

An Experimental Evaluation of Spam Filter Performance and Robustness Against Attack

Steve Webb, Subramanyam Chitti, and Calton Pu
{webb, chittis, calton}@cc.gatech.edu

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332

Abstract—In this paper, we show experimentally that learning filters are able to classify large corpora of spam and legitimate email messages with a high degree of accuracy. The corpora in our experiments contain about half a million spam messages and a similar number of legitimate messages, making them two orders of magnitude larger than the corpora used in current research. The use of such large corpora represents a collaborative approach to spam filtering because the corpora combine spam and legitimate messages from many different sources. First, we show that this collaborative approach creates very accurate spam filters. Then, we introduce an effective attack against these filters which successfully degrades their ability to classify spam. Finally, we present an effective solution to the above attack which involves retraining the filters to accurately identify the attack messages.

I. INTRODUCTION

Learning filters [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] have become a widely accepted technique for separating spam messages from an individual user’s incoming email stream. Significant research and development efforts have produced software tools (e.g., SpamProbe, SpamBayes, etc.) which can be used by individuals to effectively filter spam. In addition to empirical use and anecdotal evidence, small-scale (on the order of a few thousand messages) evaluations of personalized learning filters have corroborated their effectiveness for individual use.

In this paper, we present the first (to our knowledge) experimental evaluation of learning filter effectiveness using large corpora (about half a million known spam messages and a similar number of legitimate messages from public archives). Intuitively, the use of large corpora represents a “collaborative” approach to spam filtering. In contrast to filters trained by individuals with their own email, filters trained with large corpora are exposed to meaningful information from a variety of sources: legitimate messages from many individuals and spam messages from many sources to many destinations. In this paper, we study the strengths and weaknesses of these collaborative filters.

Our first contribution in this paper is a large-scale evaluation of learning filters that demonstrates their effectiveness with respect to known spam. In this evaluation, we observe that all of our filters achieve an accuracy consistently higher than 98%. However, we then show that this level of effectiveness

is vulnerable to attacks using *camouflaged messages* which combine both spam and legitimate content. Thus, our second contribution is a large-scale evaluation of the effectiveness of the camouflaged attack against our filters. This evaluation shows that the filters’ accuracy degrades consistently to between 50% and 75%. In response to this attack, we designed and evaluated several strategies for increasing the resilience of the filters. Our third contribution is showing that a particular strategy works: we can successfully identify camouflaged messages if the filters are explicitly trained to treat them all as spam. Using this strategy, the filters’ accuracy climbs back to between 86% and 96%.

Our large-scale evaluations are particularly useful for companies and Internet service providers considering the adoption of learning filters in their email gateways. Our results support this “upstream” migration of spam filtering as a successful way to reduce bandwidth and storage costs associated with spam. Similar to the attacks on keyword-based filters, our evaluation of camouflaged attacks shows that spammers can easily lower the effectiveness of learning filters. However, unlike the attacks on keyword-based filters, we actually provide a strategy which trains the learning filters properly so that they can resist attacks.

The rest of the paper is organized as follows. Section II summarizes our experimental setup and corpora processing. Section III illustrates the excellent performance of current spam filters with a normal training set and workload. Section IV describes the degraded performance of our filters when they are under attack by camouflaged messages and provides a successful solution. Section V shows that our solution does not harm the excellent performance shown in Section III. Section VI discusses related work, and Section VII concludes the paper and addresses our ongoing research.

II. EXPERIMENTAL SETUP

A. Spam Filters

In our experiments, we used four different spam filters: Naive Bayes [2], Support Vector Machine (SVM) [11], [3], LogitBoost [12], [4], and a Bayesian filter based on Paul Graham’s white paper [6]. This collection of filters represents the state of the art in machine learning (SVM and LogitBoost)

as well as the most popular and widely deployed spam filtering solutions (Naive Bayes and Paul Graham-based). We chose the SpamProbe 1.1x5 software package [13] for our Paul Graham-based filter because it is highly configurable and very efficient. Additionally, an independent comparison of Paul Graham-based filters [14] found SpamProbe to be one of the most accurate filters of its kind. For our Naive Bayes, SVM, and LogitBoost implementations, we chose the Weka software package [15]. Weka is an open source collection of machine learning algorithms which has become a standard implementation tool in the machine learning community.

We used two Bayesian filters because each uses a different event model. SpamProbe is based on the multinomial model, and Weka’s Naive Bayes implementation uses the multi-variate Bernoulli model. Previous research [16], [17] has shown that the multinomial model typically outperforms the multi-variate Bernoulli model, and our results show that this phenomenon also holds with large corpora and in the presence of an attack. We used a linear kernel for our SVM filter because previous research [18] has shown that simple, linear SVMs usually perform as well as non-linear ones. We used 100 iterations for our LogitBoost filter because previous research [12] showed that using more than 100 iterations results in a miniscule improvement. For more details about our filters’ settings as well as a brief description of each filter’s operation, please consult the Appendix.

