IBM SpamGuru on the TREC 2005 Spam Track

Richard Segal RSEGAL@US.IBM.COM
IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA

Abstract

IBM Research is developing an enterprise-class anti-spam filter as part of our overall strategy of attacking the Spam problem on multiple fronts. Our anti-spam filter, SpamGuru, mirrors this philosophy by incorporating several different filtering technologies and intelligently combining their output to produce a single spamminess rating. The use of multiple algorithms improves the system's effectiveness and makes it more difficult for spammers to attack. While our overall performance was strong, our results did uncover some flaws and weaknesses in our existing implementation. Our latest code, with these weaknesses addressed as well as other enhancements, produces results on par with the best performing classifiers reported for TREC 2005 on the public corpus.

1. Introduction

IBM Research is developing an enterprise-class anti-spam filter as part of our overall strategy of attacking the Spam problem on multiple fronts. Our anti-spam filter, SpamGuru, mirrors this philosophy by incorporating several different filtering technologies and intelligently combining their output to produce a single spamminess rating or score for each incoming message. The use of multiple algorithms improves the system's effectiveness and makes it more difficult for spammers to attack. While a spammer may defeat any single algorithm, SpamGuru can rely on its remaining algorithms to maintain a high-degree of effectiveness.

The IBM Research submission to the TREC 2005 spam track used the SpamGuru anti-spam framework to evaluate three experimental anti-spam technologies that are under development. The first is LNB (Less-Naïve Bayes), an extension of the ubiquitous Naïve-Bayesian text classifier that relaxes the independence assumption by modeling some of the dependencies between attributes. The second is SMTP Path Analysis, an algorithm that classifies incoming mail based on the servers used to deliver the message. The third technology is a classifier aggregation algorithm that uses the Nelder-Mead nonlinear optimizer to dynamically select weights for combining the LNB and SMTP path analysis classifiers into a single prediction.

The next section presents an overview of the SpamGuru anti-spam framework. The following sections details each of the three algorithms used in our evaluation. Section 6 presents our results on the TREC 2005 Spam Track, analyzes our performance, and discusses what we have learned that has allowed us to substantially improve our overall system. Finally, Section 7 concludes.

2. SpamGuru Overview

SpamGuru is a server-based framework for anti-spam filtering (Segal et al., 2004). The SpamGuru framework provides the plumbing needed for a complete anti-spam solution. The SpamGuru server can communicate with a variety of SMTP servers and e-mail gateways to provide anti-spam services.

The work of labeling incoming messages is done using the SpamGuru Filtering Pipeline. The filtering pipeline consists of one or more pluggable anti-spam modules. SpamGuru processes each incoming e-mail by passing it from one anti-spam module to the next. Each module analyzes the message and optionally assigns it a score based on its prediction of how likely the message is to be spam. The last module in the chain is always an aggregator that is responsible for combining the results from the individual classifiers into a single score. That score is then used by SpamGuru to decide on whether or not to flag the message as spam.

The SpamGuru Filtering Pipeline was instantiated for TREC 2005 as follows. Our primary TREC submission consisted of a MIME decoder, the LNB classifier, the SMTP Path Analysis classifier, and our optimization-based classifier aggregator. For our other submissions, we submitted each classification technology individually so that we could evaluate each indi-

vidual algorithm as well as our aggregation technology. Our second submission therefore used just the SMTP Path analysis module. Our third submission consisted of the MIME decoder and the LNB classifier. Our second submission does not consider the text in the body of a message and therefore does not benefit from decoding the message's constituent MIME parts.

3. Less Naïve Bayesian

Traditional Naïve Bayesian filters make the assumption that words are conditionally independent given the target classification and use that assumption to derive an otherwise mathematically sound formula for the probability of a document being a member of that class (Lewis, 1998). However, it is manifestly true that this conditional independence assumption does not hold. The word "inkjet" appears much more often in spam documents that also contain the word "printer" than in randomly selected spam documents. The goal of Less Naïve Bayesian (LNB) is to produce a classifier that is more accurate than simple Naïve-Bayes by taking the dependencies among features into account.

