Journal of Machine Learning Research 8 (2008) 1115-1146 Submitted 9/07; Revised 2/08; Published 6/08

A Multiple Instance Learning Strategy for Combating Good Word Attacks on Spam Filters

Zach Jorgensen ZDJORGEN@JAGUAR1.USOUTHAL.EDU
Yan Zhou ZHOU@CIS.USOUTHAL.EDU
Meador Inge WMI601@JAGUAR1.USOUTHAL.EDU
School of Computer and Information Sciences
University of South Alabama
Mobile, AL 36688, USA

Editor: Lyle Ungar

Abstract

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. We present a counterattack strategy that attempts to differentiate spam from legitimate email in the input space by transforming each email into a bag of multiple segments, and subsequently applying multiple instance logistic regression on the bags. We treat each segment in the bag as an instance. 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. We show that a classifier using our multiple instance counterattack strategy is more robust to good word attacks than its single instance counterpart and other single instance learners commonly used in the spam filtering domain.

Keywords: spam filtering, multiple instance learning, good word attack, adversarial learning

1. Introduction

It has been nearly thirty years since the first email spam appeared on the Arpanet. Today, to most end users, spam does not seem to be a serious threat due to the apparent effectiveness of current spam filters. Behind the scenes, however, is a seemingly never-ending battle between spammers and spam fighters. With millions of email users, profit-driven spammers have great incentives to spam. With as little as 0.001% response rate, a spammer could potentially profit $25,000 on a $50 product (Carpinter and Hunt, 2006). Over the years, spammers have grown in sophistication with cutting-edge technologies and have become more evasive. The best evidence of their growing effectiveness is a recent estimate of over US $10 billion worldwide spam-related cost (Jennings, 2005). In this paper, we target one of the adversarial techniques spammers often use to circumvent existing spam filters.

Adversarial attacks on spam filters have become an increasing challenge to the anti-spam community. The good word attack (Lowd and Meek, 2005b) is one of the techniques most frequently employed by spammers. This technique involves appending sets of so-called "good words" to spam messages. Good words are words that are common to legitimate emails (also called ham) but rare in spam. Spam messages injected with such words are more likely to appear legitimate and bypass spam filters. So far, relatively little research has been done to investigate how spam filters might be

trained to account for such attacks. This paper presents a possible defense strategy using multiple instance learning that has shown promising results in our experiments.

Multiple instance (MI) learning (Dietterich et al., 1997) 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 not necessarily 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. Therefore, the goal of multiple instance learning is to learn a classification function that accurately maps a given bag to a class. Formally, let $B = \{B_1, \ldots, B_i, \ldots, B_m\}$ be a set of bags where $B_i = \{X_{1i}, X_{2i}, \ldots, X_{ji}\}$ is the i^{th} bag and $X_{1i}, X_{2i}, \ldots, X_{ji}$ are the j instances contained in bag B_i. If B is a training set, then every $B_i \in B$ also has a class label $c_i \in C = \{positive, negative\}$ associated with it. The training process, using B as input, yields a binary classification function $f(B_i) : B \rightarrow C$ that maps a bag to a class label.

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. Our experimental results 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.

The remainder of this paper is organized as follows. First, we discuss recent research that has motivated our work. Next, we formalize the spam filtering problem as a multiple instance learning problem and explain our proposed counterattack strategy in more detail. Following that, we present our experimental results to demonstrate the effectiveness of our filtering strategy. Finally, we conclude our work and discuss future directions.

2. Related Work

Our work is primarily motivated by recent research on adversarial learning (Dalvi et al., 2004; Lowd and Meek, 2005a; Kolter and Maloof, 2005). Dalvi et al. (2004) consider classification to be a game between classifiers and adversaries in problem domains where adversarial attacks are expected. 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 (2005a) formalized the task of adversarial learning as the process of reverse engineering the classifier. In their adversarial classifier reverse engineer (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. (2006) emphasize the point that the 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 classifiers for adversarial environments should take into account the fact that training data is controlled by an adversarial source in order to ensure the production of accurate classifiers.

Barreno et al. (2006) 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 learning in the presence of the good word attack. Lowd and Meek (2005b) present and evaluate several variations of this type of attack on spam filters. They demonstrate two different ways to carry out the attack: passively and actively. Active good word attacks use feedback obtained by sending test messages to a spam filter in order to determine which words are "good". The active attacks were found to be more effective than the passive attacks; however, active attacks are generally more difficult to perform than passive attacks because they require user-level access to the spam filter, which is not always possible. Passive good word attacks, on the other hand, do not involve any feedback from the spam filter, but rather, guesses are made as to which words are considered good. Three common ways for passively choosing good words are identified. First, *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 classified as spam. Next, *frequent word attacks* involve the selection of words that occur most often in legitimate messages, such as news articles. This method was more effective than the previous one, but it still required as many as 1,000 good 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 not in spam messages. The authors' tests showed that this technique was quite effective, resulting in the average spam message being passed off as legitimate by adding as few as 150 good words to it. Preliminary results were also presented that suggested that frequent retraining on attacked messages may help reduce the effect of good word attacks on spam filters.

Webb et al. (2005) also examined 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, that is, email that has not been modified with good words, each of the filters is able to attain an 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. (1997) 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 (Auer, 1997; Long and Tan, 1998; Blum and Kalai, 1998).

Several probabilistic models: Diverse Density (DD) (Maron and Lozano-Pérez, 1998) and its variation EM-DD (Zhang and Goldman, 2002), and multiple instance logistic regression (MILR) (Ray and Craven, 2005), 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 (2005) 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 (2004).

Many single-instance learning algorithms have been adapted to solve the multiple instance learning problem. For example, Wang and Zucker (2000) propose the lazy MI learning algorithms, namely Bayesian-*k*NN and citation-*k*NN, 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 (2001) 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 (Ramon and Raedt, 2000), DD-SVM (Chen and Wang, 2004), MI-SVM and mi-SVM (Andrews et al., 2003), multi-instance kernels (Gärtner et al., 2002), MI-Ensemble (Zhou and Zhang, 2003), and MI-Boosting (Xu and Frank, 2004).

In this paper, we demonstrate that a counterattack strategy against good word attacks, developed in the framework of multiple instance learning, can be very effective, provided that a single instance

can be properly transformed into a bag of instances. We also explore several possible ways to transform emails into bags of instances. Our experiments also verify earlier observations, discussed in other works (Lowd and Meek, 2005b; Webb et al., 2005), that retraining on emails modified during adversarial attacks may improve the performance of the filters against the attack.

3. Problem Definition

Consider a standard supervised learning problem with a set of training data $D = \{< X_1, Y_1 >, \ldots, < X_m, Y_m >\}$, where X_i is an instance represented as a single feature vector, $Y_i = C(X_i)$ is the target value of X_i, where C is the target function. Normally, the task is to learn C given D. The learning task becomes more difficult when there are adversaries who could 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 then apply multiple instance learning to learn a hypothesis defined over a set of bags.

4. Multiple Instance Bag Creation

We now formulate the spam filtering problem as a multiple instance binary classification problem in the context of adversarial attacks. Note that the adversary is only interested in altering positive instances, that is, spam, by injecting sets of good words that are commonly encountered in negative instances, that is, legitimate emails, or ham. We propose four different approaches to 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. Later, in Section 8, we investigate and discuss possible weaknesses of some of these splitting methods.

4.1 Split-H

The first and simplest splitting method that we considered, which we call *split-half* (split-H), involves splitting an email down the middle into approximately equal halves. Formally, let $B = \{B_1, \ldots, B_i, \ldots, B_m\}$ be a set of bags (emails), where $B_i = \{X_{i1}, X_{i2}\}$ is the i^{th} bag, and X_{i1}, 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 in Section 8, 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 problem.

