

An Evaluation of Naïve Bayesian Anti-Spam Filtering Techniques

Vikas P. Deshpande, Robert F. Erbacher, and Chris Harris

Abstract— An efficient anti-spam filter that would block all spam, without blocking any legitimate messages is a growing need. To address this problem, we examine the effectiveness of statistically-based approaches Naïve Bayesian anti-spam filters, as it is content-based and self-learning (adaptive) in nature. Additionally, we designed a derivative filter based on relative numbers of tokens. We train the filters using a large corpus of legitimate messages and spam and we test the filter using new incoming personal messages. More specifically, four filtering techniques available for a Naïve Bayesian filter are evaluated. We look at the effectiveness of the technique, and we evaluate different threshold values in order to find an optimal anti-spam filter configuration. Based on cost-sensitive measures, we conclude that additional safety precautions are needed for a Bayesian anti-spam filter to be put into practice. However, our technique can make a positive contribution as a first pass filter.

Index Terms—Spam filter, Naïve Bayesian, Evaluation

I. INTRODUCTION

SPAM continues to be a growing problem accounting for upwards of 90% of all e-mail today [15]. While spam filters have become more effective [16] and wide spread, many spam messages continue to be delivered to end users. The difficulty in eliminating spam lies in differentiating it from a legitimate message. However, the message content of spam typically forms a distinct category rarely observed in legitimate messages, making it possible for text classifiers to be used for anti-spam filtering. The goal of this research is to examine the effectiveness of Naïve Bayesian anti-spam filters and the effect of parameter settings on the effectiveness of spam filtering. Additionally, we look at a novel modification to existing filters and incorporate it into the evaluation.

A Naïve Bayesian anti-spam filter is a text categorization technique based on a machine-learning algorithm. Proposed by Sahami et al. [10], the text categorization technique shows some impressive results on new unseen incoming messages. The filter requires training that can be provided by a previous set of spam and legitimate messages. It keeps track of each

Vikas P. Deshpande recently completed his master's degree in computer science at Utah State University. He can be contacted at vikas@cc.usu.edu.

Robert F. Erbacher is an assistant professor in the Computer Science Department at Utah State University. He can be contacted at Robert.Erbacher@usu.edu

Chris Harris is a master's student in the Computer Science Department at Utah State University. He can be contacted at cwharris@cc.usu.edu.

word that occurs only in spam, only in legitimate messages, and in both. Based on these word occurrence statistics (also called tokens), incoming unseen messages are processed and classified accordingly.

II. BAYESIAN CLASSIFIERS

A. Bayesian Classifier

A Bayesian classifier is the application of a Bayesian network to the process of text classification. Bayesian networks are probabilistic networks that are used as problem solving models in different fields of work.

In our case, a Bayesian network is used to represent a probability distribution of specified text contained in a spam email. In such a graph, a node represents a random variable, and a directed edge indicates a probabilistic dependency from the variable denoted by the parent node to that of the child. Hence, it is implied that any node in the network is conditionally independent of its non-descendants, given its parents. Each node is associated with a conditional probability table that indicates the distribution over that node with any possible assignment of values to its parents [10, 13].

We formulate the Bayesian network to solve our classification problem. Let C be the class variable that indicates to which class (legitimate / spam) a message belongs, and let node X_i denote any attribute (token, in our case) in the message. For our purposes we will say ck is the given of the specific values for the required attributes. The specific values would be 0 or 1 depending on their presence in the message. The problem of class nature can be solved using Baye's theorem:

$$P(C = ck | X = x) = \frac{P(X = x | C = ck)P(C = ck)}{P(X = x)}$$

The probability $P(X = x | C = ck)$ is difficult to calculate, as there is a high chance that the X attribute might be dependent on some other set of attributes. One way to overcome this difficulty is to assume independence of the attributes. This is the basic assumption of the Naïve Bayesian filter.

B. Naïve Bayesian Filter

A Naïve Bayesian model is the most restrictive form of the feature dependence spectrum. Research has been done regarding the performance of spam filters by allowing some degree of dependence between features. This study can be

formalized by introducing the notion of k-dependence Bayesian classifiers. A k-dependence Bayesian classifier is a Bayesian network wherein each feature is allowed to have a maximum of k parents. Based on this definition, we can say that a Naïve Bayesian filter is a 0-dependence Bayesian classifier. We can also state that an ideal Bayesian filter (i.e. full Bayesian filter with no independence) is an (N-1)-dependence Bayesian classifier where N is the number of domain features.

By varying the value of k, one can move step-by-step in the feature dependence spectrum and analyze the performance of the spam filter at every step. It is also worth noting that as k grows, there are more condition variables with the same amount of data. This implies a larger probability space for estimation with the same data, causing inaccuracy in probability estimates and leading to an overall decrease in performance. This performance problem has been observed in many domains while going from k=2 to k=3.