The LNB algorithm is not strictly Bayesian. LNB takes the approach of a discriminative classifier in which probabilities are replaced by a single weight that represents a word's relative spamminess. As is typically done with discriminant-style classifiers, the word weights are adjusted as new training examples arrive to ensure that the new training example is categorized correctly. What is unique is that LNB adjusts its weights in a manner that makes the algorithm less sensitive to feature dependencies.

4. SMTP Path Analysis

SMTP Path Analysis categorizes incoming spam based on the sequence of gateways that delivered the message (Leiba et al., 2005). The intuition behind the algorithm is that mail sent through the same or similar IP addresses are likely to share the same classifications.

The SMTP protocol specifies that each SMTP relay used to send an email message must add at the beginning of the message's headers a "received" line that contains (at least) information about the SMTP server receiving the message, from where the server received the message, and a time-stamp stating when the header was added. These header lines, taken together, provide a trace of the SMTP path used to deliver a message.

SMTP Path Analysis learns the spamminess or goodness of IP addresses by analyzing the past history of e-mail sent using that IP address. When training, the algorithm extracts from each message the sequence of IP addresses that mail supposedly took to get to the recipient and collects statistics about the spam and good e-mail sent through each IP address. During classification, the algorithm extracts the IP address sequence from the target message and yields a score for that message based on the IP addresses of the gateways supposedly used to deliver the message. The algorithm looks at no other information; in particular, it does not otherwise analyze the content of the message or consider any domain information.

The probability that mail passing through any previously-seen IP address is spam is estimated, when possible, based on the frequency of spam in e-mail previously sent by that host. However, due to dynamic IP addresses and other complexities of IP addresses, a substantial amount of e-mail originates from IP addresses for which we may have little to no data. We address this issue by combining statistics of the current IP address with those of "nearby" IP addresses whenever there is not sufficient data regarding the current IP address to make a reliable decision.

As described, SMTP Path Analysis is susceptible to spoofing. A spammer can easily add false received line headers to a message to make it appear to be sent through a reliable source. To address this problem, we establish a credibility value for each intermediate address, and if an address is not credible we partially ignore the remaining addresses.

5. Classifier Aggregation

SpamGuru employs an aggregate classifier to combine the results of each individual classifier into a single score that can be used to decide how each incoming message should be routed. Classifier aggregation provides two benefits. First, it improves classifier accuracy by combining the best features of multiple algorithms. Second, it improves the robustness of the overall system since a spammer trying to attack the system must defeat multiple anti-spam filtering technologies to defeat the entire system (Segal, 2005).

Each classifier in SpamGuru rates the spamminess of incoming messages by returning a score from 0 to 1000. A score of 0 indicates that the message is almost certainly good, while a score of 1000 indicates that the message is almost certainly spam. The output of most classifiers can be scaled to fit this range.

The scores of several classifiers are combined by computing a single score from the scores of each individual

classifier. There are several well-known methods for combining classifiers (Dietterich, 2000). One option is to return the minimum score output by any of the classifiers. This method tags a message as spam only if all the classifiers return a score over a threshold provided by the user. That is, all the classifiers agree that the message is spam. The minimum score aggregator produces a very low false-positive rate since a legitimate message can only be misclassified as spam if all the algorithms incorrectly label the message as spam. On the other hand, its spam detection rate can be no better than the least effective classifier.

By experimentation, we found that the most successful way to combine classifiers was to use their unthresholded output scores as input to a super-classifier; a linear one typically worked well in practice. The linear super-classifier's score was a weighted sum of the scores of the constituent classifiers. The optimal values of the weights were established by using a Nelder-Mead nonlinear optimizer to minimize a penalty function that emphasized the relative importance of false positives and false negatives in the anti-spam domain. The optimization was performed using a sliding window of several thousand most recently labeled emails.

6. Results

Figure 1 graphs SpamGuru's performance on each of the four TREC datasets. For comparison, each figure includes a classifier marked "Best*" that was constructed using the best results returned by any classifier at several representative false positive rates. Note, the hypothetical "Best*" classifier is likely to outperform all real classifiers as some classifiers perform better in different regions of the ROC curve.

SpamGuru's performed very well on the TM dataset, scoring at or just below the best classifier throughout the entire curve. It's performance was also very good for low misclassification rates on the MrX and SB datasets. However, SpamGuru's ROC curve drops far below the best classifiers for ham misclassification rates above 1%. In real-world use, poor results with a threshold that allows a high misclassification rate is not a problem as most users find false-positive rates above 1% unacceptable. It developing Spamguru, we have focused on achieving good performance with false positive rates at or below the 0.1% level. The strong performance of SpamGuru at a 0.1% false-positive rate and below is the direct result of these efforts.

