Exploring Support Vector Machines and Random Forests for Spam Detection

Gordon Rios[1] and Hongyuan Zha[2]

[1] Proofpoint Inc., Cupertino, CA 95014, USA
[2] Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA

Abstract. In this paper, we experiment with support vector machines and random forests, two of the state-of-the-art classification algorithms, for spam detection. We use public and private email corpora from a wide community of internet users collected over several years for our comparative studies. In our empirical studies, we emphasize the importance of feature engineering and the effective handling of the dynamic nature of spam emails by using small sets of highly skewed time-indexed data.

1 Introduction

Over the past several years both the open source and commercial software communities have responded with a host of solutions for the email spam problem. Much of the discussion of spam detection is focused on building filters for individual end users rather than the larger commercial problem of blocking spam for users within organizations. For individual filters people often talk about false positive (FP) rates of two to three percent. However, for a commercial system with tens or hundreds of thousands of mailboxes false positive rates over one percent would result in thousands of distressed users. Typically, a large organizational email system is usually administered by a few engineers who simply cannot handle large volumes of support calls.

Due to low accuracy of the spam classification systems we see two basic strategies to protecting organizations: ultra conservative strategies marked by low effectiveness and very low FP rates; and moderately effective ones with uncertain or unmonitored FP rates but some limited form of "tunability". The first approach often employs signatures or message hashing databases that recognize prior spam messages and block those while leaving all other messages untouched. With the explosion of spam volume and sophistication over the past two years the conservative, or reactive strategies have seen drastic drops in effectiveness. The most popular example of the second approach is the well known open source solution called SpamAssassin [1] which provides facilities for training, blocklisting and safelisting mechanisms, and rule extensibility. SpamAssassin has a great deal of support, however it is also widely known to require a tremendous amount of email administrator maintenance efforts.

Widespread attempts to remedy the issues facing spam detection for organizations resulted in introduction of machine learning methodologies to the practice of spam detection. In August of 2002 a well known anti-spam personality named Paul Graham demonstrated the application of Naive Bayesian classification to the spam problem [2]. In the machine learning community the application of Naive Bayes to spam classification was explored in detail several years earlier in [3]. Nevertheless, Graham's introduction has spawned a host of Naive Bayes related products and the introduction of this technology into many preexisting solutions. However, the myriad shortcomings of using Naive Bayes have been highlighted in the popular technology press in a storm of articles and talks most notably by Paul Graham himself. Part of the issue is certainly due to the severe violations of *conditional independence* found in commonly available feature sets.[3]

The question seems to be which machine learning technique will replace Naive Bayes as the workhorse of spam classification. In this paper, we carefully explore and contrast the properties of two of the state of the art machine learning techniques: support vector machines and random forests. To provide context within the still widespread application of Naive Bayes we conduct our comparisons against this technique as well. The

[3] SpamAssassin generates examples of fully dependent features such as HTML_MESSAGE and HTML_60_70 where HTML_60_70 can only fire if HTML_MESSAGE fires, etc.

emphasis of our paper is to treat spam classification as a dynamic process with drifting and time-varying characteristics rather than as a static text classification problem which seems to be the approach taken by much of the previous research. We also point out the challenges posed by the skewed nature of the spam distribution.

2 The Support Vector Classifier

Support vector machine (SVM) has become very popular in the machine learning and data mining community due to its good generalization performance and its ability to handle high-dimensional data through the use of kernels. Consider a set of training data points $T = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x_i \in \mathcal{R}^n$ and $y_i \in \{+1, -1\}$. SVM seeks to find a hyperplane

$$\{x \mid w^T x + b = 0\}$$

that generates the *largest* margin between the data points in the positive class and those in the negative class. It can be shown that this can be achieved through the following minimization problem [8],

$$\min \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

subject to the linear inequality constraints

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, N,$$

where ξ_i is the so-called slack variable, that allows misclassification to happen, and C is a parameter that balances the amount of misclassification and the size of the margin. The above minimization problem is usually solved through its dual formulation

$$\min \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i$$

subject to the linear constraints

$$\sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N,$$

where we have replaced the dot-product in $\mathcal{R}^n$ by a general kernel function K. The decision function is given by

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i K(x, x_i) + b\right).$$

SVM with linear kernels has proven to be quite successful for tasks such as document classification where one needs to handle many features (words or phrases) and it is also the type of kernel we use in our experiments [5]. Since we treat spam detection as a *binary* classification problem, there is no need to explore some of the ad-hoc ways of generating a multi-class classifier from a set of binary classifiers. We used SVM Light for our experiment, and the parameters were chosen so as to maximize the estimated performance [6, 7].

