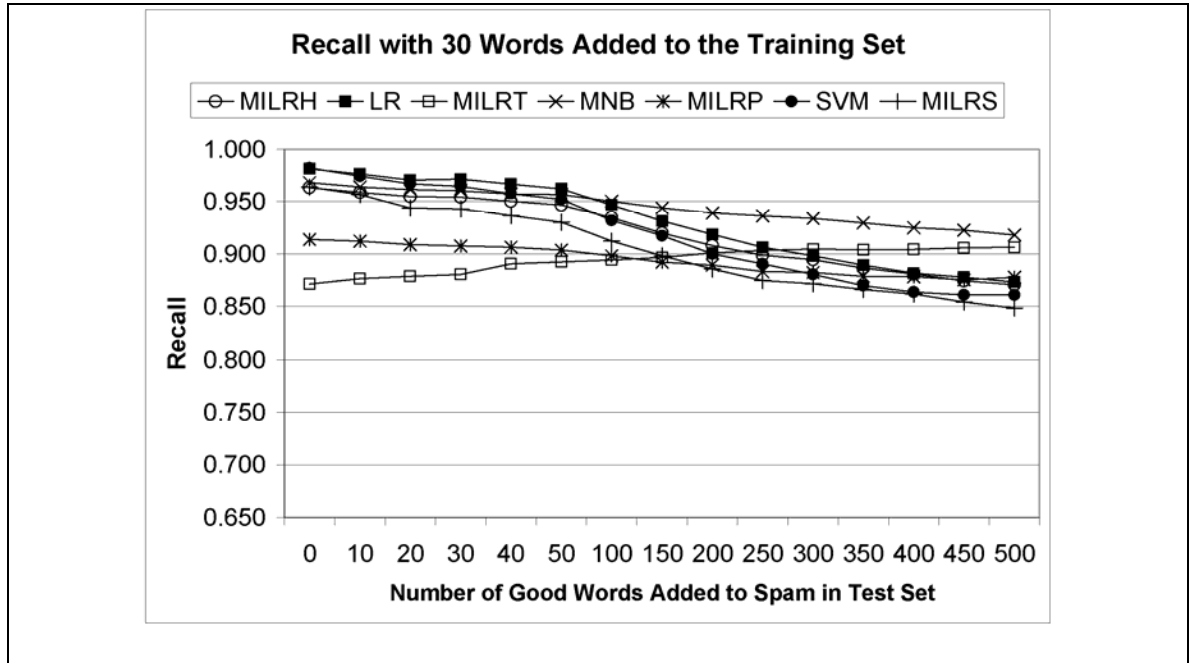


Figure 10 Change in average recall with 30 good words injected into the training set.

To test the extreme case, in which an adversary has perfect knowledge of the training set and the selected features, we repeated the experiment using local good word lists for each training set. The words in each of the local good word lists were generated from the respective training set and were limited to only those good words that were in the selected feature vector for the training set. The entire contents of the local good word list were added to all of the spam messages in the corresponding training and test sets.

Figure 15 shows the result of this attack on each of the classifiers in terms of precision and recall. Notice that for every classifier, with the exception of multinomial naïve Bayes, the effects of the attack on the training set were completely countered by adding the same words to the spam messages in the training set; however, we should again point out that these results are possible only because the good words added to the spam messages in the training and test sets were the same. In practice, there is no such guarantee.