SpamGuru's weak performance at higher false positive rates suggests that a real issue may be lurking under the covers. With further investigation, we discovered a flaw that caused most of the issue with the MrX dataset. SpamGuru places a limit on the overall message size that it will attempt to classify. This is done to achieve good scalability and to protect against denial of service attacks. Messages above this threshold are assumed ham. The MrX dataset was unusual in that it had several spam messages above SpamGuru's size threshold, a problem we had not seen in during our testing on actual e-mail streams. It is possible that we had not seen this problem previously because SpamGuru is usually deployed in conjunction with anti-virus technology that eliminates most offensive large messages.

Figure 2 shows SpamGuru's performance when long messages are truncated to meet its internal size limit. The improvement on this dataset is dramatic. SpamGuru performs close to or above the best reported performance through most of its ROC curve. Performance still does drop starting around 1% ham misclassification rate, but the drop is not nearly as significant as it was when SpamGuru assumed all large e-mail is legitimate.

The graphs in Figure 1 also illustrate the performance of our aggregation technology and the constituent classifiers that were combined. Most of the system's accuracy comes from the performance of Less Naïve Bayes. As seen in the graphs, SMTP path analysis cannot provide the same level of spam detection provided by the best Bayesian filters. In our experience, the main value of SMTP path analysis is as an adjunct classifier that can improve performance when combined with a more powerful classifier. SMTP path analysis works well in this role as its predictions are based on structural data that are not available to the commonly used bag-of-words style anti-spam filters.

The value of SMTP path analysis and SpamGuru's aggregator can be seen on the Full and TM datasets. On both these datasets, the aggregator substantial improves SpamGuru's ROC curve. Interestingly, the aggregator provides this advantage on the full dataset despite a very poor showing of SMTP path analysis on the portions of the ROC curve in which the aggregator helped most.

While we did not expect SMTP Path Analysis to be competitive with the best spam filters, its accuracy was still considerably below our expectations. We ran several additional experiments to understand these results. We discovered that there was a bug in the parser that caused many IP addresses to be missed. Since our original submission, we have fixed this bug as well as made several improvements to Less Naïve Bayesian. Figure reffig:full-new shows the performance of the