4.2 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, that is, $B_i = \{X_{is}, X_{in}, X_{ih}\}$, 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:

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

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 6.2) would have a weight greater than or equal to it. $thresh_\ell$ was selected so that some other fraction of the terms would have a weight less than or equal to it.

4.3 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 (Rocchio Jr., 1971) 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:

$$M_s = \frac{M \cdot P_s}{|P_s|^2} P_s,$$
$$M_\ell = \frac{M \cdot P_\ell}{|P_\ell|^2} P_\ell.$$

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.

4.4 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 (Zhang and Zhou, 2007).

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. In this paper, 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 (Yih et al., 2006), appeared to be the best among the single instance learners considered in our experiments. The next section outlines multiple instance logistic regression.

5. Multiple Instance Logistic Regression

Given a set of training bags

$$B = \{< B_1, Y_1 >, \ldots, < B_i, Y_i >, \ldots, < B_m, Y_m >\},$$

let $Pr(Y_i = 1 \mid B_i)$ be the probability that the i^{th} bag is positive, and $Pr(Y_i = 0 \mid B_i)$ be the probability that it is negative. Here Y_i is a dichotomous outcome of the i^{th} bag (for example, spam or legitimate). 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, that is, $Pr(Y_i = 1 \mid X_i) = \exp(p \cdot X_i + b)/(1 + \exp(p \cdot X_i + b))$, 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 $Pr(Y_{ij} = 1 \mid X_{ij})$ with a sigmoidal response function as follows:

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

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 $Pr(Y_i = 0 \mid B_i)$ with instance-level class probabilities can be computed as follows:

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

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

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

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

$$Pr(Y_i = 1 \mid B_i) = 1 - Pr(Y_i = 0 \mid B_i)$$

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 counts (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.

6. Experimental Setup

We evaluated our multiple instance learning counterattack strategy on emails from the 2006 TREC Public Spam Corpus (Cormack and Lynam, 2006). 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 above, to its single instance learning

counterpart—logistic regression (LR)—and to the support vector machine (SVM) and the multinomial naïve Bayes (MNB) classifiers. Additionally, we tested a relatively new compression-based spam filter (Bratko and Filipič, 2005) against the good word attack. The information in the next two subsections regarding corpus preprocessing and feature selection and weighting does not apply to the compression-based filter; it will be discussed separately in Section 7.3.

6.1 Experimental Data

Our experimental data consists of 36,674 spam and legitimate email messages from the 2006 TREC spam corpus. We preprocessed the entire corpus by stripping HTML and non-textual parts and applying stemming and stop-list to all terms. The **to**, **from**, **cc**, **subject**, and **received** headers were retained, while the rest of the headers were stripped. Messages that had 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 paper. Given that there are a large number of possible ways to disguise a word, most content-based spam filters will not be able to deobfuscate the text of a message efficiently (Carpinter and Hunt, 2006). Recently, an efficient complementary filter (Lee and Ng, 2005) has been demonstrated to be able to effectively deobfuscate text with high accuracy. In practice, this type of technique could be used during preprocessing.

For our experiments we sorted the emails in the corpus chronologically by receiving date and evenly divided them into 11 subsets $\{D_1, \ldots, D_{11}\}$. In other words, the messages in subset n come chronologically before the messages in subset $n+1$. Experiments were run in an on-line fashion, that is, training on subset n and testing 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 1). We used the Multiple Instance Learning Tool Kit (MILK) (Xu, 2003) implementation of MILR and the Weka 3.4.7 (Witten and Frank, 2000) implementations of LR, SVM and multinomial naïve Bayes, in our experiments. For the compression-based filter, we used the spam filter described in Bratko and Filipič (2005), which uses the prediction by partial matching algorithm with escape method D (PPMD) and is available as part of the PSMSLib C++ library (Bratko, 2008).

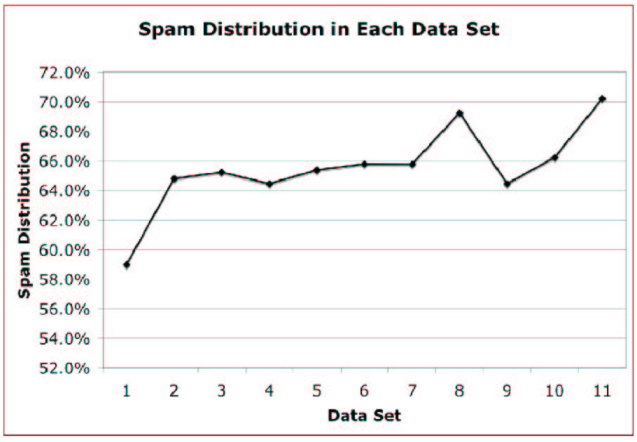


Figure 1: Percentage of emails in each data set that are spam.

6.2 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 2 shows how the performance of the classifiers varies as the number of retained features increases.

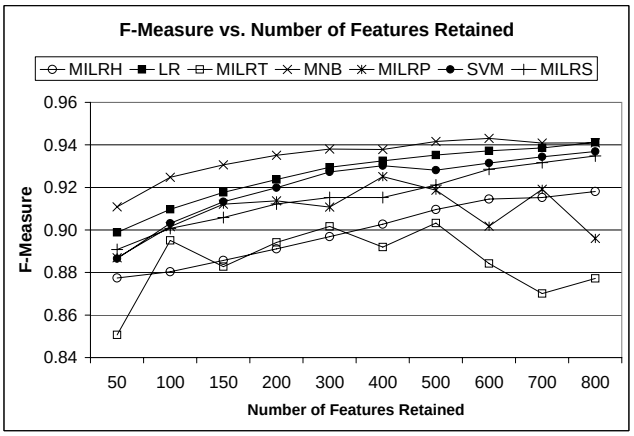


Figure 2: Effect of number of retained features on f-measure.

Attribute values for each email were calculated using the common *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 $\log_2(\frac{a}{b})$ where a is the 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 (Kibriya et al., 2004).

6.3 Good Word List Creation

The good word list used in our simulated good word attacks was generated in two different ways: 1) the *global* good word list was generated using all 36,674 messages in the 2006 Trec corpus, 2) and the *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 that in our experiments, only about 10% of the injected good words were actually retained in the feature vector, yet 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.

6.4 Threshold Values for Split-Term

The two threshold values, $thresh_s$ and $thresh_\ell$, 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. $thresh_\ell$ 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, by using 5-fold cross validation, the best threshold values that divide 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.

Subset	% of terms with weights $\geq thresh_s$	% of terms with weights $\leq thresh_\ell$
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%

Table 1: Percentages of the terms divided by the MILRT threshold values selected for each training set.

7. 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 (that is, 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. The compression-based filter and its susceptibility to the good word attack are examined separately in Section 7.3.

7.1 Experiment 1: Attacking the Test Set

In this experiment, we tested the ability of the MILR algorithm, using the four splitting methods introduced above, 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 $n = 1...10$. Fifteen variations of each test set were created to test the susceptibility of the classifiers to good word attacks of varying strengths. The first version of each test set was left unmodified, that is, 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 3 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). Figures 4-7 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 8 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. ROC graphs show the tradeoffs between true positives and false positives and are commonly used to visualize the performance of classifiers (Fawcett, 2006). 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 more spammy than a randomly chosen legitimate email. 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 (2006).