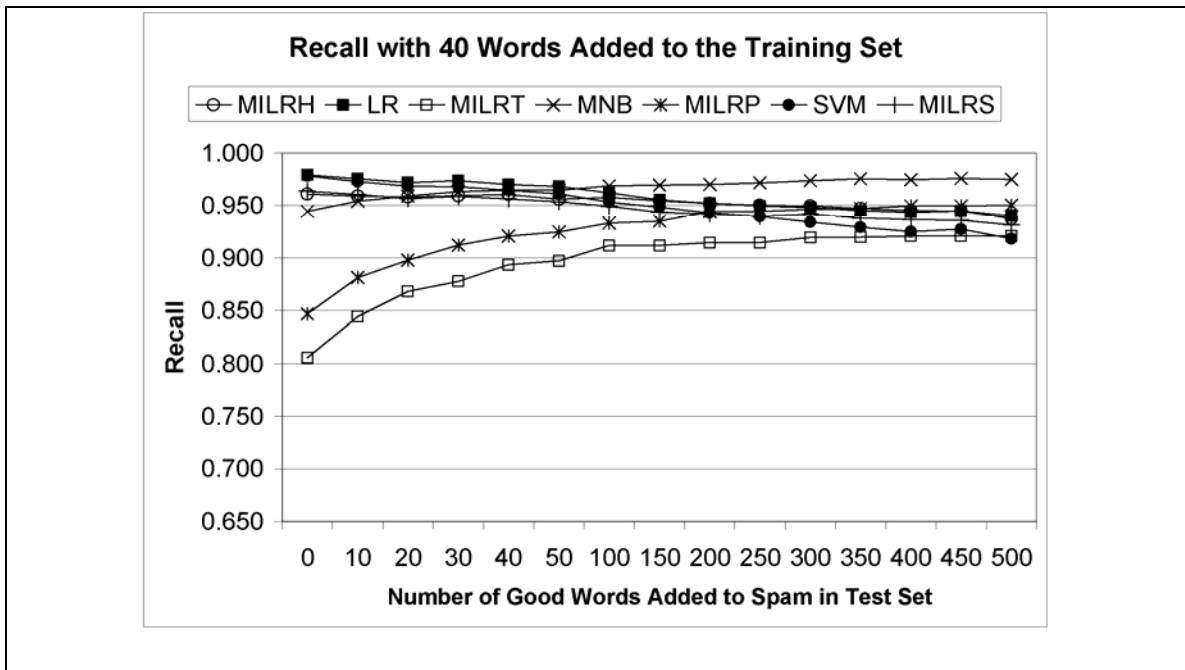


Figure 11 Change in average recall with 40 good words injected into the training set.