3 Random Forests

Random forests grow many classification trees $\{h(x, \Theta_k)\}$ where each tree in the forest depends on the value of some random vector Θ. For each step k, Θ_k is chosen independent of $\Theta_1, \ldots, \Theta_{k-1}$. To classify a new data point with a given feature vector, put the feature vector down each of the trees in the forest. Each tree gives a classification, and we say the tree *votes* for that class. The forest chooses the classification having the

most votes (over all the trees in the forest). Many versions of random forests exist depending on how Θ is generated, examples include bagging and random split selection [4]. We used a version described in [4] based on random selection of features at each node.

Breiman has shown that the generalization error of a random forest depends on the following two factors: 1) The correlation between any two trees in the forest. Increasing the correlation increases the forest generalization error rate. 2) The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate. Reducing m, the number of random features to select at each node, tends to reduce both the correlation and the strength while increasing it increases both. But we have found that for a range of m the generalization error is quite stable. For our spam detection tests, we have used the R implementation of Breiman's random features. We have also experimented with variations of the number of tree to grow and found that the performance is not very sensitive to it, and have settled on using 500 trees in our experiments which seem to be quite adequate for the size of the data we have.

4 Feature Engineering for Spam Detection

A critical component for spam detection solutions is feature engineering. A tremendous amount of domain expertise is encoded in the rule sets made publicly available on the SpamAssassin website. SpamAssassin features fall roughly into three areas:

1. Header: features about the meta information of a message including date, subject, Mail Transfer Agent (MTA), Mail User Agent (MUA), content types, etc.
2. Body and raw body text: structures, markup, words and phrases, known obfuscations, etc.
3. Meta: boolean combinations of other rules – often used to improve the precision of the rules thus lowering the FP rate.

Ongoing feature engineering is enabled by SpamAssassin's extensibility through Perl regular expressions. In fact, constant feature engineering is required to prevent decay of effectiveness in SpamAssassin and other heuristic classifiers. Spammers increasingly react to known rules by avoiding techniques that fire highly weighted rules and spoofing negative evidence rules. In our work looking at SpamAssassin performance on a very wide range of enterprise spam we've seen all out assaults take place on SpamAssassin rules that result in ten to fifteen percent drops in effectiveness in a single day.

For features based on terms and phrases feature engineering often requires introducing obfuscated versions of positive evidence terms like *Viagra* which have substantial numbers of human parsable obfuscations such as *Via@gr@* or *V—ag.r.a.* Some of the more sophisticated spammers vary their obfuscations over large sets of mailings (it is not as if they can "run out" of them) so that naive Bayesian filters attempting to learn automatically will over-fit on highly sparse features generated for "one off" applications.

5 Experimental Results With Time-indexed Data

In exploring the properties of support vector machines and random forests we have extracted a set of 60,000 spam messages and 15,000 valid messages from large email corpora collected over the past six years.[4] The purpose of these experiments is to make reasonably robust estimates of how well the techniques can be expected to maintain effectiveness, at relatively low FP rates.[5] We've noticed that most of the existing evaluations of spam detection algorithm completely ignore the time issue, randomly selecting training and testing sets from a email corpus. In contrast, we were interested in lookahead or generalization accuracy. For each value of the lookahead parameter $k, k = 1, \ldots, K$ we computed statistics for the models trained on set

[4] The email was collected from a range of public sources like www.spamarchive.org and private corpora representing a wide community of email users. Also contained in the 75,000 messages used are approximately 3,000 messages representing the sum total of reported errors from a popular commercial enterprise spam filtering product.