B. Feature Selection

The feature space of a spam filter is the set of all tokens that occurs in any email message. For our corpora, the size of this feature space is greater than 300,000 (denoted as 300K for brevity) for a training set of 10K messages. Many of these tokens are not relevant to the distinction between spam and legitimate messages and introduce noise into the classification process. If these tokens were used while building classifiers, the classifiers would be overfitted to the data, introducing errors into the classification process. Additionally, in practice, it is infeasible to train a sophisticated machine learning algorithm such as SVM or LogitBoost in such a large feature space. Thus, in order to make classification accurate and training feasible, a small subset of features is selected from the original feature space, and then, the filters are trained on this subset. This process of selecting a small number of informative features from a large feature space is called dimensionality reduction or feature selection.

In our experiments using Weka, we used an information theoretic measure called Information Gain [19] to select a user-specified number n of features. First, the Information Gain was calculated for each token in the feature space, and then, the n tokens with the best score were selected. Finally, these n features were used to train the filters. SpamProbe does not use feature selection in the traditional sense. Instead, it uses the less sophisticated algorithm described in [6] in which tokens are selected based on the distance between their spam probability value and 0.5.

C. Corpora

Unlike previous evaluation experiments involving spam filters [1], [2], [3], [5], [4], [6], [7], [8], [9], [10], which were user-specific or anecdotal, our experiments were not directly related to a specific individual’s message set. Instead, we used very large, public corpora for our analysis.

1) *Message Collections*: Initially, we collected 750K spam messages from the publicly available spam corpora maintained by SpamArchive¹ and SpamAssassin². After the spam messages were collected, 800K legitimate messages were collected from a number of sources. First, 4K legitimate messages were collected from the legitimate corpus maintained by SpamAssassin.² Then, another 4K legitimate messages were taken from the authors’ personal mailboxes. Finally, a newsgroup crawler was used to obtain 792K messages from various publicly available newsgroups. Since a number of these newsgroups were not moderated, we used SpamProbe, trained with 8K randomly selected spam messages and the 8K legitimate email messages previously mentioned, to classify the newsgroup messages (refer to Section III for a validation of this method for filtering and labeling messages). The newsgroup messages that were labeled as spam by the filter were discarded, and afterwards, our collection of legitimate messages contained 700K messages.

In addition to the legitimate email corpus we collected, we also utilized the Enron corpus³ [20]. However, in our initial experiments with this corpus, we discovered a number of messages that appeared to be spam. To alleviate this problem, we used SpamProbe, trained with 5K randomly selected spam messages and 5K randomly selected legitimate messages, to classify the Enron messages. The messages that SpamProbe labeled as spam were discarded, and then, the remaining messages were manually inspected to ensure their legitimacy. After this process was complete, we were left with 475K Enron messages. Since the Enron corpus has become the standard legitimate email corpus amongst researchers, the results we present in this paper were obtained using our cleansed version of this corpus.

2) *Corpora Preprocessing*: Previous research [21], [10] claimed that significant performance improvements can be achieved by incorporating email headers in the filtering process. Our own experiments have also verified these results. However, we discovered that most of these performance improvements are caused by unfair biases in the corpora. An example of such a bias is the “Message-ID” header in the Enron corpus. Almost all of the messages in this corpus contain a “Message-ID” header which contains “Java-Mail.evans@thyme” as a substring. Consequently, if this header is included in the filtering process, it will unfairly bias each filter’s evaluation of messages taken from our Enron corpus. Another example of a bias introduced by the

¹SpamArchive’s spam corpora can be found at <ftp://spamarchive.org/pub/archives/>.

²SpamAssassin’s spam and legitimate corpora can be found at <http://spamassassin.org/publiccorpus/>.

³The Enron corpus can be found at <http://www-2.cs.cmu.edu/enron/>.

corpora is the “Received” header in our SpamArchive corpus. More than 85% of the messages in this corpus contain at least one “Received” header, and most of those messages contain multiple “Received” headers. However, “Received” headers are not found in the messages in the Enron corpus. Consequently, if this header is included in the filtering process, it will unfairly bias each filter’s evaluation of the messages taken from our SpamArchive corpus. Due to these and other biases introduced by the corpora, our experiments in this paper used messages with all but two of their headers removed: “Subject” and “Content-Type.”

3) *Message Representation*: A text message cannot be interpreted directly by some of our classifier building algorithms. Thus, an indexing procedure that maps a text message into a compact representation of its content needed to be uniformly applied to all of the messages in our corpora. We chose the “set of words” representation of a message as our indexing procedure.

All of our messages were tokenized using SpamProbe [13]. This tokenization resulted in each message being represented as a “set of words” (i.e., a tokenized message was represented as the set of tokens occurring in it). In previous research [22], [23], it was discovered that representations more sophisticated than this do not yield significantly better results.