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

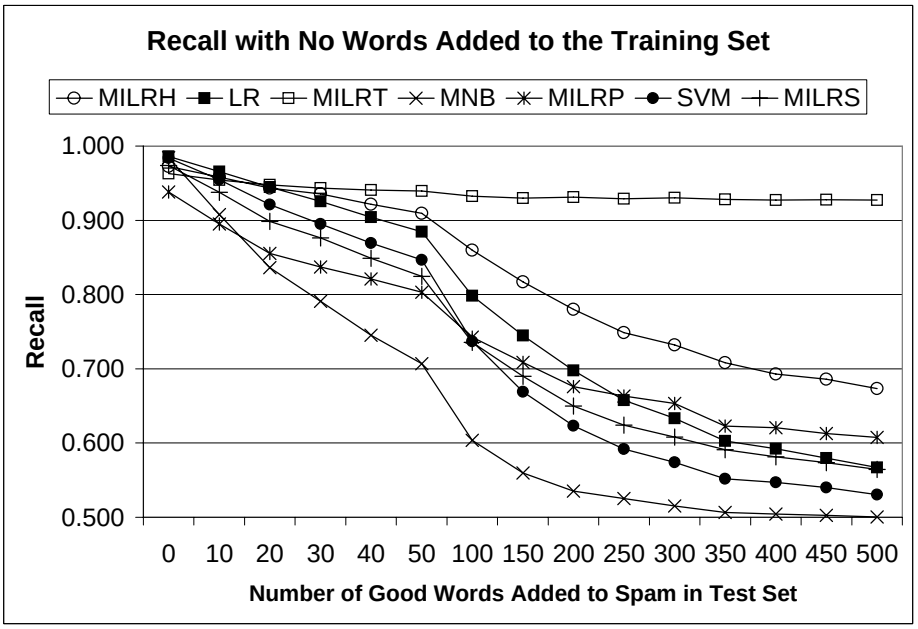
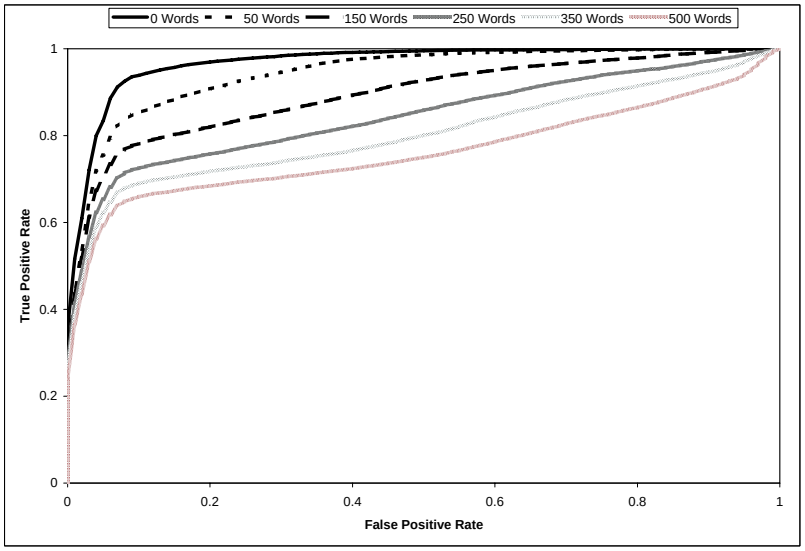


Figure 3: The change in average recall, corresponding to a fixed precision of 0.9, as the quantity of good words injected into half of the spam messages in the test set increases; no good words were injected into the training set.

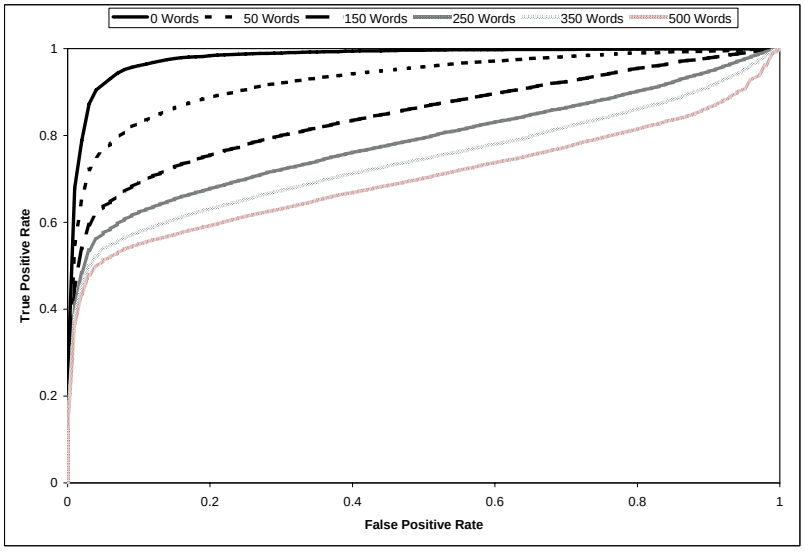
Words	MILRT	MILRH	MILRS	MILRP	LR	MNB	SVM
0	0.946	0.966	0.962	0.957	**0.981**	0.976	0.979
10	0.945	0.962	0.944	0.934	**0.968**	0.935	0.961
20	0.943	0.956	0.928	0.914	**0.958**	0.898	0.946
30	0.941	**0.953**	0.917	0.901	0.948	0.871	0.933
40	0.941	**0.949**	0.903	0.888	0.939	0.842	0.921
50	0.940	**0.943**	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

Table 2: Area Under the ROC Curve as the Quantity of Injected Good Words is Increased.

average recall of MNB and SVM dropped by 49.2% (from 0.984 to 0.500) and 46% (from 0.984 to

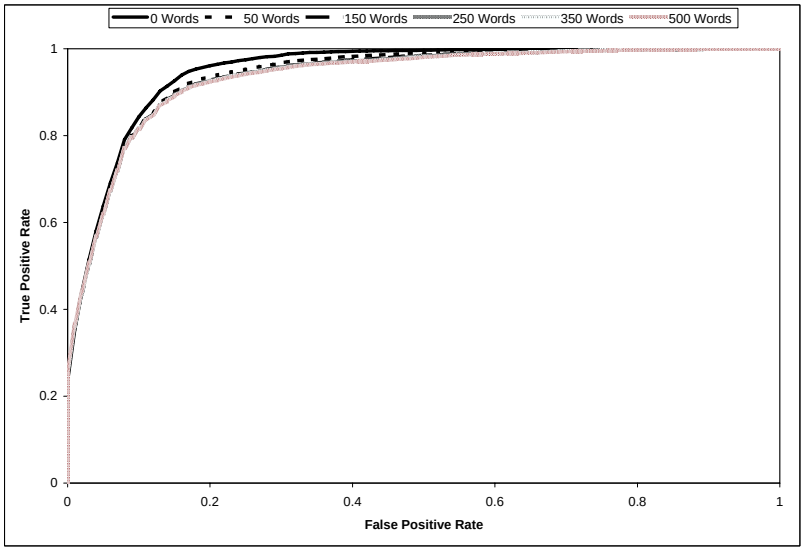


(a) MILRH

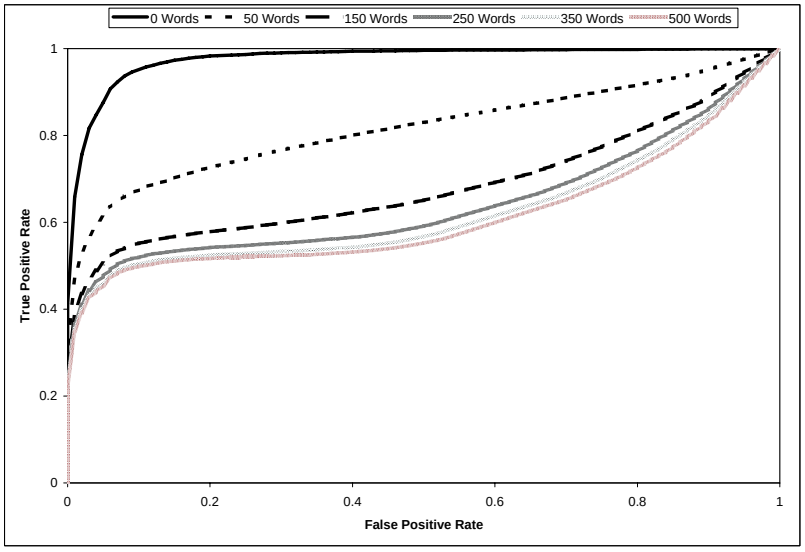


(b) LR

Figure 4: Average ROC curves of (a) MILRH and (b) LR when specific quantities of good words are injected into half of the messages in the test set.