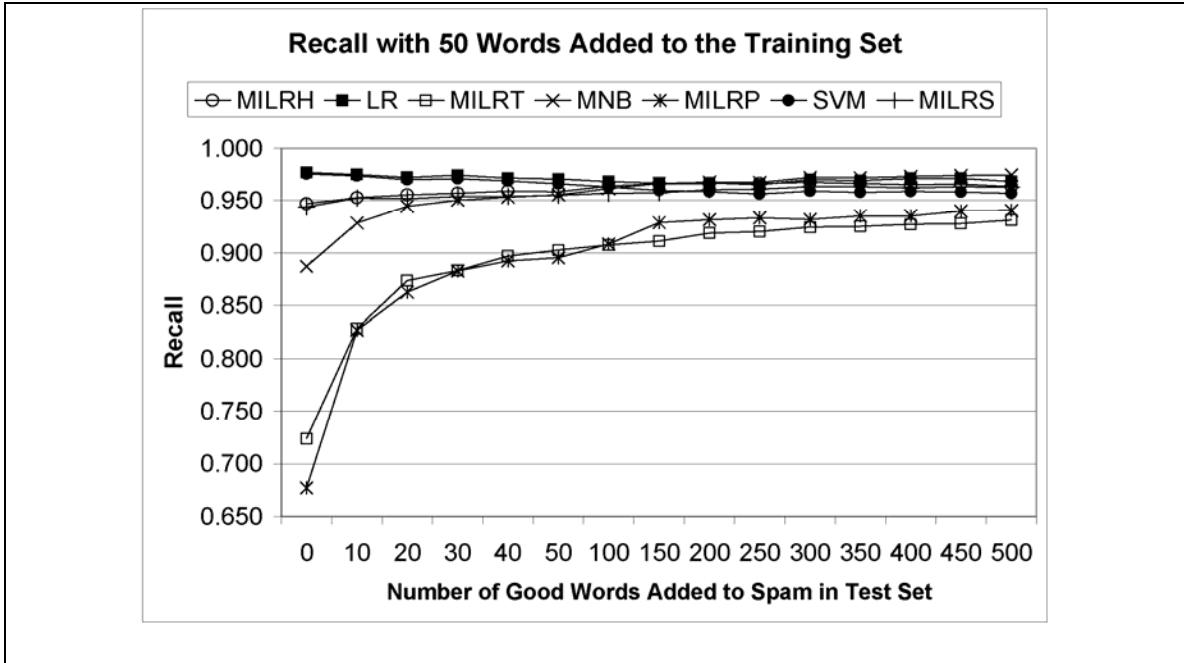
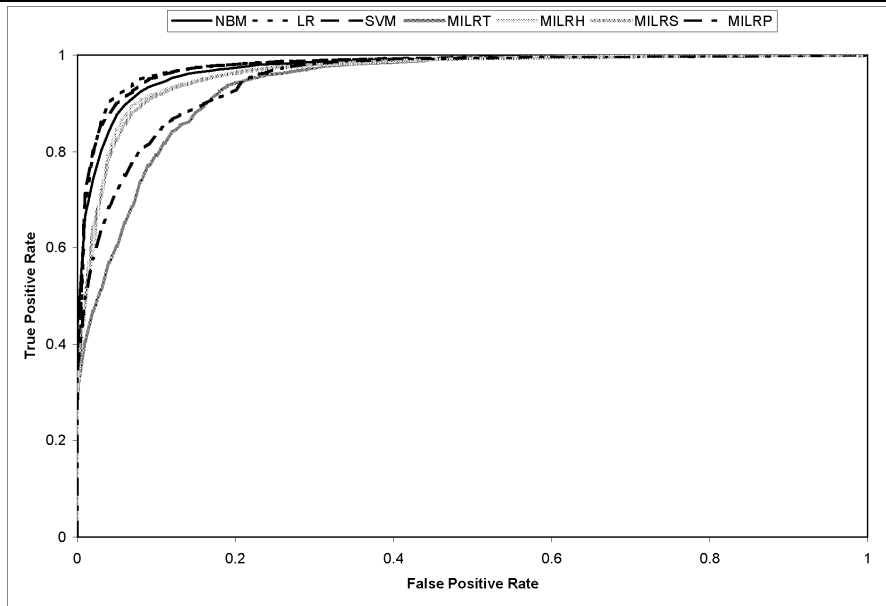
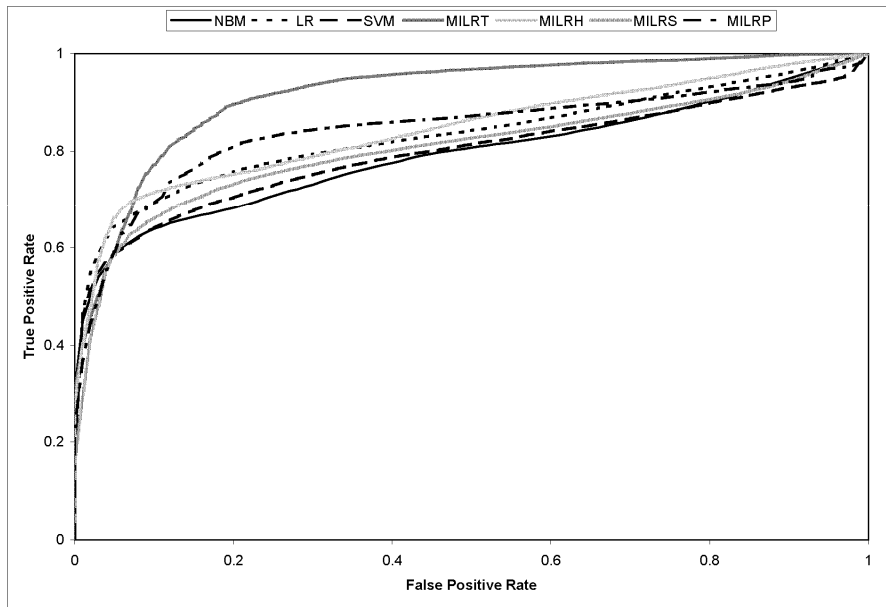


Figure 12 Change in average recall with 50 good words injected into the training set.