III. BASELINE EVALUATION OF SPAM FILTER EFFECTIVENESS

A. Cost Sensitive Performance Measures

To accurately evaluate our spam filters’ performance, we need to incorporate the fact that classifying a legitimate message as spam (i.e., a false positive) is more costly to users than classifying a spam message as legitimate (i.e., a false negative). We are able to model this cost disparity using a performance metric called Weighted Accuracy ($WAcc$) [5]. $WAcc$ assigns a weight of λ to legitimate messages, treating each legitimate message as if it were λ messages. Thus, if a legitimate message is (un)successfully classified, it counts as if λ legitimate messages are (un)successfully classified. The equation for $WAcc$ is the following:

$$WAcc = \frac{\lambda n_{L \rightarrow L} + n_{S \rightarrow S}}{\lambda \cdot N_L + N_S}.$$

In the above equation, $n_{L \rightarrow L}$ is the number of correctly classified legitimate messages, $n_{S \rightarrow S}$ is the number of correctly classified spam messages, N_L is the total number of legitimate messages, and N_S is the total number of spam messages. In this paper, most of the figures will demonstrate our filters’ performance by showing their $WAcc$ values with $\lambda = 9$. Our results are also available using additional performance measures (e.g., Total Cost Ratio, Precision, etc.) and additional values for λ in [24], but they are omitted in this paper to conserve space.

B. Evaluation of the Training Set Size

To thoroughly investigate the effect of the training set size on our filters, we performed an experiment with five different training set sizes: 500, 1K, 5K, 10K, and 50K.

First, we trained the filters with each training set, and then, we used the filters to classify a workload that consisted of 10K messages. Half of the messages used in the training sets and workload were randomly selected from our SpamArchive corpus, and the other half were randomly selected from our Enron corpus. Additionally, for these experiments, we retained 320 spam features and 320 legitimate features (using the methods explained above in Section II-B).

Figure 1 illustrates the filters’ average $WAcc$ results for $\lambda = 9$ after 10 iterations of this experiment. SpamProbe, SVM and LogitBoost all exhibit similar performance, but Naive Bayes performs slightly worse than the others. As the number of messages in the training set increases, the filters’ performance also increases. However, for training sizes larger than 10K, we found that the filters’ performance improves only modestly, failing to justify the additional training time necessary to use those larger training sizes. Consequently, for the remainder of this paper, unless stated otherwise, we used 10K as the training set size for our experiments. We now turn our attention to the workload size.

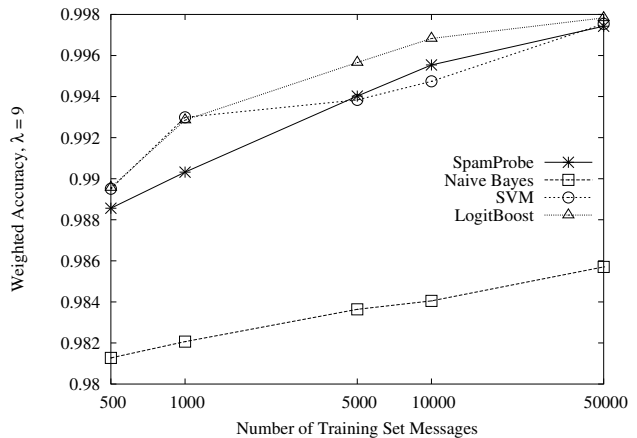


Fig. 1. Average $WAcc$ results ($\lambda = 9$) for the baseline evaluation using a 10K message workload and 640 retained features.

C. Evaluation of the Workload Size

To analyze the effect of the workload size on our filters, we performed an experiment with five different workload sizes: 500, 1K, 5K, 10K, and 50K. First, we trained our filters with a training set of 10K messages. Then, we used the filters to classify the various workloads. Half of the messages in the training set and workloads were spam, and the other half were legitimate. Additionally, we retained 640 features (320 legitimate and 320 spam).

Figure 2 shows the filters’ average $WAcc$ results for $\lambda = 9$ after 10 iterations of this experiment. For smaller workload sizes (i.e., 500, 1K, and 5K), the filters demonstrate slightly fluctuating results; however, when the workload size is greater than 5K, the filters’ performance is extremely stable. Additionally, the similarities between the results for 10K messages and 50K messages indicate that our corpora are internally consistent. Based on these results, we chose to use a workload size

of 10K messages for the remaining experiments in this paper. Now that we have experimentally determined an appropriate training set and workload size, we will investigate various feature sizes.

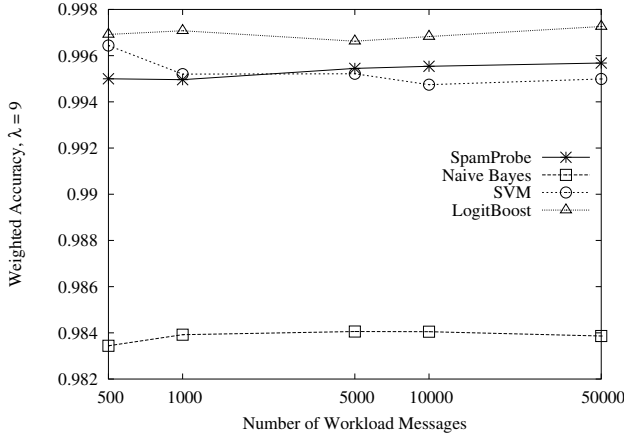


Fig. 2. Average $WAcc$ results ($\lambda = 9$) for the baseline evaluation using a 10K message training set and 640 retained features.