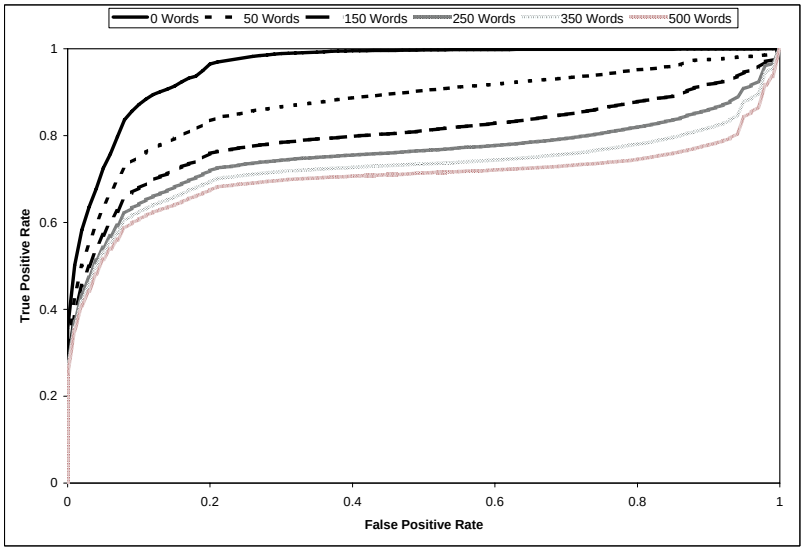


(a) MILRT

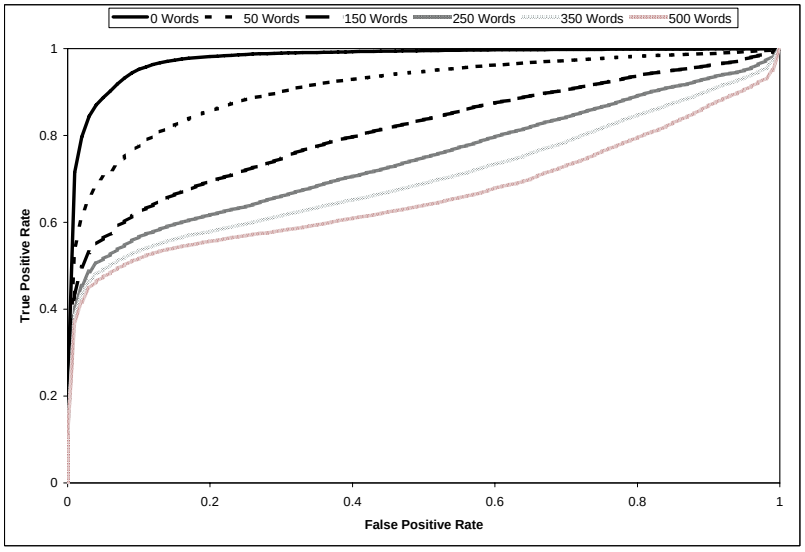


(b) MNB

Figure 5: Average ROC curves of (a) MILRT and (b) MNB when specific quantities of good words are injected into half of the messages in the test set.

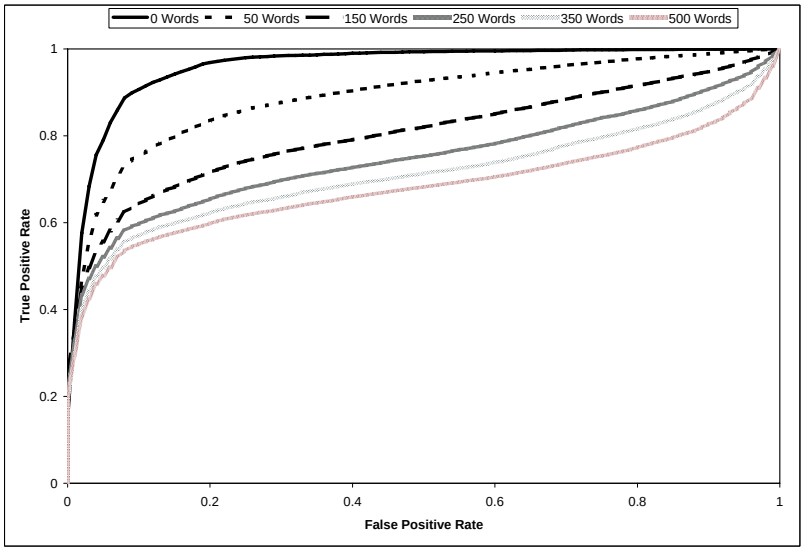


(a) MILRP



(b) SVM

Figure 6: Average ROC curves of (a) MILRP and (b) SVM when specific quantities of good words are injected into half of the messages in the test set.



MILRS

Figure 7: Average ROC curves of MILRS when specific quantities of good words are injected into half of the messages in the test set.

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.

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. 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

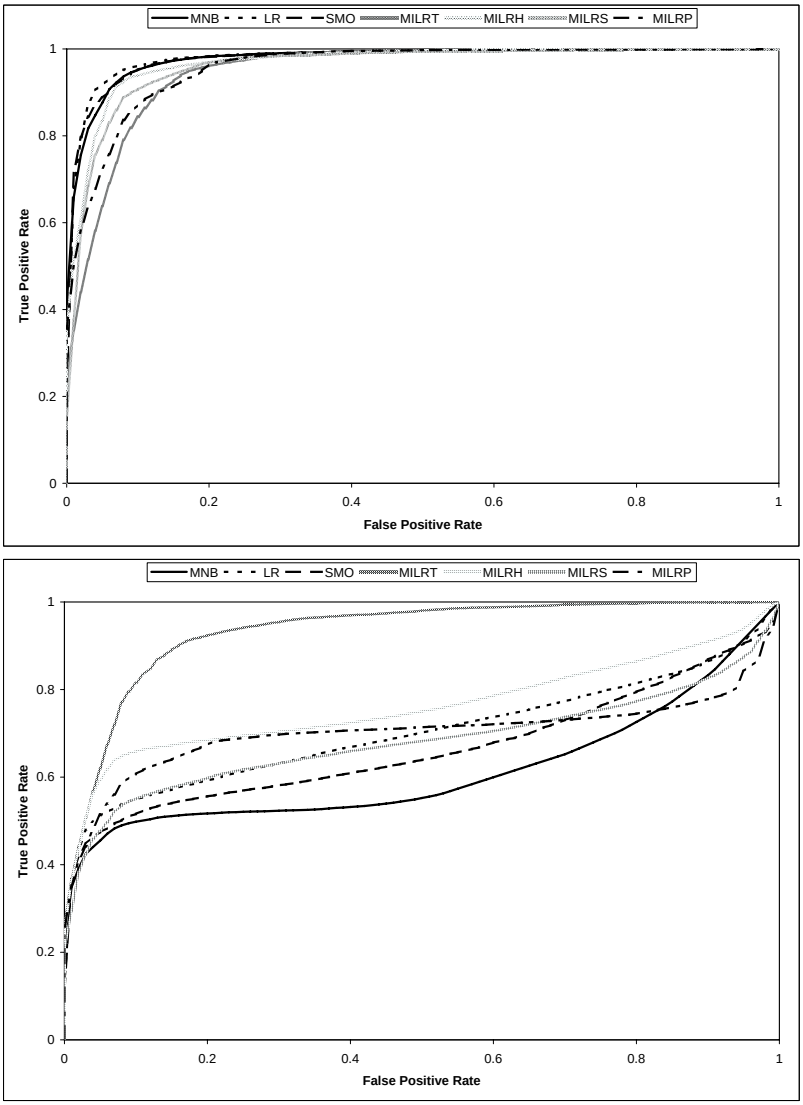


Figure 8: ROC curves for all classifiers when 0 words (top) and 500 words (bottom) have been injected into the test set.