Using a Bayesian network, we can model the complex dependencies between features to infer the solution class. As the number of features increases, it becomes increasingly difficult for a message to be classified with all its dependencies. As a result, spam filters implement a Naïve Bayesian concept wherein features are assumed to be independent of each other. If we assume the attributes are conditionally independent of each other, the probability will result in:

$$P(X = x | C = ck) = \prod_i P(X_i = x_i | C = ck)$$

$$P(C = ck | X = x) = \frac{\prod_i P(X_i = x_i | C = ck) P(C = ck)}{P(X = x)}$$

If an email message is considered to be a set of attributes (i.e., words), then using a Bayesian network, we can calculate the probability of whether a message belongs to a specific class, namely, a legitimate message or a spam.

III. EXPERIMENTS

Our experiment comprises of two phases: the training phase and the classification phase. In the training phase, the filter is trained using a known corpus of spam and good emails. A database of tokens appearing in each corpus and their total occurrences are maintained in a database. Based on their occurrences in each set of spam and good emails, each token is assigned a probability for its capacity of determining an email to be spam given its presence. Then, using this knowledge of tokens, the filter classifies every new incoming email in the classification phase. Once the status of a new email is confirmed, all its tokens are also recorded, thus updating the database. This self-learning function of our filter makes it unique among the other available spam filters. Even if the filter misclassifies any message, the user can rectify it, and the spam filter would update its database accordingly. Thus, the filter learns from its mistakes, too.

We used 1250 legitimate messages and 11350 spam messages. Spam messages were collected from an archive provided by Nik Martin, available at the site hosted by Paul Graham (www.paulgraham.com) [6]. Since these are selections from real messages and the selection of the training set is random we expect the results described here to be representative of different training and testing data sets. The proportion of spam to legitimate messages is quite large, making it more likely that legitimate messages can easily be misclassified as spam. This makes the situation more challenging, as the cost of false positives is much higher than that of false negatives. We feel that by minimizing the false positives in such a situation, we have achieved an efficient Bayesian spam filter. Moreover, by recording tokens from such a huge quantity of spam, we have covered almost all the topics for spam and are in a pretty good position to classify new incoming emails for spam. Additionally, this is more realistic of real-world scenarios given typical proportions of e-mail.

Each word in each email message is considered to be a token. The whole message, including the header, is parsed for tokens. The token separator is a blank space. Words quoted in double and single quotes, numbers, and all words separated by blank spaces are also considered as tokens. The number of tokens under study and used for classification is 90930. Since we are not using any type of lemmatizer, we consider different forms of the same word as different tokens. For example, run, running and runner are all considered as different tokens even though they stem from the single word "run." There are studies [7] that prove the positive effect of a lemmatizer on a filter's performance. The implementation of a lemmatizer is one of the topics of our future study.

Only the message content is used for classification purposes. Doing so eliminates the interference of tokens present in headers in determining the status of a message. In this way, there is no bias among the classification techniques that are considered for evaluation as some techniques consider only a few (or percentage of) tokens for assigning a final score to the message.

Our evaluation was conducted in the classification phase. We evaluated four effective filtering techniques of the Bayesian spam filter for their classification performance. We evaluated these techniques using cost-sensitive measures, as we believe that the cost of a false positive is much higher than that of a false negative. Eighty new incoming messages were tested in a batch of two (first batch: 50; second batch: 30) to get the significant evaluation results. These tested messages belong to the same email account previously used in the training phase. The effective configuration of each technique was used for evaluation purposes. In the standard deviation technique, the value of standard deviation was set to 0.4, and in the percentage technique, 30 percent of total tokens were used to calculate the final score. The tabulated results and related plotted data are explained in the results section.

A. Description of Four Filters Tested

Once the Naïve Bayesian filter is trained using huge datasets of spam and non-spam messages, it is now ready to perform its basic functionality of filtering, i.e. classifying new incoming unseen messages. Currently, there are many classification techniques used with Naïve Bayesian filters available on the market. We discuss four significant techniques in detail that we tested.