[5] Enterprise clients of spam filtering products balk at FP rates $> 0.25\%$.

i on holdout sets $i+k, \quad i=1,\ldots,N-k$ where we used $K=5$ and $N=30$ for our experiments yielding a sample size of 25.

We looked at both median and pessimistic estimates over thirty time-indexed sets[6] each composed of 2,000 spam and 500 valid messages. We use skewed sets because in practice it is far easier to collect spam than valid email[7] and thus all industrial training sets can be expected to be highly biased towards spam.[8]

We ran the same set of tests for a very high dimensional proprietary set of term and phrase features for SVM and Naive Bayes. We found broader effectiveness but slightly faster decay rates than for models that combined both types of features. However, for the purposes of this paper and to maximize reproducibility all results presented here were achieved using rules extracted using the 2.63 release of the SpamAssassin spam classifier (the most recent release as of this writing.)

5.1 Median True Positive (TP) Rates

In this section we present the performance of each technique using Receiver Operating Characteristic (ROC) curves of the median TP rate for various numbers of lookahead periods.[9] This is a straightforward assessment of spam filtering effectiveness presented for one (left panel of Figure 1), three (right panel of Figure 1), and five (left panel of Figure 2) lookahead periods. Careful examination of Figure 1 shows that the median effectiveness for RF is a solid 5% higher than the SVM classifier. The right panel of Figure 1 shows that RF and SVM are much closer for low FP rates and all three methods converge for FP rates between 1.2% and 1.5%. Interestingly enough at five periods ahead SVM clearly dominates at FP rates below 0.8% suggesting that these models have more durable generalization accuracy.

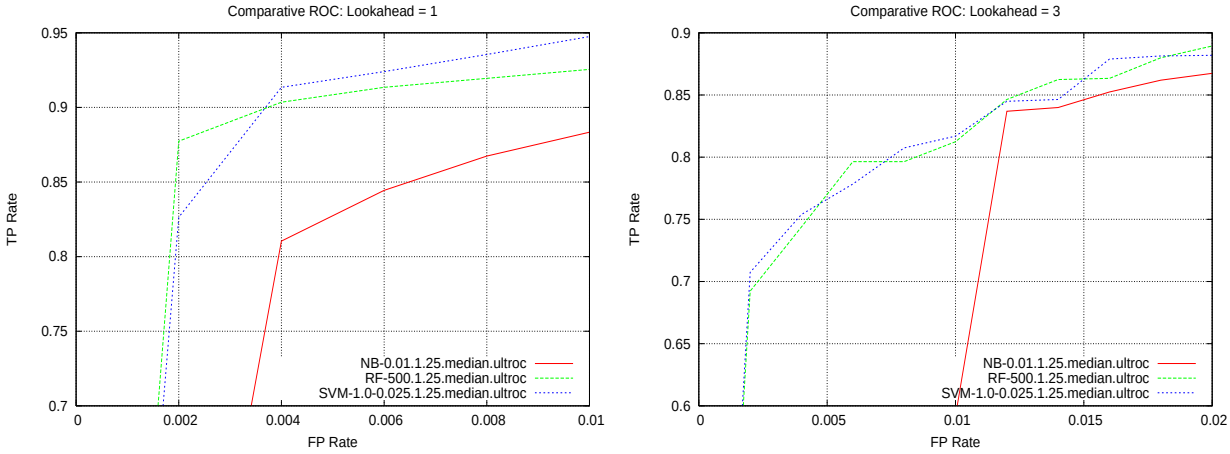

Fig. 1. Median TP Rate: (left) One Period Ahead, (right) Three Periods Ahead

[6] Time indexing is based on the date in the *Received header* which is only missing for cases where valid mail was sent and received within the same organization. In cases of valid mail without a *Received header* because it was internal to an organization we substituted the Date header.

[7] Especially when the raw source messages must be collected – for users on *Exchange(tm)* and *Outlook(tm)* systems it is frequently difficult to get the complete source for a message.