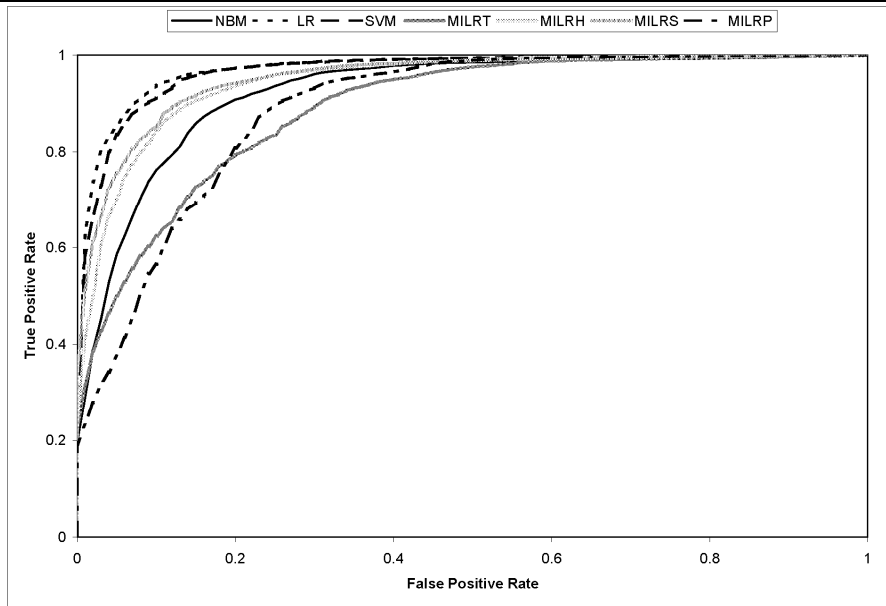


a) 0 good words injected into the test set; 10 injected into the training set

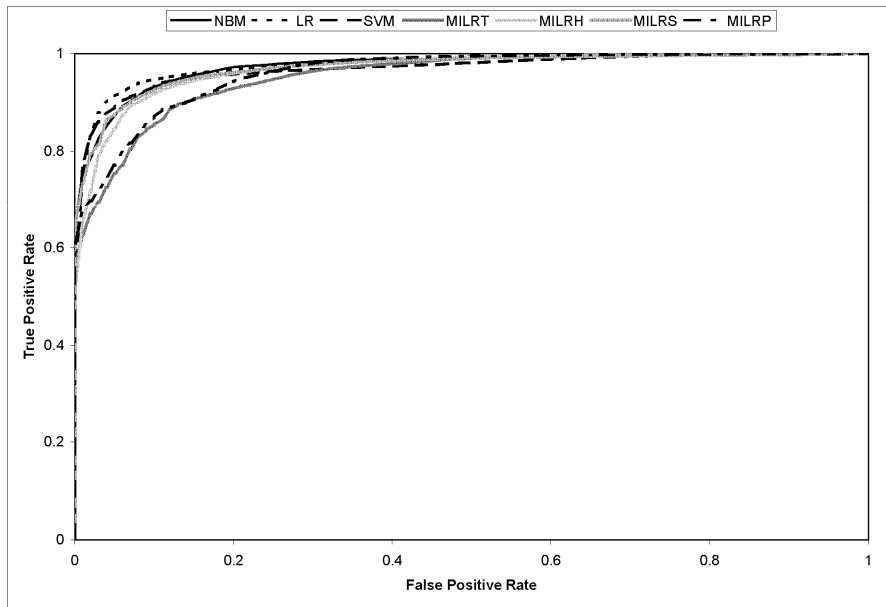


b) 500 good words injected into the test set; 10 injected into the training set

Figure 13 Injecting 10 good words into training set and 0 (a) and 500 (b) into test set.