1) All Tokens Filter

This technique demands use of all tokens from a new email for classification. As each token is associated with a probability that determines the chances of the email being a spam, tokens from each new email would be used to calculate a combined probability to assign a final score to the email. In the case of a new token in an email (i.e. with no record in the database), it would be assigned a probability of 0.4. This assumption has been practically implemented and been found successful in Naïve Bayesian filters. It implies that a new token is considered to be a good token rather than a part of a spam. It also indicates the positive approach adopted by spam filters, since the cost of a false positive is much higher than that of a false negative. However, we turn off this feature for the purposes of our evaluation because we do not want to favor one technique (by taking a positive approach) over others. This global technique makes sense as we parsed all tokens from training datasets to build a database to be used for classification. Hence, it is logical to use the same technique for classification. It should be noted that the classification phase is critical due to the heavy cost of a false positive as compared to the training phase wherein we know exactly whether an email is a spam or not. This technique might be deceived by an email in which there is a big story of how a person got rich instantly followed by a link to a spam site. Such emails would contain a large amount of good tokens as compared to a spam. There is a high possibility that such emails would deceive spam filters only to be categorized as good email. But it is equally true that spammers avoid writing a big story as it is very likely that email readers would rather delete than read a big article from some unknown source. Thus, the use of the all tokens method is found to be effective in practical filters. For example, Bill Yerazunis has used this technique in his Controllable Regex Mutilator (CRM114).

2) Fixed Number of Tokens Filter

The use of a fixed number of tokens, successfully implemented by Paul Graham [5], takes only a fixed number of tokens into consideration from a new email for assigning a final score to it. The number can vary from 15 to 25, but these tokens are assumed to be the most effective in the given email. The effective tokens are the ones with probabilities that deviate the most from 0.5, i.e. it can be a good token or a bad one. The combined probabilities of these tokens assign a final score to the given new email. In this way, the most effective tokens are emphasized for the task. This technique directly targets those words that are found most of the time in either

legitimate emails or spam. As a result, the final score will most probably end up near 1 if the email is a spam or near 0, otherwise. Thus, this technique alleviates the doubt of email classification where the final score ends up near 0.5. This method of effectiveness was proposed by Sahami et al. who calculated its effectiveness with the help of the mathematical formula of mutual information. It is recommended that the same token should not be counted more than once while calculating a final score. This way, the filter makes an unbiased decision with no interference from any specific token even if it occurs multiple times in the message. The number of tokens used (15/20/25) is a personal decision, based on the success of the spam filter on personal emails. If the number of tokens in a new email happens to be less than a fixed number, say 10, then the use of all tokens is the logical back-up technique to be used for classification.

This technique has some advantages over other techniques. First, to avoid the problem of false positives, the threshold value can be raised to any value near to 0.9 from 0.5. Second, in the case of huge emails, the classification would be faster.

3) Standard Deviation Threshold Filter

This technique, like the previous technique, considers only the effective tokens. However, it emphasizes the spam probability of tokens rather than the number of tokens. If a standard deviation threshold (σ_T) is of value x , then all tokens with a spam probability in the range of $0.5-x$ to $0.5+x$ would be discarded. The remaining tokens would be the effective ones used to calculate the combined probability and assign a final score to the new email. BogoFilter, a spam filter that is currently available on the market, has adopted this approach. The value of σ_T can be varied based on the filter's success on one's personal messages. The value found to be most successful is 0.4. Thus, tokens under consideration would be the ones with probabilities less than 0.1 and higher than 0.9.

The specialty of the technique is that it assigns the score to the email independent of its size. Based on the content of an email, there might be only ten effective tokens, or sometimes there may be even more than 100. But for every classification, only effective tokens with probabilities 0.9 and above and 0.1 and lower would be considered. Like the previous technique, the score in this case would be near 1 (if spam) or near to 0, otherwise. Thus, it is less likely that the score would end up near 0.5, giving rise to the possibility of false positives.

The same token should not be considered more than once to avoid the interference from any specific token if it had occurred a few times in the message. The threshold, like the previous technique, can be raised to 0.9 to reduce the possibilities of false positives. The processing time for classification would vary according to the size of the email.

4) Relative Number of Tokens Filter

We developed and evaluated a novel technique along with the existing and well known techniques. The goal with this technique was to examine the impact of considering a relative number of tokens in contrast with the fixed number of tokens

of the previously discussed techniques. Since the Naïve Bayesian filter is trained with the contents of email messages, it is logical to apply the same content-based approach for classification as well. In this technique, we select some percentage (say 30 percent) of effective tokens out of the total tokens of an email message. These tokens will be used to calculate the combined probability and assign a final score to the email message. The percentage value can be tuned, based on the success of the filter on personal email messages.

This approach is the combination of both the above techniques: the use of a fixed number of tokens and the use of a standard deviation. It values both the effectiveness and number of tokens while classifying a message. So, if an email contains 100 tokens, then the 30 most effective tokens among them will be used for classification. There is a high possibility that many of these 30 tokens would fall in the range discarded by the σ_T threshold. In this way, we utilize the advantages of both the above techniques.

As it is a content-based approach, there are chances that the final score of an email might fall near 0.5. To avoid false positives, the threshold value can be raised to a higher value.

B. Experimental Filter

The experimental Bayesian filter is a content-based approach. This attribute gives this approach an advantage over other approaches. Spammers cannot modify the content to deceive the filters, as content is the only reason to send spam at the first place. Content in this case includes headers and the message itself.