D. Evaluation of Feature Size

In the previous two experiments, our filters used 640 retained features. Our next experiment explores the effect of the number of retained features on the filters by using seven different feature sizes: 10, 20, 40, 80, 160, 320, and 640. In this experiment, we trained the filters with a training set of 5K spam messages and 5K legitimate messages. Then, we used the filters to classify a workload of 5K spam messages and 5K legitimate messages. The only variable was the total number of retained features.

Figure 3 displays the filters’ average $WAcc$ results for $\lambda = 9$ after 10 iterations of this experiment. All of the filters are quite successful when classifying the spam and legitimate messages in the workload. SpamProbe and Naive Bayes both demonstrate their best performance with lower feature sizes, but SpamProbe’s performance is consistently better. SVM and LogitBoost exhibit very similar behavior, and they both perform significantly better with larger feature sizes (i.e., more than 80 features).

These results concretely demonstrate the filters’ ability to correctly classify the spam and legitimate messages in our large email corpora. However, in the next section, we present an attack which significantly degrades our filters’ classification performance.

IV. EVALUATION OF SPAM FILTER EFFECTIVENESS AGAINST CAMOUFLAGED MESSAGES

A. Camouflaged Messages

The baseline results presented in Section III clearly illustrate the effectiveness of learning filters when distinguishing between spam and legitimate messages. However, we have recently observed a new class of messages that our filters are unable to correctly classify. These new messages, which we

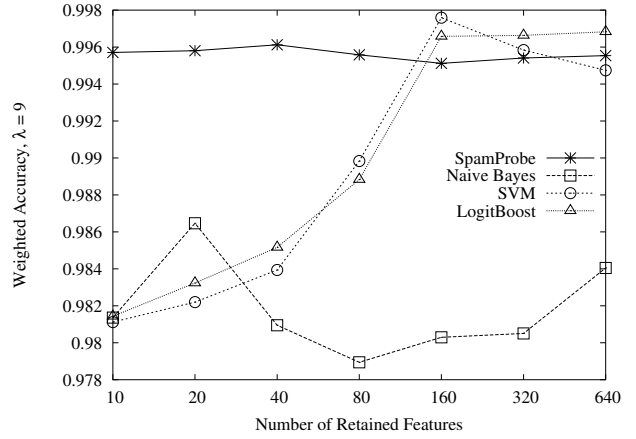


Fig. 3. Average $WAcc$ results ($\lambda = 9$) for a 10K message baseline training set and a 10K message baseline workload.

will refer to as camouflaged messages, contain spam content as well as legitimate content, and as a result, they are able to confuse our filters. The following sections describe experiments that quantitatively evaluate the influence of camouflaged messages on our learning filters. In these experiments, we vary the amount of camouflage (i.e., the relative proportions of the spam and legitimate content) used in the messages, and we also vary how the camouflaged messages are used in the training set and workload.

The camouflaged messages used in these experiments were generated using messages from the original training set of 10K messages and original workload of 10K messages used by the baseline experiment in Section III-D. Each camouflaged message was created by combining parts of a spam message with parts of a legitimate message. For each pair of spam and legitimate messages $\{s, l\}$, two camouflaged messages $\{c_1, c_2\}$ were created. c_1 was created by combining a larger portion of spam content with a smaller portion of legitimate content, and c_2 was created similarly by combining a larger portion of legitimate content with a smaller portion of spam content. In the experiments for this paper, c_1 (c_2) contained twice as much spam (legitimate) content as legitimate (spam) content. However, this ratio was one of the parameters we varied in our experiments. For the results using other ratios, please consult [24].

Before continuing to our experimental evaluation, an important observation must be made about the experiments involving camouflaged messages. In the baseline experiments, the messages were taken from known collections of spam and legitimate messages, and as a result, the training and evaluation of the filters was straightforward. However, it is unclear how we should determine the “spamicity” or “legitimacy” of camouflaged messages because they contain both spam and legitimate content. This uncertainty becomes even more pronounced as the amount of legitimate content found in these messages increases from 0% to 100%. At what point do we “draw the line” to distinguish between a spam message and a legitimate message? To address this question, we present a

threshold t which is used to quantify the amount of legitimate content a camouflaged message must contain in order to be identified as a legitimate message. A camouflaged message is identified as legitimate if and only if the proportion of its legitimate content is greater than or equal to t . Using this heuristic, we find that three distinct scenarios emerge. When $t = 0\%$, all camouflaged messages are treated as legitimate; when $0\% < t < 100\%$, the camouflaged messages are identified based on the proportion of their spam and legitimate content, and when $t = 100\%$, all of the camouflaged messages are treated as spam.

We exhaustively explored the design space for all three of these scenarios. By treating all camouflaged messages as spam (i.e., when $t = 100\%$) and carefully selecting the training set, we show in Section IV-B.2 that the filters can accurately identify camouflaged messages. However, similar results are not achieved in the other two scenarios. The results of our experimental evaluation of those other two scenarios are presented in [24].

B. Identifying All Camouflaged Messages as Spam

In this section, we evaluate our filters' ability to identify all camouflaged messages as spam using a series of experiments.