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 9 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.

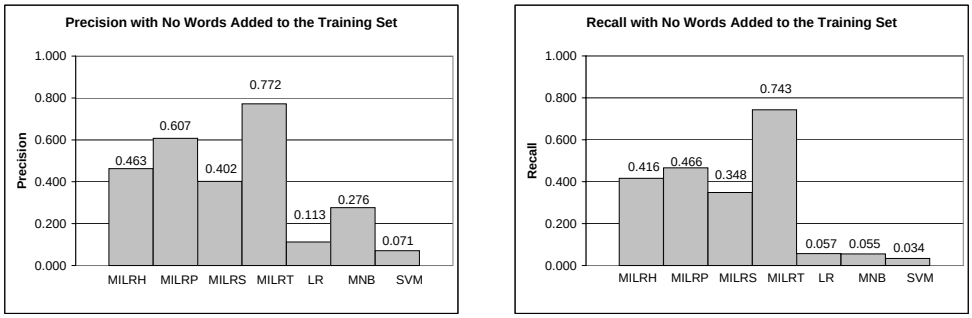


Figure 9: The average precision (left) and recall (right) of each classifier when injecting the entire local good word list into all of the spam messages in the test set.

7.2 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 shown here. 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 10– 14 show the change in average recall as the number of good words injected into the training set increased from 10 to 50. Figure 15 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 16 shows the same curves after 50 good words have been added to 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 3 and 10). 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.

Increasing the number of good words injected into the training set from 10 to 20 (see Figure 11) 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. These results

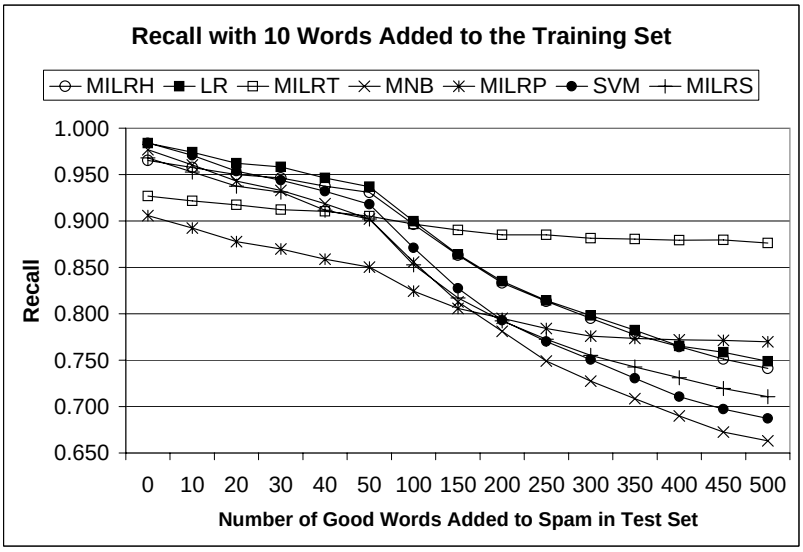


Figure 10: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 10 good words were also injected into half of the spam messages in the training set.

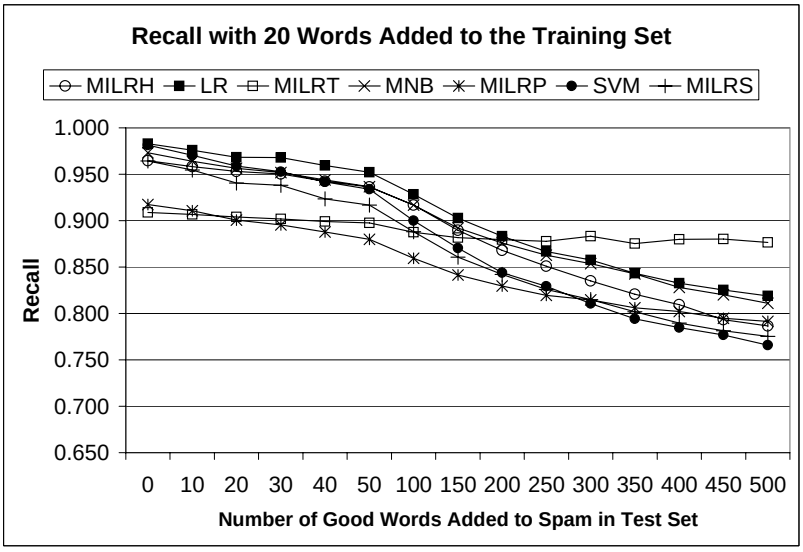


Figure 11: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 20 good words were also injected into half of the spam messages in the training set.

confirm the observations of several other researchers (Lowd and Meek, 2005b; Webb et al., 2005), 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.

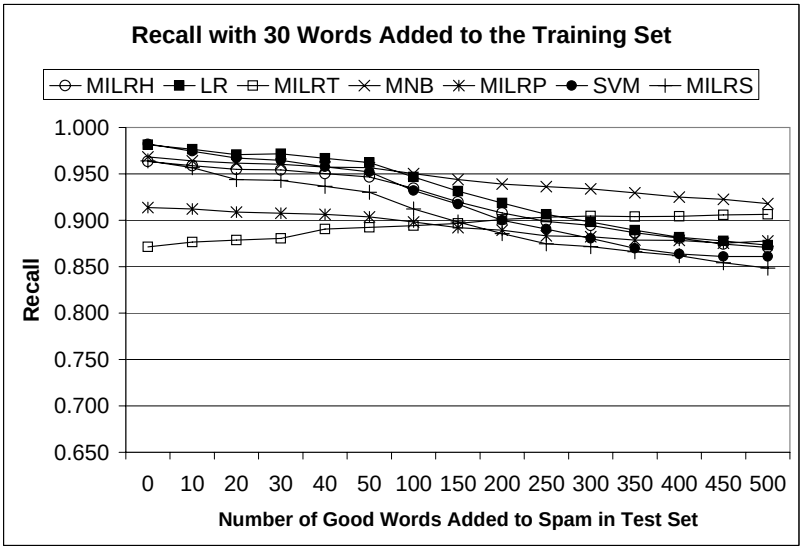


Figure 12: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 30 good words were also injected into half of the spam messages in the training set.

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 17 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 naive 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.

7.3 Attacking the Compression-Based Filter

Relatively new to the spam filtering scene is the idea of using statistical data compression algorithms for spam filtering. Bratko and Filipič (2005) proposed and investigated the use of character-level data compression models for spam filtering. The general idea is to construct two compression models, one from a collection of spam emails and one from a collection of legitimate emails, and

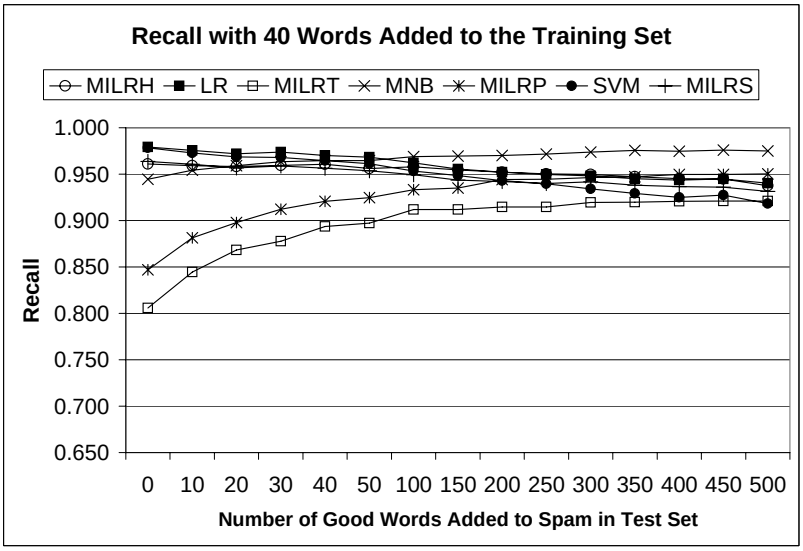


Figure 13: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 40 good words were also injected into half of the the spam messages in the training set.