First the filter must be trained to work accordingly. A considerable number of good email and spam are required to train the filter. Two tables would be maintained, one each for legitimate email and spam. Let us call them the good table and the bad table, respectively. The good table contains tokens that occur in the good emails, along with their number of occurrences. Similarly, the data for bad emails is maintained in the bad table.

Based on these two tables, we build another table using the **Bayesian formula** of probability [4, 5]:

$$P(\text{bad} | \text{token}) = \frac{P(\text{token} | \text{bad})P(\text{bad})}{P(\text{token} | \text{bad})P(\text{bad}) + P(\text{token} | \text{good})P(\text{good})}$$

where

$P(\text{token} | \text{bad})$ = probability of a token given that it is present in spam email.

$P(\text{token} | \text{good})$ = probability of a token given that it is present in good email.

$P(\text{bad} | \text{token})$ = probability of email being spam given that a specific token is present.

This spam probability table will contain all tokens that occur in all email, along with the probability of the email being spam with that token present. For every new email, a fixed number of effective tokens are collected to calculate the **combined probability**. This number can vary from 3 to 25

depending on the success of the filter for select messages.

$$\text{Score} = \frac{\prod_i P(t_i)}{\prod_i P(t_i) + \prod_i (1 - P(t_i))}$$

If the score rises above a threshold value, the email would be declared as spam, else as a good email. Effective tokens are those whose probabilities differ the most, on either side, from the threshold value. These tokens are either significantly good tokens or bad tokens, and they are responsible for deciding the overall status of the message.

The challenges present are speed, efficiency, database size, and the need of training data. The larger the set of tokens the greater the size of the database and the longer the time for training. So, there is a need to consider only those tokens that make an impact in deciding the status of an email. Since training and email classification will occur during the same phase of time, special care must be taken to make both operations as independent as possible.

If the token is present only in the good table, its probability in the spam probability table would be recorded as 0.1. If the token is present only in the bad table, its probability in the spam probability table would be recorded as 0.9.

C. Cost Evaluation

1) Effect of False Positives

A false positive is mistakenly classifying a legitimate email as a spam, and a false negative is mistakenly classifying a spam as a legitimate email. The cost of a false positive is much higher than that of a false negative. The existence of false positives destroys the faith of the user in their spam filter because email users tend to delete spam from a bulk folder without reading them, and deleting legitimate messages (due to spam filters) is unacceptable. In that case, it is acceptable to allow some false negatives rather than having any false positives.

Let $L \rightarrow S$ be false positive error type and $S \rightarrow L$ be false negative error type. Assuming that $L \rightarrow S$ is λ times costlier than $S \rightarrow L$, we classify a message as spam if:

$$\frac{P(C = \text{spam} | X = x)}{P(C = \text{legitimate} | X = x)} > \lambda$$

In our case wherein we are considering a Naïve Bayesian filter's independency, the assumption holds. Therefore, $P(C=\text{spam} | X=x) = 1 - P(C=\text{legitimate} | X=x)$, which leads to the following criteria:

$P(C=\text{spam} | X=x) > t$, where t = threshold value

Thus $t = \lambda / (1 + \lambda)$ as $\lambda = t / (1 - t)$

Depending on the action taken on a spam folder, the threshold value can be altered. If spam are deleted directly once they are classified, then t is held as high as 0.999 ($\lambda = 999$), i.e. blocking a legitimate message is as bad as letting 999-spam messages pass the filter. Lower values of λ are acceptable depending on the different configurations made available for the spam folder. If the configuration is set up to resend the email back to the sender asking him to send it to a

private unfiltered email address of the recipient, then $\lambda = 9$ ($t=0.9$) seems to be reasonable. Even $\lambda = 1$ ($t=0.5$) is acceptable if the recipient happens to go through every email in the bulk folder before manually deleting them.

Two factors could be used in the context to measure the performance of a filter, namely, spam precision and spam recall. Let $n_{L \rightarrow S}$ and $n_{S \rightarrow L}$ be the numbers of $L \rightarrow S$ and $S \rightarrow L$ errors, and let $n_{L \rightarrow L}$ and $n_{S \rightarrow S}$ count the correctly treated legitimate and spam messages respectively. Spam recall (SR) and spam precision (SP) are defined as follows:

$$SR = \frac{n_{S \rightarrow S}}{n_{S \rightarrow L} + n_{S \rightarrow S}}$$

$$SP = \frac{n_{S \rightarrow S}}{n_{L \rightarrow S} + n_{S \rightarrow S}}$$

2) Total Cost Ratio

The evaluation factors that are frequently used in case of classification are accuracy (Acc) and the error rate (Err = 1 - Acc). Accuracy can be defined as the number of correct classifications, i.e. spam correctly classified as spam and legitimate messages as legitimate out of the total messages. The error rate is the ratio of the sum of false positives and false negatives out of the total messages.

$$Acc = \frac{n_{S \rightarrow S} + n_{L \rightarrow L}}{N_L + N_S} \quad Err = \frac{n_{L \rightarrow S} + n_{S \rightarrow L}}{N_L + N_S}$$

Where N_L and N_S are the number of legitimate and spam messages, respectively.