1) *Original Training with Camouflaged Workload*: In our first experiment, using a training set of only original messages, we analyzed the filters' ability to correctly classify a workload consisting entirely of camouflaged messages. Each filter was trained with the original training set of 10K messages used in the baseline experiment. Then, the filters were used to classify a workload consisting of 10K camouflaged messages which were created using the original workload of 10K messages used in the baseline experiment in Section III-D.

When we identify all of the camouflaged messages in the workload as spam, $WAcc$ simplifies to spam recall because $n_{L \rightarrow L}$ and N_L are zero⁴. As a result, we use spam recall as the evaluation metric for these camouflaged experiments.

Figure 4 shows the filters' average spam recall results after 10 iterations of this experiment. This figure clearly shows that our original filters are unable to successfully classify the camouflaged messages in the workload as spam. SpamProbe consistently outperforms the other filters, but its best average spam recall value is only 76.57%. The Naive Bayes filter performs the worst, consistently classifying only around 50% of the messages as spam. As in previous experiments, SVM and LogitBoost mimic each other's performance, but their highest average spam recall values are only 63.38% and 64.19%, respectively.

Obviously, when our filters only use original messages in their training set, they are incapable of correctly classifying the camouflaged messages in the workload as spam. Thus, to

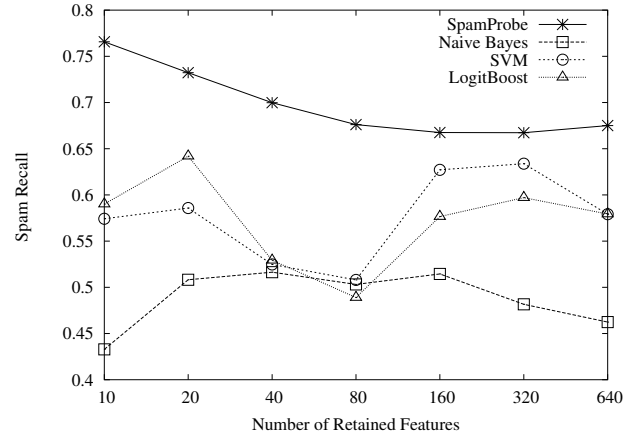


Fig. 4. Average spam recall results for a 10K message baseline training set and a 10K message camouflaged workload.

improve this situation, we performed additional experiments in which we varied the filters' training sets.

2) *Camouflaged Training with Camouflaged Workload*: Our next experiments attempt to correctly classify camouflaged messages by introducing camouflaged messages into the filters' training sets. By training the filters with camouflaged messages, we expected to increase the filters' ability to successfully recognize camouflaged messages in the workload. To evaluate this expectation, we conducted two experiments.

The first experiment used a training set of 10K camouflaged messages which were created using the original training set of 10K messages used in the baseline experiment in Section III-D. Initially, the filters were trained to recognize all of the camouflaged messages as spam. Then, the filters were used to classify the same workload of 10K camouflaged messages used in the previous experiment in Section IV-B.1.

We omit this experiment's results because they are virtually identical to the results presented in the next experiment. Also, the filters created in this experiment are unable to correctly identify legitimate messages because they have only been trained to recognize camouflaged messages as spam. We save the discussion of this second point for Section V.

The second experiment used the same camouflaged training set as the previous experiment, but it also used the original training set of 10K messages used in the baseline experiment. Thus, the filters were trained with 10K original messages as well as 10K camouflaged messages. Then, the trained filters were used to classify the same camouflaged workload used in the previous two experiments.

Figure 5 displays the filters' average spam recall values after 10 iterations of this experiment. By comparing these results to Figure 4, we see that all of the filters drastically improved their average spam recall values in this experiment. SpamProbe exhibited the smallest performance increase, but its best average spam recall value is still 87.49%. The Naive Bayes filter improved the most, increasing its best average spam recall value from 51.64% to 86.70%. SVM and LogitBoost both perform extremely well, correctly identifying more than

⁴ $WAcc$ simplifies to spam recall:

$$WAcc = \frac{\lambda \cdot n_{L \rightarrow L} + n_{S \rightarrow S}}{\lambda \cdot N_L + N_S}$$

$$\frac{\lambda \cdot (0) + n_{S \rightarrow S}}{\lambda \cdot (0) + N_S} = \frac{n_{S \rightarrow S}}{N_S} = \text{spam recall}$$

96% (96.38% and 96.19%, respectively) of the camouflaged messages in the workload as spam.

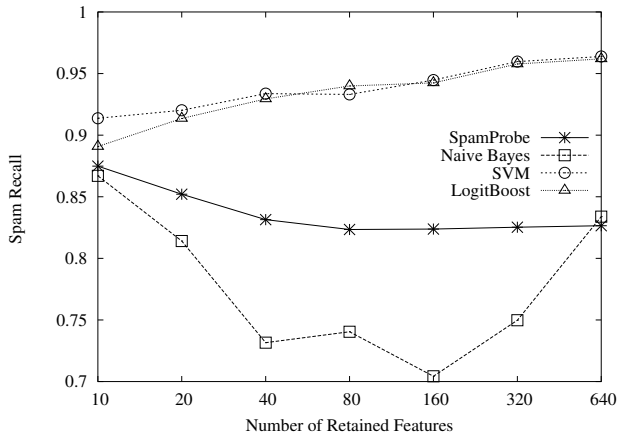


Fig. 5. Average spam recall results for a 10K message baseline training set, a 10K message camouflaged training set, and a 10K message camouflaged workload.