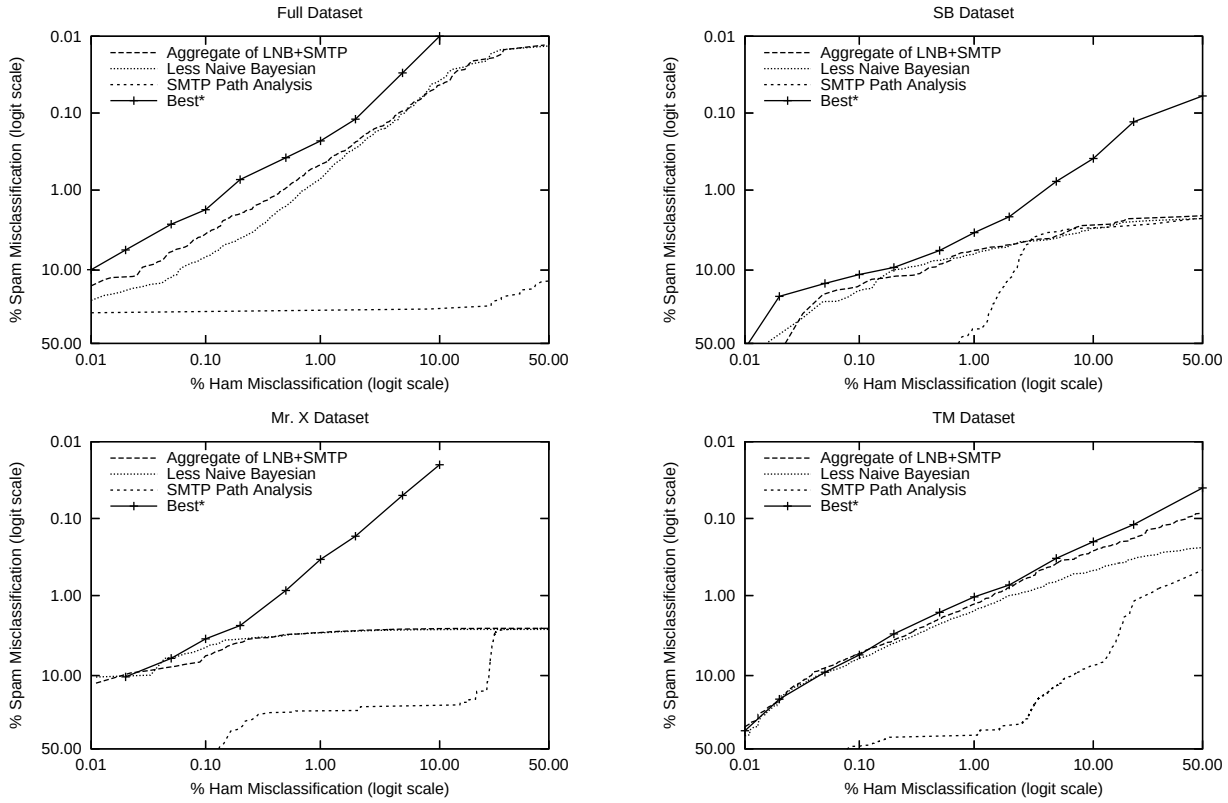

Figure 1. Performance of SpamGuru on the TREC datasets.

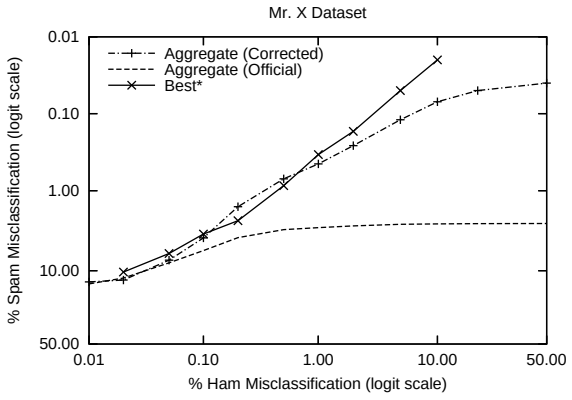

Figure 2. SpamGuru results with large message flaw fix.

latest version of SpamGuru on the full dataset. The performance of SMTP path analysis dramatically improved as the result of the improved parsing. The new results are more in line with our expectations, SMTP path analysis works well but its performance is not sufficient for it to function as a standalone classifier.

These results also show a substantial improvement in Less Naïve Bayes. The new version of Less Naïve Bayes performs very close to the "Best*" classifier throughout its ROC curve. This new version has several upgrades, most notably an improved method for smoothing probabilities and better handling of pure words — words that only appear in spam or only appear in ham. However, its strong performance provides the aggregator with very little room for improvement, even with a much improved SMTP Path Analysis algorithm.

The final experiment we performed was to evaluate whether our extension to Naïve Bayes provided any real benefit. Figure 4 compares Less Naïve Bayesian to traditional Naïve Bayes with a geometric mean. The first Naïve-Bayesian implementation shown in the graph was created by disabling our extensions in our Less Naïve Bayesian code. As a result, this comparison is free from most biases that can be introduced by

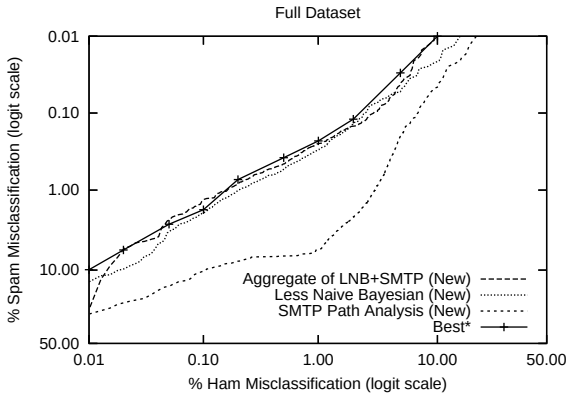

Figure 3. Latest version of SpamGuru on Full dataset.