In our cost-sensitive evaluation, we assume that the error of a false positive is much higher than that of false negative. Conversely, the above formulae of accuracy and error rate do not consider the cost-sensitive factor. Let's assume that the error cost of a false positive is λ times greater than that of a false negative; the implication being that we treat a legitimate message as being worth λ messages. So, if a legitimate message is misclassified, it counts as λ errors, and if it is classified correctly, it counts as λ successes. This assumption can be formulated in the form of a weighted accuracy (W_{Acc}) and a weighted error rate ($W_{Err} = 1 - W_{Acc}$) as:

$$W_{Acc} = \frac{\lambda n_{S \rightarrow S} + n_{L \rightarrow L}}{\lambda N_L + N_S} \quad W_{Err} = \frac{\lambda n_{L \rightarrow S} + n_{S \rightarrow L}}{\lambda N_L + N_S}$$

To get a better idea of the filter's performance in terms of accuracy and error rate, we must compare these factors with a baseline approach [7]. In a baseline approach, we assume that no sort of filter is active, i.e. all spam pass the filter, and legitimate messages are never blocked. The weighted accuracy and error rate of the baseline are:

$$W_{Acc}^b = \frac{\lambda N_L}{\lambda N_L + N_S} \quad W_{Err}^b = \frac{\lambda N_S}{\lambda N_L + N_S}$$

We calculate TCR (Total Cost Ratio) to compare with the baseline approach [7]:

$$TCR = \frac{W_{Err}^b}{W_{Err}} = \frac{N_S}{\lambda n_{L \rightarrow S} + n_{S \rightarrow L}}$$

The higher the value of the Total Cost Ratio, the better the performance. With $TCR < 1$, a baseline approach is a better option, implying that the absence of a filter gives better results than the use of a filter. If cost is relative to wasted time, then TCR measures the time wasted to delete manually all spam messages as compared to the sum of time wasted to delete manually all spam messages misclassified as legitimate ($n_{S \rightarrow L}$) and time wasted by recovering all legitimate messages mistakenly classified as spam ($\lambda n_{L \rightarrow S}$).

IV. RESULTS

A. False Positives and False Negatives

Table 1 lists false positives, false negatives, and correct classifications of all four techniques with different configurations for the threshold. Table 2 lists spam recall, spam precision, weighted accuracy, baseline-weighted accuracy, and total cost ratio (TCR) for the same. TCR is calculated for all techniques for different values of thresholds. In a cost sensitive evaluation, TCR can be used as a scale of better performance. Table 1 indicates the fall in number of false positives and rise in number of false negatives by raising the threshold bars for all four techniques. Table 2 indicates that the fixed token technique outperforms for every value of λ . Unlike other techniques, the fixed token approach gives excellent results for $\lambda=999$. The all token approach is worst among them all. Our percentage approach performs better than the standard deviation for $\lambda = 1$ and $\lambda=999$.

Table 1: The results (false positives, false negatives and correct classifications) of four filtering techniques for different values of λ .

Filter Technique	λ	$n_{L \rightarrow S}$	$n_{S \rightarrow L}$	$n_{L \rightarrow L}$	$n_{S \rightarrow S}$
All Token	1	11	0	14	25
Fixed Token	1	6	0	19	25
Std Deviation	1	9	1	16	24
Percentage	1	9	0	16	25
All Token	9	11	0	14	25
Fixed Token	9	5	0	20	25
Std Deviation	9	6	1	19	24
Percentage	9	9	0	16	25
All Token	999	9	3	16	22
Fixed Token	999	0	5	25	20
Std Deviation	999	4	4	21	21
Percentage	999	3	6	22	19

Based on both the tables, we can say that by lowering the threshold value from 0.999 to 0.5, we have risked an increase in number of false positives. But at the same time, the evaluation has shown an increase in TCR values, indicating that an increase in false positives does not prove costly. However, in practice, no user would use a threshold of 0.5 as

it implies that the user has to go through every spam email before deleting it.

The filter would just be helping the user in locating the spam. An ideal filter would be one wherein spam messages are deleted without the supervision of the user and no legitimate messages are deleted in the process.

Table 2: The results (TCR) of four filtering techniques for different values of λ .