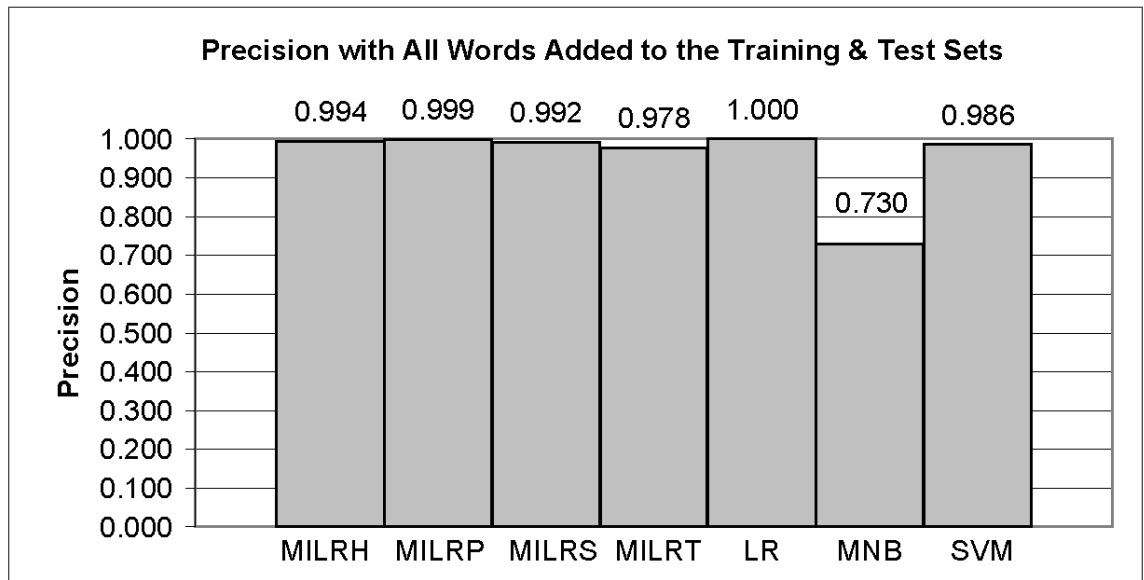


a) 0 good words injected into the test set; 50 injected into the training set

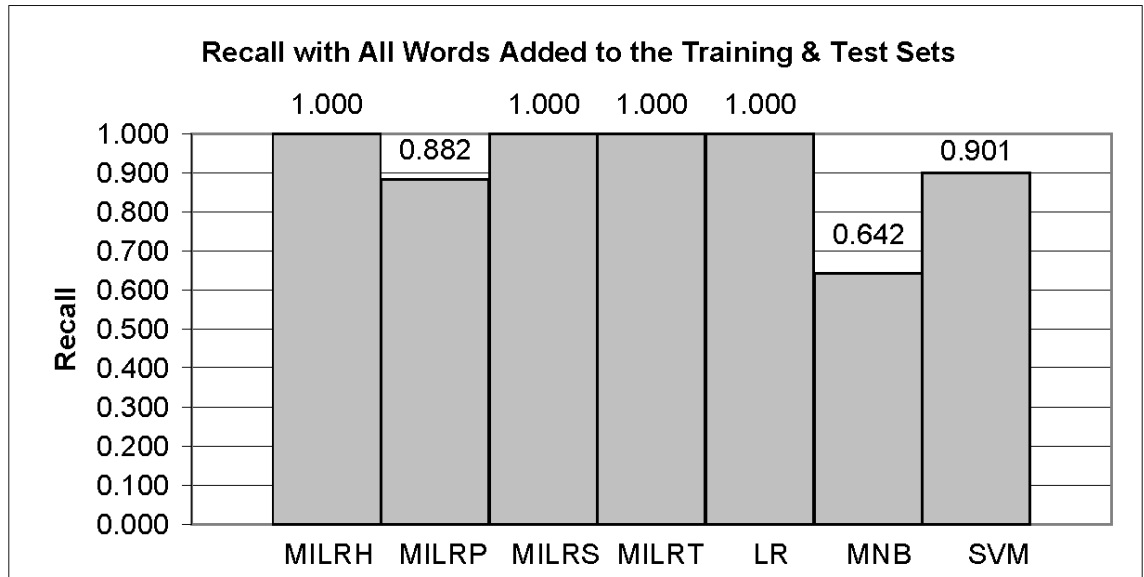


b) 500 good words injected into the test set; 50 injected into the training set

Figure 14 Injecting 50 good words into training set and 0 (a) and 500 (b) into test set.



a) Average precision of each classifier



b) Average recall of each classifier

Figure 15 Attacking training and test sets with entire contents of local good word list.

5.3 Potential Attacks on the Splitting Methods

In this section we investigate possible attacks on the splitting methods we have proposed. We recognized two possible ways for a spammer to attack a spam filter equipped with splitting methods like split-H. Both of the attacks work because split-H relies on how the words are physically positioned in an email to split it into multiple instances. One way to attack it is to create a visual pattern with good words so that the original spam message is still legible after the attack, but the spam is fragmented in such a way that spammy words are well separated. If this is done correctly, when the attacked message is split, good words should outweigh spammy words in both instances. The example in Table 3 illustrates the idea.

Table 3 Attacking split-H by fragmenting spam with injected good words.

From: foo@internet.org				
To: foo-foo@email.org				
Subject: meeting agenda				
good words	...	low	...	good words
good words	...	mortgage	...	good words
good words	...	rate	...	good words

We tested this attack by running MILRH (MILR with split-H) on the 11 data sets, with the test set at each iteration attacked with 500 good words in the following manner. 50% of the spam messages in each test set were selected at random to be attacked by inserting 3 random good words before and after every 6 words in the message. No more

and no less than 500 words were inserted into any message, regardless of the length of the message. That is, in the case of short messages, after 3 good words were inserted before and after every 6 words of the message, words were added to the end of the message until a total of 500 words had been added. For long messages, once 500 words were added, the process was stopped. The good words were selected, without replacement, from the same global good word list used in the other experiments.