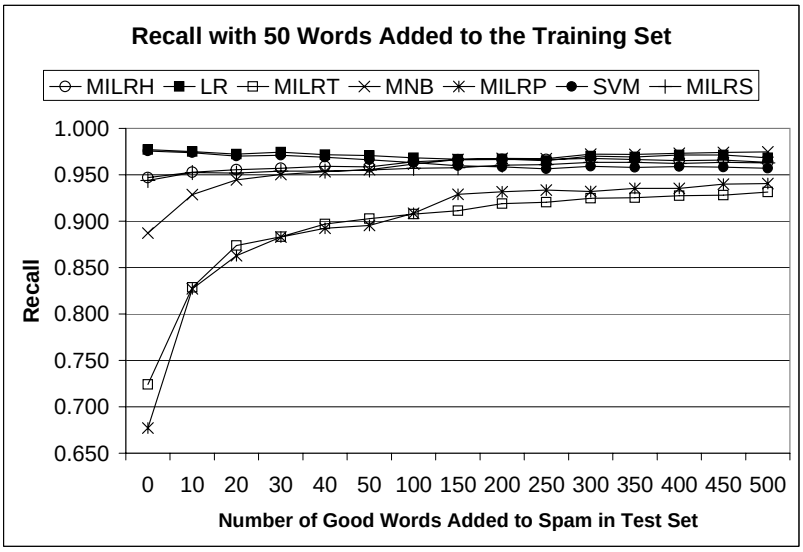


Figure 14: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 50 good words were also injected into half of the the spam messages in the training set.

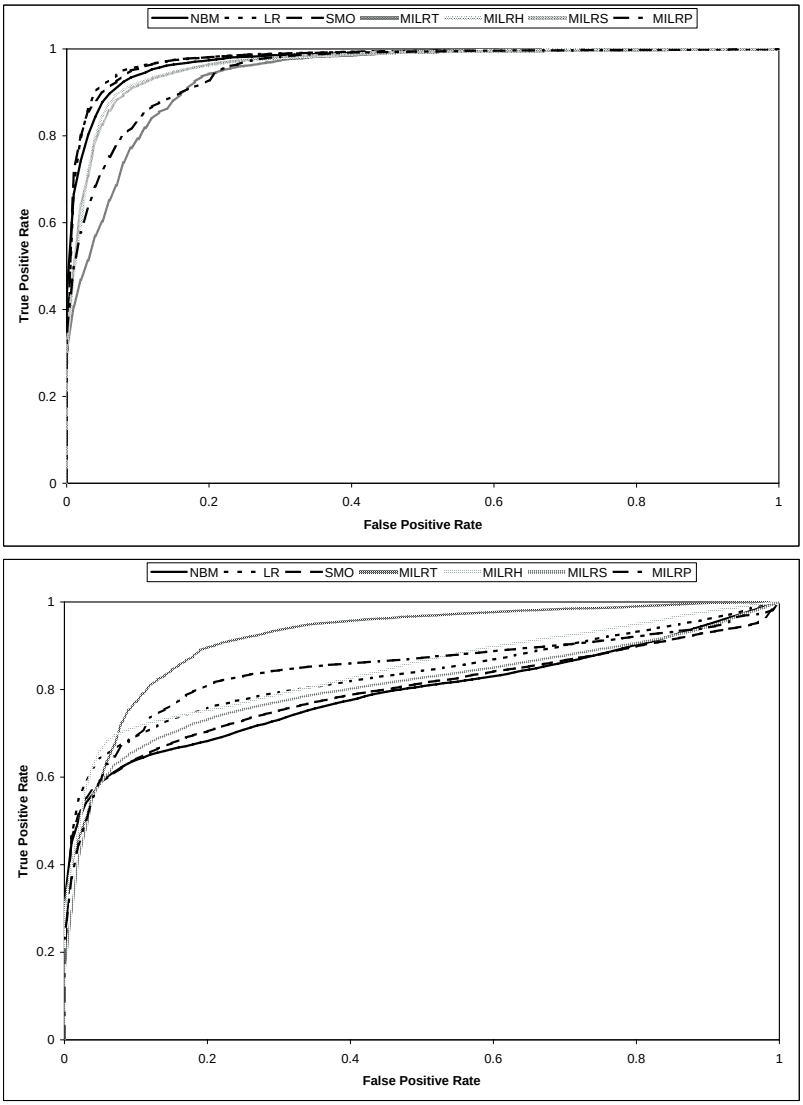


Figure 15: ROC curves for all classifiers when 0 good words (top) and 500 good words (bottom) have been injected into the test set and 10 good words have been injected into the training set.

then to classify a message according to which of the resulting models compresses the message more efficiently. Using statistical data compression for spam filtering has a number of advantages over machine learning algorithms that use word-level models. First, the compression algorithms work on the character level rather than the word level. For this reason, preprocessing and feature selection, both of which are highly prone to error, are unnecessary. Instead the algorithm is able to make full use of all message features. Another benefit of the character-level nature of the compression algorithms is that they are more robust to obfuscation. Spammers often disguise spammy words by deliberately misspelling them or by inserting punctuation between characters. This can cause

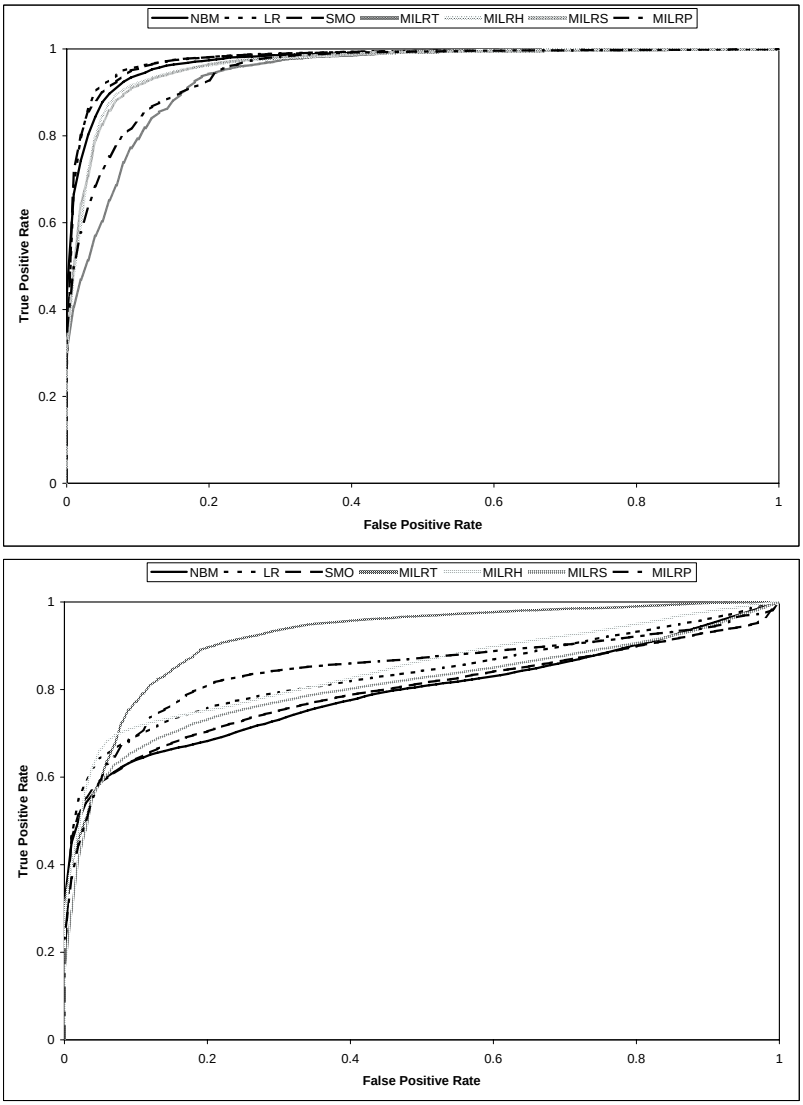


Figure 16: ROC curves for all classifiers when 0 good words (top) and 500 good words (bottom) have been injected into the test set and 50 good words have been injected into the training set.