Filter Technique	λ	Spam Recall	Spam Precision	Weighted Accuracy	Baseline Weighted Accuracy	TCR
All Token	1	100%	69.4%	78%	50%	2.3
Fixed Token	1	100%	80.7%	88%	50%	4.2
Std Deviation	1	96%	72.7%	80%	50%	2.5
Percentage	1	100%	73.5%	82%	50%	2.8
All Token	9	100%	69.4%	60.4%	90%	0.25
Fixed Token	9	100%	83.3%	82%	90%	0.56
Std Deviation	9	96%	80%	78%	90%	0.45
Percentage	9	100%	73.5%	67.6%	90%	0.31
All Token	999	88%	71%	64.0%	99.9%	0.002
Fixed Token	999	80%	100%	100%	99.9%	5
Std Deviation	999	84%	84%	84%	99.9%	0.006
Percentage	999	76%	86.4%	88.0%	99.9%	0.008

One can observe that the value of $n_{S \rightarrow L}$ is much less than that of $n_{L \rightarrow S}$. This is due to the fact that the number of spam used in the training phase is much greater than that of legitimate messages. Our filter, being a self-learner, would improve its performance in the future and would keep the value of $n_{S \rightarrow L}$ as small as possible. We believe, after a period of time, our filter would perform at its peak performance and would remain constant thereafter. The ideal filter should give spam precision of 100 percent, spam recall of 100 percent and a positive value for TCR for all the values of λ .

B. Total Cost Ratio

The values of TCR and weighted accuracy show the better performance of 5 tokens over others for each value of λ . The performance degrades as we consider a higher number of tokens for the classification. However, the effective configuration still cannot be used as a stand-alone first-pass filter for $\lambda=999$ and $\lambda=9$. It needs help of other techniques, such as blacklists and whitelists, for effective spam filtering. To get an optimal number of tokens, we further evaluated by covering the range of 3 to 12 tokens. Tables 3 and 4 list their results.

The results of 7 and 10 tokens remained the same to each other as well as remained constant for different values of λ . However, the results for 3 fixed tokens were the worst, and results of 12 fixed tokens were near to that of 5, 7, and 10 fixed tokens. It can be said that in the case of the fixed token approach, the filter reaches optimal performance in the range of 5 to 12 tokens and degrades thereafter. This observation is confirmed by the plotted graphs (Figures 1 and 2). They

indicate the maximum peak (i.e. TCR value) in the range of 5 to 12.

Table 3: The results (false positives, false negatives and correct classifications) of four filtering techniques for different values of λ .

Filter Technique	λ	$n_{L \rightarrow S}$	$n_{S \rightarrow L}$	$n_{L \rightarrow L}$	$n_{S \rightarrow S}$
Fixed - 3	1	10	7	5	8
Fixed - 7	1	1	1	14	14
Fixed - 10	1	1	1	14	14
Fixed - 12	1	2	0	13	15
Fixed - 3	9	10	7	5	8
Fixed - 7	9	1	1	14	14
Fixed - 10	9	1	1	14	14
Fixed - 12	9	2	0	13	15
Fixed - 3	999	7	12	8	3
Fixed - 7	999	1	1	14	14
Fixed - 10	999	1	1	14	14
Fixed - 12	999	1	0	14	15

Table 4: The results (TCR) of four filtering techniques for different values of λ .

Filter Technique	λ	Spam Recall	Spam Precision	Weighted Accuracy	Baseline Weighted Accuracy	TCR
Fixed - 3	1	53.33%	44.44%	43.33%	50%	0.882
Fixed - 7	1	93.33%	93.33%	93.33%	50%	7.45
Fixed - 10	1	93.33%	93.33%	93.33%	50%	7.45
Fixed - 12	1	100%	88.24%	93.33%	50%	7.45
Fixed - 3	9	53.33%	44.44%	35.33%	90%	0.155
Fixed - 7	9	93.33%	93.33%	93.33%	90%	1.5
Fixed - 10	9	93.33%	93.33%	93.33%	90%	1.5
Fixed - 12	9	100%	88.24%	88%	90%	0.833
Fixed - 3	999	20%	30%	53.3%	99.9%	0.002
Fixed - 7	999	93.33%	93.33%	93.33%	99.9%	0.015
Fixed - 10	999	93.33%	93.33%	93.33%	99.9%	0.015
Fixed - 12	999	100%	93.75%	93.34%	99.9%	0.015

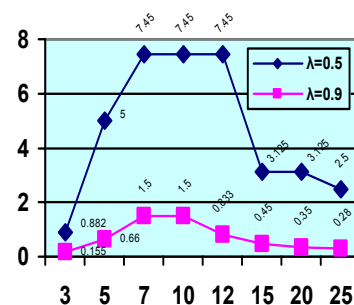


Figure 1: TCR for effective configurations of the fixed token approach at threshold values of $t=0.5$ and $t=0.9$ ($\lambda =0.5$ and $\lambda=0.9$).