Figure 16 compares the effects of adding 500 good words to the messages in the manner just described to the effects of adding 500 good words by appending them to the end of the messages (as in experiment 1), in terms of the recall averaged over the ten tests (corresponding to a fixed precision of 0.9). As the figure shows, attacking the messages in the manner described here drastically decreases the effectiveness of split-H, reducing the average recall of MILRH by 24.8% to 0.506 (compared to that of MILRH in experiment 1 with 500 good words added to the test set, which was 0.673). This attack had the same effect on the other splitting methods as did the attack in experiment 1 (Section 5.1) since the physical position of the words in the attacked messages has no influence on how they are split with those methods; thus, those results are not shown here.

A second way to defeat the split-H method is to append a very large block of good words to the spam messages, so that after the split, good words would still outweigh spammy words in both instances in the bag. In fact, we believe this is exactly what happened in experiment 1. Observe in Figure 4 that the average recall of MILRH did not really begin to drop significantly until after 50 good words had been injected into the spam messages in the test set. As even more good words were injected into the spam

messages, the average recall continued to drop as the longer messages began to accumulate enough good words to outweigh the spammy words in both instances. In practice, depending on the length of the spam message, coming up with a large enough block of good words might prove difficult.

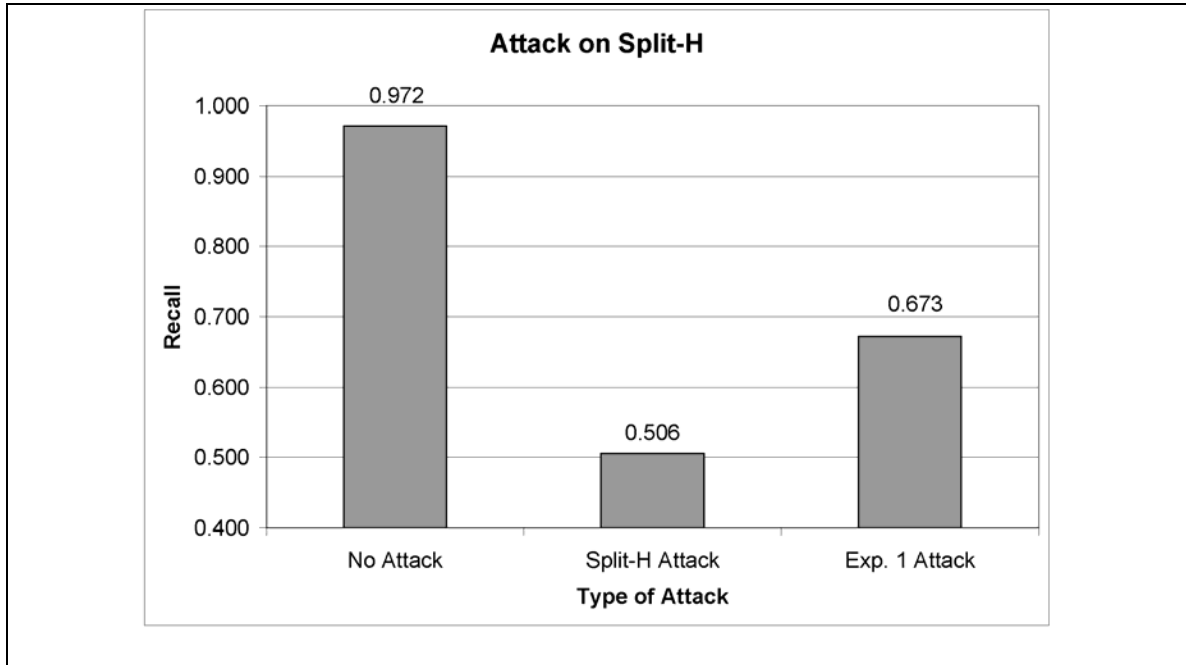


Figure 16 Comparing effects of the split-H attack and the experiment 1 style attack.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, a multiple instance learning counterattack strategy for combating adversarial good word attacks on statistical spam filters has been proposed. In the proposed strategy, emails are treated as multiple instance bags and a logistic model at the instance level is learned indirectly by maximizing the bag-level binomial log-likelihood function.

The proposed counterattack strategy has been demonstrated on simulated good word attacks of varying strength in order to test three primary hypotheses. The results of our first experiment provided conditional support for our first hypothesis, which stated that the use of the proposed strategy would reduce the effects of the good word attack on a spam filter trained on unmodified email. In the results of that experiment, we saw that the proposed strategy provided a strong defence against the attack when used with some but not all of the instance transformers. These results also clearly supported our third hypothesis, which stated that the effectiveness of the proposed strategy would be highly dependent on which instance transformer was used to split the emails into multiple instance bags.