a word-level spam filter to misclassify emails containing such words unless extra care and effort are expended to detect and deal with these obfuscations. Bratko and Filipič (2005) implemented their compression-based spam filter using the prediction by partial matching algorithm with escape method D (PPMD). Their experiments for the Trec 2005 spam track showed promising results. They also demonstrated that their filter was indeed quite robust to obfuscation. To our knowledge, however, the effects of the good word attack on such filters have not yet been investigated.

We repeated the two experiments from Sections 7.1 and 7.2 on the compression-based spam filter discussed by Bratko and Filipič (2005). Since preprocessing of the input corpus is unnecessary

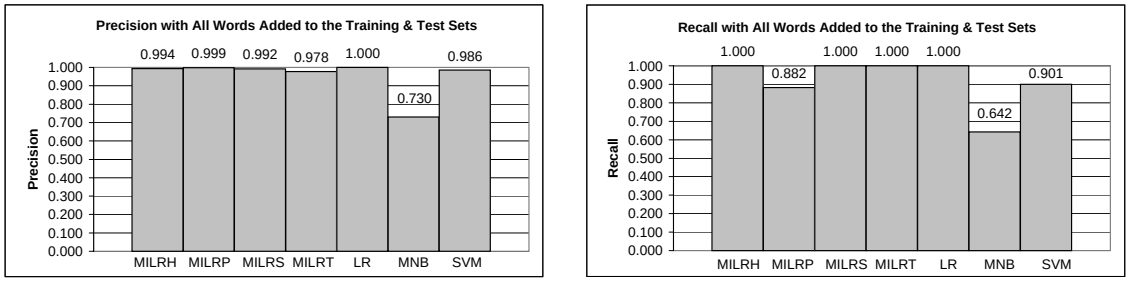


Figure 17: The average precision (left) and recall (right) of each classifier when injecting the entire contents of each local good word list into all of the spam messages in the corresponding training and test sets.

for this type of filter, we ran the experiments using the raw version of the corpus. However, we also ran the two experiments using the preprocessed corpus in order to observe how well the filter stood up against the attack using the same data available to the other filters. Figure 18 shows the results of the first experiment with the PPMD filter on the raw (PPMD1) and preprocessed (PPMD2) corpora. The attack had no effect on the precision of the filter, regardless of which version of the corpus was used; it remained consistently at 0.999 and is not shown on the chart. On the other hand, the recall suffered as a result of the attack, on both corpora. The decrease in recall on the preprocessed corpus was comparable to that of the single instance algorithms. When the raw corpus was used, however, the effect was much less severe. The compression-based filter implementation discussed by Bratko and Filipič (2005), which we used in these experiments, was set by default to truncate all messages to 2500 bytes, presumably for efficiency reasons. However, since our simulated attack appends good words to the bottom of the spam messages, it is possible that truncating the messages could result in some or all of the added good words being removed from spam messages that are longer than 2500 bytes. Therefore, we ran the experiment twice more, truncating at 5000 bytes and 10000 bytes. The result was a drop in average recall (when 500 good words are injected) to 0.702 when truncating at 5000 bytes and a drop to 0.627 when truncating at 10000 bytes, for PPMD1 (raw corpus) (see Figure 19). For PPMD2 (preprocessed corpus) the average recall dropped to 0.503 when truncating at 5000 bytes and dropped to 0.482 when truncating at 10000 bytes (see Figure 20). There was no additional drop in precision when truncating at 5000 or 10000 bytes.

Figure 21 shows the results of the second experiment on the PPMD filter, in terms of average recall, when 10 good words have been injected into the training set. Again, there was virtually no change in average precision so it is not shown on the charts. Apparently injecting 10 good words into the training set was enough to counter any number of good words in the test set. As figure 21 shows, the results of adding up to 50 good words to the training set are nearly identical.

Although it is difficult to directly compare the compression-based filter to the other filters discussed in this paper, due to differences in their modeling and preprocessing requirements, it is safe to say that the compression-based filter is susceptible to good word attacks. However, this type of filter also has a definite advantage over the other algorithms in that it is able to take advantage of message features that the other algorithms cannot easily use, making it more difficult to attack.

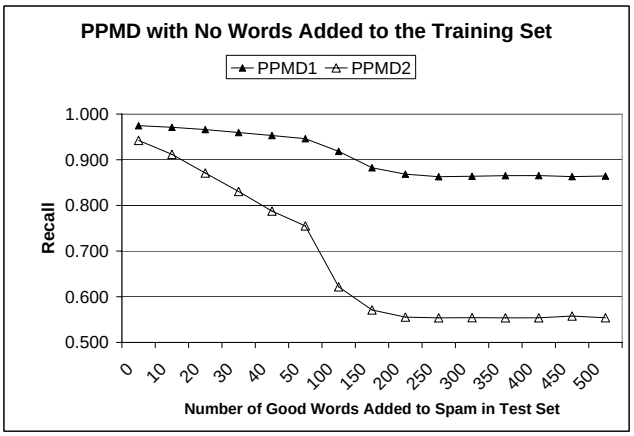


Figure 18: Effect of the good word attack on the PPMD algorithm as the number of good words added into half of the spam messages in the test set increases; no good words were added to the spam in the training set. Messages truncated at 2.5kB.

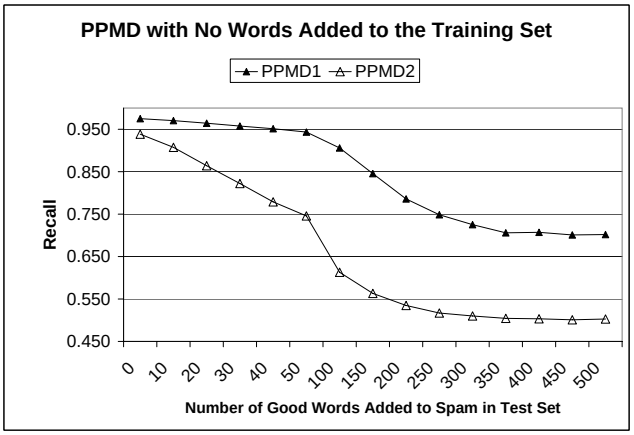


Figure 19: Effect of the good word attack on the PPMD algorithm as the number of good words added into half of the spam messages in the test set increases; no good words were added to the spam in the training set. Messages truncated at 5kB.