These results show that if we train our filters to treat all camouflaged messages as spam, they will successfully classify the camouflaged messages in the workload as such. Thus, we have discovered a solution for successfully identifying camouflaged messages.

V. BASELINE EVALUATION OF SPAM FILTER EFFECTIVENESS REVISITED

In Section IV, we evaluated techniques for classifying camouflaged messages in the filters’ workloads. In this section, we take that evaluation a step further in order to determine the effect our proposed solutions have on each filter’s baseline performance with the original workload. If one of the proposed solutions sacrifices the filters’ ability to classify messages in the original workload, its overall worth is significantly discounted.

The series of experiments we used to analyze each proposed solution’s effect on the baseline performance is very similar to the series of experiments performed in Section IV-B.2. The training set used in each of the experiments in this series is the same as the training set in the corresponding experiment in the previous series. However, instead of using the camouflaged workload of 10K messages used in the previous series, this series’ experiments used the original workload of 10K messages used in the baseline experiment.

The first experiment in this series used the filters that were only trained to treat all camouflaged messages as spam. We omit the figure for this experiment’s results because the filters created in this experiment perform so poorly with the baseline workload. As explained in Section IV-B.2, these filters are unable to correctly classify legitimate messages because they have only been trained to identify spam. Since this proposed solution dramatically reduces the filters’ baseline performance, it must be rejected.

The results obtained in the first experiment highlight an important point about our proposed solutions. To be a truly

effective solution, the training method must create filters which successfully classify camouflaged messages while also maintaining the filters’ baseline performance. With that in mind, we now describe our final experiment.

The last experiment used the filters that were trained with the original training set and trained to treat all camouflaged messages as spam. Figure 6 shows the average $WAcc$ results for $\lambda = 9$ after 10 iterations of this experiment. By comparing Figures 6 and 3, we discover that the average $WAcc$ values for the filters in this experiment are very similar to the corresponding values in the baseline experiment. SpamProbe’s performance experienced almost no change from the baseline experiment. SVM and LogitBoost experienced a modest performance decrease, but the decline in performance was only a couple percentage points.

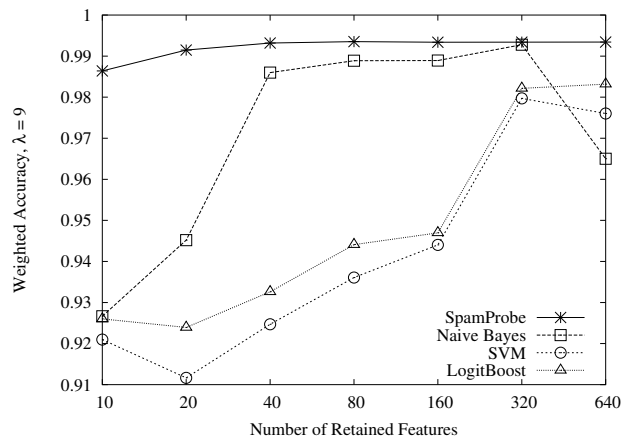


Fig. 6. Average $WAcc$ results ($\lambda = 9$) for a 10K message baseline training set, a 10K message camouflaged training set, and a 10K message baseline workload.

These results illustrate the filters’ ability to correctly classify the original workload when they are trained with the original training set and trained to treat all camouflaged messages as spam. Additionally, since we concluded in Section IV-B.2 that these filters could correctly classify camouflaged messages as spam, we have experimentally proven that this proposed solution is truly effective at accurately classifying both camouflaged and original messages.

VI. RELATED WORK

The use of machine learning algorithms in spam filters is a well established idea. Previous research [1], [2], [5], [7] has shown that the Naive Bayes algorithm can be used to build very effective personalized spam filters. Similar results have also been obtained for Support Vector Machines [3], [7], [10] and Boosting algorithms [4], [7], [10]. However, our work is unique in that we study the effectiveness of spam filtering on a very large scale. We used corpora that are two orders of magnitude larger than those used in previous evaluations, and as a result, we have shown that machine learning techniques can be used to successfully filter spam at the gateway.

We are not the first to suggest that learning filters are potentially vulnerable to attack. Graham-Cumming gave a

talk at the 2004 MIT Spam Conference [8], in which he described an attack against an individual user's spam filter. This attack inserted random words into spam messages. A previous paper [9] attempted to expand this attack by adding commonly occurring English words instead of random words. Our research differs from this previous work in a number of ways. First, our described attack (i.e., camouflaged messages) uses portions of real legitimate email, and as a result, it creates more confusion than the previously mentioned attacks. Second, our evaluation incorporated a larger set of learning filters. Finally, we focused our attention on gateway filtering, not user-specific filtering.

Attacking a spam filter can be thought of as an example of how machine learning algorithms can be defeated when an adversary has control of the workload. Thus, this paper is similar in spirit to [25], in which the authors used game theory to model attacks on spam filters. In that paper, the emphasis was placed on the game theoretic modeling; however, we showed that in the domain of spam filtering, a spammer can successfully attack current filters without using sophisticated game theoretic methods.