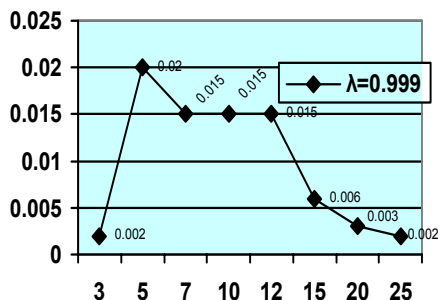


Figure 2: TCR for effective configurations of the fixed token approach at threshold values of $t=0.999$ ($\lambda=0.999$).

C. Fixed Token for Various λ

An evaluation of the fixed token approach was conducted with 15 tokens. To get an optimal anti-spam filter, we further evaluated the fixed token approach with a different number of fixed tokens, i.e. 5, 15, 20, and 25. Tables 5 and 6 list their results.

Table 5: The results (false positives, false negatives and correct classifications) of four configurations (5, 15, 20 and 25) of Fixed token approach for different values of λ .

Fixed Filter Number of Tokens	λ	$n_{L \rightarrow S}$	$n_{S \rightarrow L}$	$n_{L \rightarrow L}$	$n_{S \rightarrow S}$
5	1	4	1	21	24
15	1	7	1	18	24
20	1	8	1	17	25
25	1	10	0	15	25
5	9	4	2	21	23
15	9	6	1	19	24
20	9	8	0	17	25
25	9	10	0	15	25
5	999	1	3	24	22
15	999	4	1	21	24
20	999	7	0	18	25
25	999	9	0	16	25

V. DISCUSSION

In each spam filter, keeping the cost of false positives to a minimum becomes the prime priority. Our experiments show that all techniques of the Bayesian approach allow some number of false positives; however, some techniques keep the figure to a minimum. There are many other techniques studied that can be used along with a simple content-based filter to improve its performance. A lemmatizer that converts each word to its base form can be included in our filter. In that way, any modifications of the same word would not escape the attention of the anti-spam filter. For example, s^*e*x would be treated similarly to sex [8]. A stoplist that removes the 100 most frequent words of the British National Corpus (BNC) from messages is also helpful in cases like that of the all

Table 6: The results (TCR) of four configurations (5, 15, 20 and 25) of Fixed token approach for different values of λ .

Fixed Filter Number of Tokens	λ	Spam Recall	Spam Precision	Weighted Accuracy	Baseline Weighted Accuracy	TCR
5	1	96%	85.7%	90%	50%	5.0
15	1	96%	77.4%	84%	50%	3.1
20	1	100%	75.8%	84%	50%	3.1
25	1	100%	71.4%	80%	50%	2.5
5	9	92%	85.2%	84.8%	90%	0.66
15	9	96%	80.0%	78.0%	90%	0.45
20	9	100%	75.8%	71.2%	90%	0.35
25	9	100%	71.4%	64.0%	90%	0.28
5	999	88%	95.7%	96.0%	99.9%	0.02
15	999	96%	85.7%	84.0%	99.9%	0.006
20	999	100%	78.1%	72.0%	99.9%	0.003
25	999	100%	73.5%	64.0%	99.9%	0.002

tokens method wherein each word is responsible for assigning a final score to a message. A user can also add words manually to their stoplist to tune their personal spam filter.

Our evaluation does not take into consideration non-textual factors like images and attachments. There is a high probability that an email with no textual content but only an image or an attachment is a spam. Training the filter with a corpus containing non-textual content would improve its effectiveness during the classification phase. In the case of a hyperlink, we can have a web crawler that would visit the mentioned site and apply the same Bayesian approach to rate that page. If the score of that page goes above the threshold value, there is a possibility that the message containing the hyperlink is a spam. Thus, hyperlinks would be useful in assigning a final score to a message with the help of web crawler.

There are some specific traits that help us detect spam. For example, no sender's address and the use of dark red colors are some traits commonly found in spam. These traits are termed attributes. Attributes can be textual phrases like "only above 21 years." If the filter is trained for such attributes, there is a proven study of a positive change in results during the classification phase of the filter [1].

A Bayesian filter can also be used for classifying messages into different folders [2]. It can suggest to which specific folder a new message belongs. People create folders to organize their messages for archival purposes, messages that need replies and, of course, a bulk folder to have a second look at messages before deleting them as spam. But people with a large number of folders find it difficult to organize their messages. Given that a text classifier can choose a correct folder 85 percent of the time, chances are high that the appropriate folder would always be in the first three guesses. Thus, the user is restricted to choosing a folder out of three guesses, as opposed to choosing a folder out of some 20 odd folders [9].