8. 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

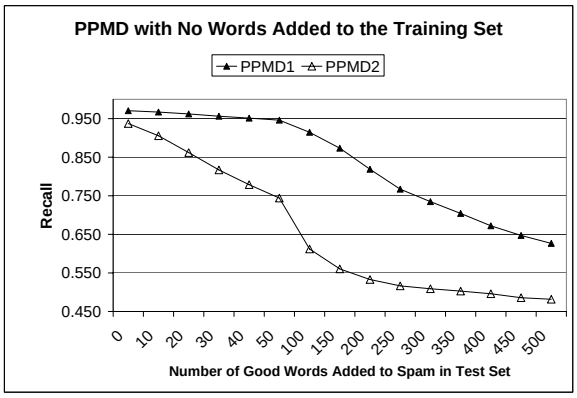


Figure 20: Effect of the good word attack on the PPMD algorithm when half of the spam messages in the test set were altered. Messages truncated at 10kB.

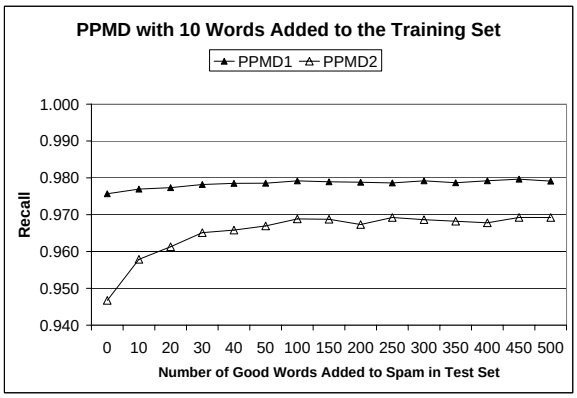


Figure 21: Effect of the good word attack on the PPMD algorithm when half of the spam in the training/test sets were altered; 10 words were added to the training spam.

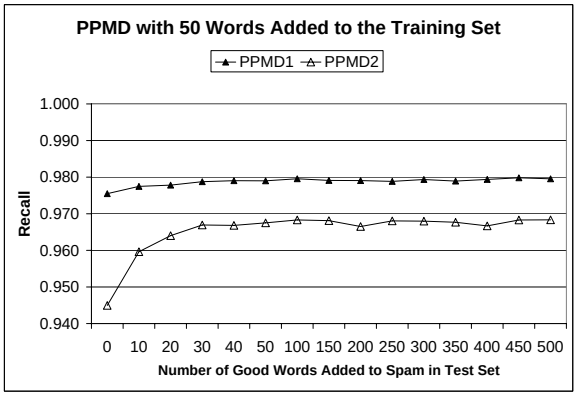


Figure 22: Effect of the good word attack on the PPMD algorithm when half of the spam in the training/test sets were altered; 50 words were added to the training spam.

the attacked message is split, good words should outweigh spammy words in both instances. The example in Table 3 illustrates the idea.

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

Table 3: Attacking split-H by fragmenting spam with injected 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 23 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 7.1) 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 3 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.

9. Conclusions and Future Work

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

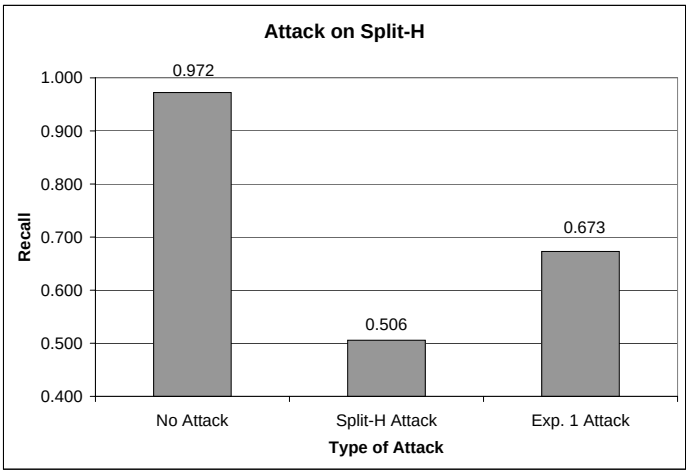


Figure 23: A comparison of the effects of the split-H attack and the experiment 1 style attack, in terms of average recall, with precision fixed at 0.9.

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 good word attacks of varying strength and has been shown to be effective. Additionally, we have confirmed earlier reports that re-training on attacked as well as normal emails may strengthen a spam filter against good word attacks. One of the advantages of our proposed strategy, as demonstrated by our experiments, is that it is effective even when trained on normal email and that frequent re-training on attacked messages is not necessary to maintain that effectiveness. We presented several possible methods for creating multiple instance bags from emails. As was observed from our experimental results, the splitting method used ultimately determines how well the strategy performs. The splitting methods we presented here work fairly well, especially the split-term method, but there are possibly other, perhaps better, methods that could be used. We plan to investigate other possible splitting methods in the future.

Since it is an arms race between spammers and filter designers, we also plan to make our MI strategy adaptive as new spam techniques are devised, and on-line as the concept of spam drifts over time. In addition, we plan to investigate the possibility of extending the proposed multiple instance learning strategy to handle similar adversarial attacks in other domains.

References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS 15*, pages 561–568. MIT Press, 2003.

P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1997. Morgan Kaufmann.

M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. 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, 2006. ACM Press. ISBN 1-59593-272-0. doi: http://doi.acm.org/10.1145/1128817.1128824.

A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30 (1):23–30, 1998.

A. Bratko. Probabilistic sequence modeling shared library. `http://ai.ijs.si/andrej/psmslib.html`, 2008.

A. Bratko and B. Filipič. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia, 2005.

J. Carpinter and R. Hunt. Tightening the net: A review of current and next generation spam filtering tools. *Computers and Security*, 25(8):566–578, 2006.

Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

Y. Chevaleyre and J.D. Zucker. 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, 2001.

G. V. Cormack and T. R. Lynam. Spam track guidelines — TREC 2005-2007. `http://plg.uwaterloo.ca/ gvcormac/treccorpus06/`, 2006.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM Press, 2004.

T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89(1-2):31–71, 1997.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

T. Gärtner, P. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186, San Francisco, CA, 2002. Morgan Kaufmann.

R. Jennings. The global economic impact of spam. Technical report, Ferris Research, 2005.

A.M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes. Multinomila naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

J.Z. Kolter and M.A. Maloof. 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, 2005. ACM Press.

H. Lee and A. Ng. Spam deobfuscation using a hidden Markov model. In *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.

P. Long and L. Tan. PAC learning axis-aligned rectangles with respect to product distribution from multiple-instance examples. *Machine Learning*, 30(1):7–21, 1998.

D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647. ACM Press, 2005a.

D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proceedings of the 2nd Conference on Email and Anti-Spam*, 2005b.

O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, 10:570–576, 1998.

J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection: 9th International Symposium (RAID)*, pages 81–105, 2006.

J. Ramon and L.D. Raedt. Multi instance neural networks. In *Proceedings of ICML-2000 workshop on Attribute-Value and Relational Learning*, 2000.

S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 697–704, New York, NY, 2005. ACM Press.

J. Rocchio Jr. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 68–73. Prentice Hall, 1971.

J. Wang and J.D. Zucker. Solving the multiple-instance learning problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1125, San Francisco, CA, 2000. Morgan Kaufmann.

S. Webb, S. Chitti, and C. Pu. An experimental evaluation of spam filter performance and robustness against attack. In *The 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 19–21, 2005.

I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, CA, USA, 2000.

X. Xu. Statistical learning in multiple instance problems. Master's thesis, University of Waikato, 2003.

X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *Proceedings of the Pacific-Asian Conference on Knowledge discovery and data mining*. Springer-Verlag, 2004.

W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam*, 2006.

M.-L. Zhang and Z.-H. Zhou. Multi-label learning by instance differentiation. In *The 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 669–674, Vancouver, Canada, 2007.

Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1073–1080, Cambridge, MA, 2002. MIT Press.

Z.H. Zhou and M.L. Zhang. Ensembles of multi-instance learners. In *ECML-03, 15th European Conference on Machine Learning*, pages 492–502, 2003.