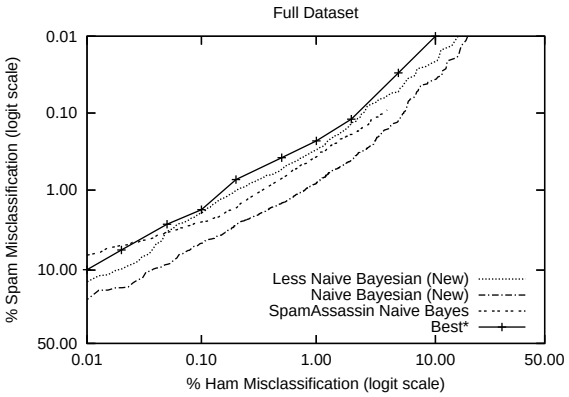

Figure 4. Less Naïve Bayesian vs. Naïve Bayesian.

implementation details such as choice of tokenization algorithm or choice of smoothing function. The second Naïve-Bayesian implementation used for comparison is the one distributed with the popular SpamAssassin anti-spam filter. This second implementation serves as a control to ensure that our implementation of Naïve Bayes is reasonable.

The results of this experiment demonstrate that Less Naïve Bayesian does indeed offer a substantial improvement over our implementation of Naïve Bayes on this data. Less Naïve Bayes outperformed our Naïve Bayes implementation by a large margin throughout the entire ROC curve. It also outperformed the SpamAssassin Naïve-Bayes implementation over most of the curve, but by a much smaller margin. However, the results are not clear cut as the SpamAssassin implementation of Naïve Bayes performed much better than our Naïve Bayes version. The advantages demonstrated by LNB in this experiment may be due to weaknesses in our Naïve Bayes implementation. As future work, we would like to explore why our basic Bayesian implementation did not perform as well as SpamAssassin's implementation. This may provide new insight into the implementation details that are important for building accurate Bayesian classifiers. It also may provide a strong starting point for our Less Naïve Bayesian algorithm and allow us to achieve even higher performance levels.

7. Conclusion

The SpamGuru entry to the TREC 2005 spam track performed very well. SpamGuru's results were consistently near the best when evaluated on low ham misclassification rates. We finished with the fourth lowest spam misclassification rate at the 0.1% ham misclassification level (Cormack & Lynam, 2006). Only the three variations of the Josef Stefan Institute's compression-based classifier performed better on this metric.

In previous work, we have evaluated our system only on ham misclassification rates at or below 0.1% as this is the region of greatest importance for end users. Participating in TREC has taught us that considering ham misclassification rates above 1% can be useful in testing and debugging. While most anti-spam systems will never be used in production with such high false positive rates, the weaknesses show by SpamGuru in this region of the ROC curve have helped us uncover several flaws in our submission. When these errors were corrected and combined with some recent enhancements to our Less Naïve Bayesian classifier, SpamGuru achieves very close to best-in-class performance on the Full dataset for most of the ROC curve.

Acknowledgments

The authors want to thank the many in IBM who have helped in the development of SpamGuru and the development of many of the ideas presented in this paper. Those involved include Bill Arnold, Nathaniel Borenstein, Jason Crawford, Mike Halliday, Shlomo Hershkop, Tien Huynh, Barry Leiba, Jeff Kephart, Joel Ossher, V. T. Rajan, Isidore Rigoutsos, Mark Wegman, and Ian Whalley. We also give special thanks to the track organizers for all their hard work in putting together this workshop and their patience in addressing some technical difficulties that arose with our initial submission.

References

Cormack, G., & Lynam, T. (2006). TREC 2005 Spam Track overview. *The Fourteenth Text REtrieval Conference (TREC 2005) Notebook*.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems* (pp. 1–15). Springer-Verlag.

Leiba, B., Ossher, J., Rajan, V., Segal, R., & Wegman, M. (2005). SMTP path analysis. *Proceedings of the Second Conference on Email and Anti-Spam*.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Proceedings of ECML-98, 10th European Conference on Machine Learning*.

Segal, R. (2005). Combining multiple classifiers. *Virus Bulletin*.

Segal, R. B., Crawford, J., Kephart, J. O., & Leiba, B. (2004). SpamGuru: An enterprise anti-spam filtering system. *Proceedings of the First Conference on Email and Anti-Spam*.