[8] Of course one could just sample from the spam messages to achieve more balanced sets but we have found that the loss of exposure to different combinations of features, text obfuscations, and other novel spam tricks, significantly damages effectiveness.

[9] We computed the TP rate for each discrete level FP rate (since all the sets had the same number of true negatives) and computed the median of these rates.

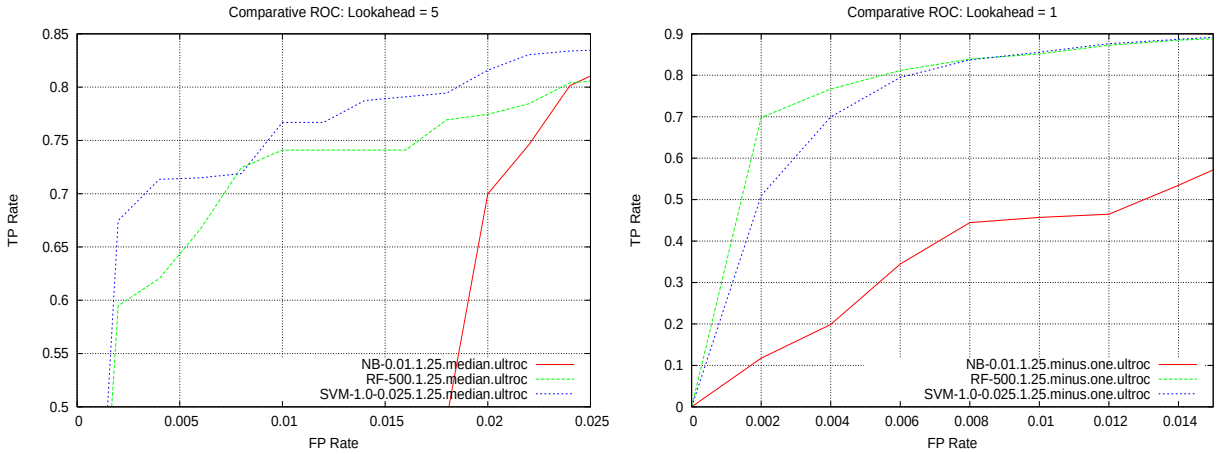

Fig. 2. (left) Median TP Rate: Five Periods Ahead, (right) Minus One Mad TP Rate: One Period Ahead

5.2 Pessimistic True Positive (TP) Rates

It is not enough to look at expected performance (with even robust estimates) because we want to characterize the variation of the model's accuracy. Next we looked at pessimistic estimates of the TP rate to better understand variation of accuracy. Our first pessimistic estimate is present the median minus one Mean Absolute Deviation (MAD) across the modeling techniques.[10] From the right panel of Figure 2 we see that the RF models have almost 20% better effectiveness than the nearest competitor at 0.2% FP rate. The left panel of Figure 3 shows that SVMs close about half of the gap seen for the single period lookahead. The most extreme pessimistic measure, namely the minimum over the 25 test sets, is shown for one and three periods of lookahead in the right panel of Figure 3 and Figure 4 respectively. The minimum TP rates for one lookahead period shows total superiority of the RF models over the entire useful range of FP rates. For the three periods of lookahead RF maintains a significant advantage over the smaller but more important range of FP rates up to 0.6%.