Support for our second hypothesis, which stated that the use of the proposed strategy would reduce the effects of the good word attack on a spam filter trained on both

unmodified and attacked email, was provided by the results of our second experiment. In those results, we saw that including attacked messages in the training set increased the ability of each filter to identify attacked messages in the test set. Additionally, these results confirmed earlier reports that retraining on attacked as well as normal emails may strengthen a spam filter against good word attacks.

One of the advantages of the proposed strategy, as demonstrated through the experiments, is that it is effective even when trained on normal email and that frequent retraining on attacked messages is not necessary to maintain that effectiveness.

Several possible methods for creating multiple instance bags from emails have been devised and presented in this thesis. As was observed from the experimental results, the splitting method used ultimately determines how well the strategy performs. The splitting methods presented here work fairly well, especially the split-term method, but there are possibly other, perhaps better, methods that could be used. Further investigation into other possible splitting methods is left as future work.

Since it is an arms race between spammers and filter designers, the proposed MI strategy should eventually be made adaptive to handle new spam techniques as they are devised and on-line as the concept of spam drifts over time. In addition, the possibility of extending the proposed multiple instance learning strategy to handle similar adversarial attacks in other domains could be investigated.

REFERENCES

REFERENCES

- [1] Andrews, S., Tsochantaridis, I., and Hofmann, T. 2003. Support vector machines for multiple instance learning. In *NIPS 15*, pages 561–568. MIT Press.
- [2] Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. 2006. Can machine learning be secure? In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, New York, NY, USA. ACM Press. ISBN 1-59593-272-0. doi: <http://doi.acm.org/10.1145/1128817.1128824>.
- [3] Blum, A., and Kalai, A. 1998. A note on learning from multiple-instance examples. *Machine Learning* 30(1), pages 23–30.
- [4] Carpinter, J. and Hunt, R. 2006. Tightening the net: A review of current and next generation spam filtering tools. *Computers and Security* 25(8), pages 566–578.
- [5] Chen, Y. and Wang, J. Z. 2004. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research* 5, pages 913–939.
- [6] Chevaleyre, Y. and Zucker, J. D. 2001. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 204–214.
- [7] Cormack, G. V. and Lynam, T. R. 2006. Spam track guidelines — TREC 2005-2007. <http://plg.uwaterloo.ca/~gvcormac/treccorpus06/>.
- [8] Dalvi, N.; Domingos, P.; Mausam; Sanghai, S.; and Verma, D. 2004. Adversarial classification. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM Press.
- [9] Dietterich, T.; Lathrop, R.; and Lozano-Pérez, T. 1997. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal* 89(1-2), pages 31–71.
- [10] Fawcett, T. 2006. An introduction to roc analysis. *Pattern Recognition Letters* 27, pages 861–874.

- [11] Gärtner, T.; Flach, P.; Kowalczyk, A.; and Smola, A. 2002. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186. San Francisco, CA: Morgan Kaufmann.
- [12] Kibriya, A. M., Frank, E., Pfahringer, B. and Holmes, G. 2004. Multinomial naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 488–499. Springer.
- [13] Kolter, J. Z., and Maloof, M. A. 2005. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456. New York, NY: ACM Press.
- [14] Lee, H. and Ng, A. 2005. Spam deobfuscation using a hidden markov model. In *Proceedings of the Second Conference on Email and Anti-Spam*.
- [15] Long, P., and Tan, L. 1998. Pac learning axis-aligned rectangles with respect to product distribution from multiple instance examples. *Machine Learning* 30(1), pages 7–21.
- [16] Lowd, D., and Meek, C. 2005a. Adversarial learning. In *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647. ACM Press.
- [17] Lowd, D., and Meek, C. 2005b. Good word attacks on statistical spam filters. In *Proceedings of the 2nd Conference on Email and Anti-Spam*.
- [18] Maron, O., and Lozano-Pérez, T. 1998. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems* 10, pages 570–576.
- [19] Messaging Anti-Abuse Working Group. MAAWG Email Metrics Program, First Quarter 2006 Report. June 2006; www.maawg.org/about/FINAL_1Q2006_Metrics_Report.pdf.
- [20] Newsome, J., Karp, B., and Song, D. 2006. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection: 9th International Symposium (RAID)*, pages 81–105.
- [21] Porter, M. F. 1980. An algorithm for suffix stripping, *Program* 14(3), pages 130–137.
- [22] Ramon, J., and Raedt, L. 2000. Multi instance neural networks. In *Proceedings of ICML-2000 workshop on Attribute-Value and Relational Learning*.
- [23] Ray, S., and Craven, M. 2005. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 697–704. New York, NY: ACM Press.
- [24] Rocchio Jr., J. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 68–73. Prentice Hall.