VII. CONCLUSION AND ONGOING RESEARCH

In this paper, we used large corpora (half a million known spam and about the same number of legitimate messages from public collections) to evaluate learning filter effectiveness in spam filtering. The use of such large corpora represents a collaborative approach to spam filtering because it combines many sources in the training of filters. First, we evaluated a production-use filtering tool (SpamProbe) and three classic machine learning algorithms (Naive Bayes, Support Vector Machine (SVM), and LogitBoost). We found these filters to be highly effective in identifying spam and legitimate email, with an accuracy above 98%. Second, we described a simple method of attacking these learning filters with camouflaged messages which significantly lowered the effectiveness of all the learning filters in our large-scale experiments. Third, by exploring the design space of training strategies, we found that by treating all camouflaged messages as spam during the training phase, we can restore the filters' effectiveness to within a couple percentage points of their baseline accuracy.

We are currently investigating more sophisticated attacks against learning filters. In this paper, we simply mixed spam content with legitimate content to create the camouflaged messages. This attack strategy is "neutral" with respect to (i.e., unaware of) the filter training. One could apply the ideas similar to adversarial classification [25] to design more intelligent attack strategies that may defeat the all-spam training method described in Sections IV-B and V. We are currently designing appropriate responses to these sophisticated attacks and evaluating their effectiveness.

ACKNOWLEDGEMENTS

This work was partially supported by NSF under the ITR (grants CCR-0121643 and CCR-0219902) and CyberTrust/DAS (grant IIS-0242397) programs.

REFERENCES

- [1] P. Pantel and D. Lin, "Spamcop: A spam classification & organization program," in *Proc. AAAI Workshop on Learning for Text Categorization*, 1998.
- [2] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," in *Proc. AAAI Workshop on Learning for Text Categorization*, 1998, pp. 55–62.
- [3] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.
- [4] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," in *Proc. 4th International Conference on Recent Advances in Natural Language Processing (RANLP 2001)*, 2001, pp. 58–64.
- [5] I. Androutsopoulos *et al.*, "An evaluation of naive bayesian anti-spam filtering," in *Proc. Workshop on Machine Learning in the New Information Age, 11th European Conf. on Machine Learning*, 2000, pp. 9–17.
- [6] P. Graham. (2002) A plan for spam. [Online]. Available: <http://www.paulgraham.com/spam.html>
- [7] I. Androutsopoulos, G. Paliouras, and E. Michelakis, "Learning to filter unsolicited commercial e-mail," National Center for Scientific Research "Demokritos", Tech. Rep. 2004/2, 2004.
- [8] J. Graham-Cumming, "How to beat a bayesian spam filter," MIT Spam Conference, 2004.
- [9] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters," in *Proc. 1st Conference on Email and Anti-Spam (CEAS 2004)*, 2004.
- [10] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 4, pp. 243–269, 2004.
- [11] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
- [12] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–374, 2000.
- [13] B. Burton. (2004) Spamprobe version 1.1x5. [Online]. Available: <http://sourceforge.net/projects/spamprobe/>
- [14] G. Cormack and T. Lynam. (2004) A study of supervised spam detection applied to eight months of personal email. Submitted for publication. [Online]. Available: <http://plg.uwaterloo.ca/~gvcormac/spamcormack.html>
- [15] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, 2000.
- [16] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proc. AAAI Workshop on Learning for Text Categorization*, 1998, pp. 41–48.
- [17] K. Schneider, "A comparison of event models for naive bayes anti-spam e-mail filtering," in *Proc. 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 03)*, 2003, pp. 307–314.
- [18] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. 10th European Conference on Machine Learning (ECML 98)*, 1998, pp. 137–142.
- [19] Y. Yang and J. O. Pederson, "A comparative study of feature selection in text categorization," in *Proc. 14th International Conference on Machine Learning (ICML 97)*, 1997, pp. 412–420.
- [20] B. Klimt and Y. Yang, "Introducing the enron corpus," in *Proc. 1st Conference on Email and Anti-Spam (CEAS 2004)*, 2004.
- [21] P. Graham. (2003) Better bayesian filtering. [Online]. Available: <http://www.paulgraham.com/better.html>
- [22] C. Apte, F. Damerau, and S. M. Weiss, "Automated learning of decision rules for text categorization," *Information Systems*, vol. 12, no. 3, pp. 233–251, 1994.
- [23] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proc. 7th International Conference on Information and Knowledge Management (CIKM 98)*, 1998, pp. 148–155.
- [24] S. Webb, S. Chitti, and C. Pu, "An experimental evaluation of spam filter performance and robustness against attack," College of Computing, Tech. Rep., 2005.
- [25] N. Dalvi *et al.*, "Adversarial classification," in *Proc. 10th Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2004)*, 2004, pp. 99–108.