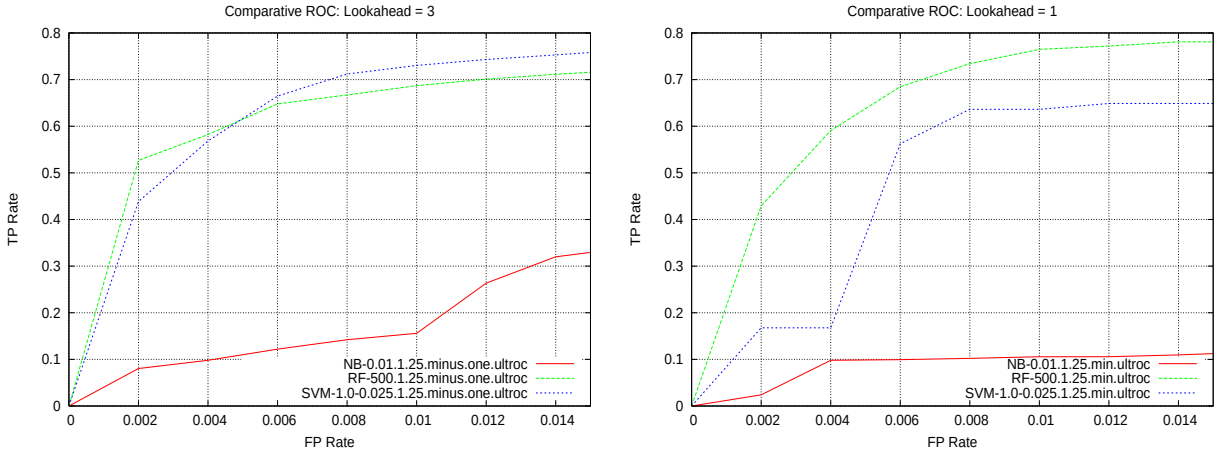

Fig. 3. ((left) Minus One Mad TP Rate: Three Periods Ahead, (right) Minimum TP Rate: One Period Ahead

[10] Technically, we measure the mean of the negative deviation.

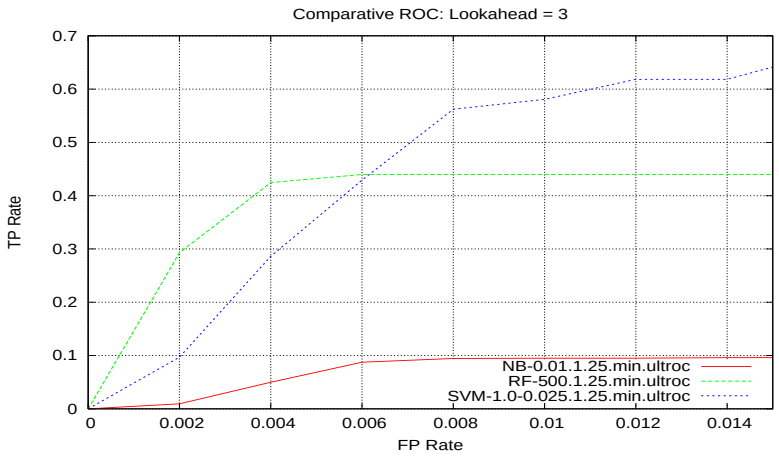

Fig. 4. Minimum TP Rate: Three Period Ahead

6 Conclusions

The performance of the SVM and RF models are comparable across intermediate levels of lookahead testing over the lower FP rates required for enterprise spam filtering. However, for single period lookahead the RF models show clear superiority in both expected accuracy and pessimistic measures of expected accuracy. Both SVM and RF show superior median results at low FP rates relative to Naive Bayes though it does reasonably well at high FP rates. Both RF and SVM demonstrate significantly better pessimistic estimates than the Naive Bayes models at all levels of lookahead. Overall the results are gratifying in the sense that the ensemble based technique (RF) showed itself to be more robust at low FP rates.

Acknowledgment

We thank Luke Lu from Proofpoint Inc. for his help in generating and explaining the details of the SpamAssassin feature set used in the experiments. We also thank John O'Neill from Proofpoint Inc. for generating the time indexed data. Of course, a big thanks to the open source software community for producing and maintaining such invaluable tools as R, XEmacs, and Python. Hongyuan Zha's work was also supported in part by NSF grant DMS-0311800.

References

1. SpamAssassin *SpamAssassin(tm).* `www.spamassassin.org`.
2. Paul Graham *A Plan for Spam.* `www.paulgraham.com/spam.html`.
3. M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI, 1998.
4. L. Breiman. Random Forests. *Machine Learning*, 45:2–32, 2001.
5. T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer, 2002.
6. T. Joachims. SVM Light. `svmlight.joachims.org`.
7. T. Joachims. Estimating the Generalization Performance of a SVM Efficiently. *Proceedings of the International Conference on Machine Learning*, Morgan Kaufman, 2000.
8. V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.