Experiments have been conducted to study the workings of a Bayesian filter with dependent features. In our study, we assumed that features were independent of each other. But the results in the case of a 1-order dependent Bayesian filter are better than the Naïve Bayesian filter. This implies that phrases like “Click here,” and “Buy Free” are better indicators of spam than the independent words “Click,” “here,” “Buy,” and “Free”. If the order of dependency is increased above three, the results tend to decline. As the order increases, we have more conditional variables with the same amount of data. This complicates the probability estimates and, hence, leads to an overall decrease in predictable accuracy [11].

To get an efficiency above 99 percent with less than 1 percent false positives, a content-based filter would not be enough. Other approaches can be integrated with the Bayesian filter to get the desired result. For example, the rule-based and the blacklists approaches are simple to integrate. Doing so has proved helpful in spam filtering. The signature-based and filters fight back approaches are some of the advanced techniques that have also proved their usefulness when paired with a Bayesian spam filter

VI. CONCLUSIONS

Our cost-sensitive evaluation suggests that a content-based filter using a Bayesian approach alone is not sufficient to function as an anti-spam filter due to large number of false positives. However, the fixed token approach has been found to be the most effective among the four techniques evaluated in the report. The fixed token approach achieves its peak performance when the number of effective tokens selected to classify a message fall in the range of 5 to 12. This configuration performs satisfactorily for $t = 0.9$ ($\lambda = 9$) and $t = 0.5$ ($\lambda = 1$). No configuration performs well enough to be used for $t = 0.999$ ($\lambda = 999$).

To obtain an optimal anti-spam filter, we suggest the use of a lemmatizer, a stoplist and integration with other techniques, such as the blacklist and rule-based methods. The results are based on 80 personalized messages and we expect better results in the future as our filter follows a self-learning algorithm. Due to less variability in the content of spam messages, only some tokens were able to make an impact on the classification process. This made catching spam messages without blocking legitimate messages a bit difficult. As a first-pass filter, however, the few tokens technique must be analyzed to give maximum effectiveness. Thus, we believe that our fixed token technique with 5 to 12 token configurations would be able to make a positive contribution as a first-pass filter. However, this number might differ a bit from person to person depending on the type of spam message received.

REFERENCES

- [1] Androustopoulos, I., Koutsias, J., Chandrinou, K., and Spyropoulos, C. “An experimental comparison of naive Bayesian and keyword-based

- anti-spam filtering with personal email messages,” in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.
- [2] Cohen, W., “Learning rules that classify e-mail,” In *AAAI Spring Symposium on Machine Learning in Information Access*, 1996.
- [3] Domingos, P. and Pazzani, M., “Beyond independence: Conditions for the optimality of the simple Bayesian classifier,” in *Proceedings of the 13th Int. Conference on Machine Learning*, 1996.
- [4] Graham, P. *A Plan for Spam*. <<http://www.paulgraham.com/spam.html>> August 2002.
- [5] Graham, P. *Better Bayesian Filtering*. <<http://www.paulgraham.com/better.html>> January 2003.
- [6] Graham, P. <<http://www.paulgraham.com/antispam.html>> August 2002.
- [7] Potamias, G., Moustakis, V., and Van Someren, M. (eds.), “An evaluation of naïve Bayesian anti-spam filtering,” in *Proceedings of the Workshop on Machine Learning in the New Information Age*, 2000.
- [8] Provost, J., “Naive Bayes vs. Rule Learning in Classification of Email,” in *Artificial Intelligence Lab*, University of Texas at Austin, A technical report, AI-TR-99-284.
- [9] Rennie, J., “ifile: An application of machine learning to e-mail filtering,” in *KDD-2000 Text Mining Workshop*.
- [10] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E., “A Bayesian approach to filtering junk email,” in *AAAI Workshop on Learning for Text Categorization*, 1998, AAAI Technical Report WS-98-05.
- [11] Sahami, M., “Learning limited dependence Bayesian classifiers,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
- [12] Spertus, E., “Smokey: Automatic recognition of hostile messages,” in *Proceedings of the 14th National Conference on AI and the 9th conference on Innovative applications of AI*, 1997.
- [13] Langley, P., Wayne, I., and Thompson, K., “An analysis of Bayesian classifiers,” in *Proceedings of the 10th National Conference on AI*, 1992.
- [14] <<http://www.spamlaws.com/>> June 2004.
- [15] Brad Stone, “Spam Doubles, Finding New Ways to Deliver Itself,” *New York Times*, Late Edition - Final, Section A, Page 1, Column 2, December 6, 2006. <http://select.nytimes.com/gst/abstract.html?res=F10812FD3D550C758CDDAB0994DE404482>
- [16] http://www.spamfo.co.uk/component/option,com_content/task,view/id,291