APPENDIX

Spam filtering can be characterized as the problem of assigning a boolean value (“Spam” or “Legitimate”) to each message m in a collection of messages M . More formally, the task of spam filtering is to approximate the unknown target function $\Phi : M \rightarrow \{Spam, Legitimate\}$, which describes how messages are to be classified, by means of a function $\phi : M \rightarrow \{Spam, Legitimate\}$, called the classifier (a.k.a. rule, hypothesis, or model), such that Φ and ϕ coincide as much as possible.

In the machine learning approach to spam filtering, a general inductive process (also called the learner) automatically builds a classifier by observing a set of documents manually classified as “Spam” or “Legitimate” (the training set). In machine learning terminology, this classification problem is an example of supervised learning since the learning process is supervised by the knowledge of the category of each message used during training.

Different learning methods have been explored by the research community for building spam classifiers (also called spam filters). In this paper, we focused on three learning algorithms which have proved to be effective in spam filtering: Naive Bayes [1], [2], [5], Support Vector Machine (SVM) [3], and LogitBoost [4].

Naive Bayes: The Naive Bayes learning algorithm is one of the simplest and most widely used. Given a message m , represented as a vector of features, we can use Bayes Theorem and the Theorem of Total Probability to calculate the probability that the message is either “Spam (S)” or “Legitimate (L)”:

$$P(m = c | \vec{X} = \vec{x}) = \frac{P(m=c) \cdot P(\vec{X} = \vec{x} | m=c)}{\sum_{c' \in \{S, L\}} P(m=c') \cdot P(\vec{X} = \vec{x} | m=c')}.$$

The classification of x is the category c that maximizes the above equation. Note that the denominator is the same for all categories and can be ignored. The most important attributes are the a priori probabilities of each category and the a posteriori probabilities of the vectors, given the category, which need to be estimated from the training data. The number of possible vectors (i.e., all combinations of different feature values) is very large (i.e., exponential in the number of features), and many of these vectors will be missing or sparsely found in the training data. Thus, estimating the a posteriori probabilities can be quite difficult. To overcome these problems, we make a simplifying assumption which leads to the Naive Bayes classifier: all the features are considered conditionally independent, given the category. With this simplifying assumption, we can rewrite the above equation:

$$P(m = c | \vec{X} = \vec{x}) = \frac{P(m=c) \cdot \prod_{i=1}^n P(X_i = x_i | m=c)}{\sum_{c' \in \{S, L\}} P(m=c') \cdot \prod_{i=1}^n P(X_i = x_i | m=c')}.$$

The new equation requires the estimation of a much smaller number of conditional probabilities (i.e., linear in the number of features), making their estimation feasible using the training data. Although the independence assumption is over simplistic, studies in several domains (including spam filtering) have shown Naive Bayes to be an effective classifier.

The probabilities required in the above equation can be calculated as follows:

$$P(m = c) = \frac{N_c}{N_{Training}}$$

and

$$P(X_i = x_i | m = c) = \frac{N_{c \wedge X_i = x_i}}{N_{Training}}.$$

In the above equations, N_c is the number of messages of type c , $N_{c \wedge X_i = x_i}$ is the number of messages of type c with x_i as its i^{th} feature’s value, and $N_{Training}$ is the total number of training messages. These probabilities, along with the above equation, constitute the Naive Bayes classifier.

Support Vector Machine: Support Vector Machine (SVM) is a powerful machine learning technique based on the structured risk minimization principle from Computational Learning Theory. Given a set of messages, represented as feature vectors x_i , the simplest version of the Support Vector Machine learner tries to find a hyperplane in the feature space of these vectors which best separates the two different kinds of messages. More formally, training a Support Vector Machine is equivalent to solving the following optimization problem:

$$\min_{w, b, \xi_i} \left(\frac{1}{2} W^T \cdot W + C \sum_{i=1}^N \xi_i \right)$$

subjected to

$$y_i (W^T \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0.$$

In the above equation, W is a weight vector that should be minimized in order to find an optimal linear separating hyperplane in the feature space, and ξ_i are slack variables which are used together with $C \geq 0$ to find a solution to the above equation in the non-separable cases.

The equation above is representative of the simplest class of SVMs. In general, the feature vectors can be mapped into a higher dimensional space using a non-linear kernel function, and the best separating hyperplane can be found in this higher dimensional space. However, research in text classification [18] has shown that simple linear SVMs usually perform as well as non-linear ones.

LogitBoost with Regression Stumps: LogitBoost belongs to the class of Boosting Classifiers which are classification algorithms that try to construct a good classifier by repeated training of a weak classifier. Boosting uses the weak learning procedure to construct a sequence of classifiers f_1, f_2, \dots, f_k , and then, it uses a weighted vote among these classifiers to make a prediction.

The number of classifiers learned by Boosting is equal to the number of iterations for which it is run. Boosting associates a weight with each training instance in each iteration. Initially, all the instances are assigned equal weights, and during each iteration, the weights of the instances are updated so that the weak learner learns a different classification function in each iteration. Intuitively, those training instances which are misclassified by all of the previous iterations are assigned the highest weights in the i^{th} round, forcing the i^{th} classifier to concentrate on those instances in the i^{th} round.

The equations involved in building the LogitBoost classifier are rather involved and beyond the scope of this paper; the interested reader is referred to [12] for a thorough description.