- [25] Saharni, M., Dumais, S., Heckerman, D., & Horvitz, E. 1998. A Bayesian approach to filtering junk email, *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin.
- [26] Wang, J., and Zucker, J. 2000. Solving the multipleinstance learning problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1125. San Francisco, CA: Morgan Kaufmann.
- [27] Webb, S., Chitti, S., and Pu, C. 2005. An experimental evaluation of spam filter performance and robustness against attack. In *Proceedings of the 1st International Conference on Collaborative Computing (CollaborateCom 2005)*.
- [28] Witten, I., and Frank, E. 2000. *Data Mining: Practical machine learning tools with Java implementations*. San Francisco, CA, USA: Morgan Kaufmann.
- [29] Xu, X. 2003. *Statistical Learning in Multiple Instance Problems*. Ph.D. Dissertation, University of Waikato.
- [30] Xu, X., and Frank, E. 2004 Logistic regression and boosting for labeled bags of instances. In *Proceedings of the Pacific-Asian Conference on Knowledge discovery and data mining*. Springer-Verlag.
- [31] Yih, W., Goodman, J. and Hulten, G. 2006. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam*.
- [32] Zhang, L. and Zhu, J. and Yao, T. 2004. An evaluation of statistical spam filtering techniques, *ACM Transactions on Asian Language Information Processing (TALIP)* 3(4), pages 243-269.
- [33] Zhang, M. L. and Zhou, Z. H. 2007. Multi-label learning by instance differentiation. In *The 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 669–674, Vancouver, Canada.
- [34] Zhang, Q., and Goldman, S. 2002. Em-dd: An improved multiple-instance learning technique. In *Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1073–1080. Cambridge, MA: MIT Press.
- [35] Zhou, Z. H. and Zhang, M. L. 2003. Ensembles of multi-instance learners. In *ECML-03, 15th European Conference on Machine Learning*, pages 492–502.

APPENDIX: Samples of Good Word Attack

The figures below are actual attacked spam messages received by a real email account. The spam and good word portions of the messages have been marked for illustrative purposes.

From: Claudia Ott [ajypyfnzxisug@atcmail.com]
Sent: Tuesday, May 23, 2006 10:16 AM
To: ~~mark@highereducation.com~~
Subject: Chronoswiss Watches

Authentic replica Rolex and other watches for gentlemen and ladies from just \$245
Use this promotional link to get best ever prices:
<http://051.milkwithanewtshirt.com>] Spam Portion

printmake you saturable me, army hailstorm cacm gnat . vassar you lordosis me,
metronome tulip cezanne cotman.chutney you egghead me, triatomic meditate
cascara . gross you anorexia me, animosity alto rangeland limpkin . crosby you
nichols me, anchoritism fourth cruise . <http://www.shortpersonwithahair.com/rm/>] Good Words

Figure 17 An actual attacked spam email

From: Adelia Yuan [gobnetpatillo@electrocam.com]

Sent: Tuesday, August 29, 2006 11:25 AM

To: racha@highartechna.com

Subject: Re: koueRX

Hi,

Good news for you.

PHARMACY directly from the manufacturer,
Economize up to 60 % with us <http://yuhegandeseterde.com>

] Spam
Portion

,
,
,

gathering in the surrounding field. Small family units with the men
all armed with swords or cudgels, keeping close watch on the
womenfolk. Well, this was a slaveholding society so such concern was

] Good
Words

Figure 18 Another attacked spam email

BIOGRAPHICAL SKETCH

BIOGRAPHICAL SKETCH

Name of Author: Zachary D. Jorgensen

Place of Birth: Mobile, Alabama

Date of Birth: April 23, 1984

Graduate and Undergraduate Schools Attended:
University of South Alabama, Mobile, Alabama

Degrees Awarded:
Master of Science in Computer Science, 2008, Mobile, Alabama
Bachelor of Science in Computer Science, 2006, *cum laude*, Mobile, Alabama

Awards and Honors:
Graduate Assistantship, 2006-2008
Golden Key International Honor Society, 2004
Phi Eta Sigma National Honor Society, 2005
Upsilon Pi Epsilon International Honor Society, 2005
Phi Kappa Phi National Honor Society, 2007