Handling Redundancy in Email Token Probabilities
Version 0.94, released May 6, 2004

Gary Robinson
CEO, Transpose, LLC
grobinson@transpose.com

One of the many techniques which has recently been employed for filtering spam is one described in the Linux Journal article A Statistical Approach to the Spam Problem [Robinson]. This technique incorporates ideas from the seminal article A Plan For Spam [Graham] as well as R.A. Fisher's technique for combining p-values by means of the chi-square distribution [Hedges & Olkin]. The technique presented in here takes the chi-square-based approach a step further by taking into account two facts: a) there is redundancy in the token probabilities, and b) spam and ham emails have different amounts of such redundancy. Fivefold cross-validation was carried out on the new technique and is described here testing whether these factors actually lead to better performance. The results were positive and statistically significant.

BACKGROUND

Chi-square-based spam filters which the author is familiar with carry out the following steps:

For each word token in our corpus, we calculate

b(*w*) = (the number of spam e-mails containing the word w) / (the total number of spam e-mails).

g(*w*) = (the number of ham e-mails containing the word w) / (the total number of ham e-mails).

p(*w*) = b(*w*) / (b(*w*) + g(*w*))

Next we calculate:

$$f(w) = ((s * x) + (m * p(w)) / (s + m) \quad (1)$$

where

s is the strength we want to give to our background information;

x is our assumed probability, based on our general background information, that a word we don't have any other experience of will first appear in a spam; and

m is the number of e-mails we have received that contain word w.

We will refer to the results of Eq. 1 as "f(*w*)" or as Grahamian token probabilities, in reference to Paul Graham's article cited above. The definition in Eq. 1 is a bit different from Graham's in that it takes "prior knowledge" into account in a Bayesian style, in order not to be misled when there is only a small amount of data. However it is strongly derived from Graham's work and when there is a large amount of data, is equivalent to Graham's as a limit case.

It is useful for us to assume a "null hypothesis" that the f(*w*)'s are drawn from a population of random variables that is uniformly distributed and statistically independent. This assumption is false, of course. But it is a valuable characteristic of our f(*w*)'s that the more "spammy" or "hammy" an email is, the more in violation of that null assumption the f(w)'s associated with that email will tend to be; further, the degree of that violation can be measured in one direction or the other, giving us a tool us to classify emails as spam or ham.

R.A. Fisher showed that we can compute a p-value representing the confidence with which we can reject the null hypothesis as follows:

$$H = C^{-1}\left(-2\ln\prod_{w}\mathrm{f}(w),\ 2n\right) \qquad 2$$

where C^{-1} is the inverse chi-square function. The first parameter of that function is the chi-square value, and the second parameter is the associated degrees of freedom. H is the probability of encountering a value as extreme as the one actually encountered, or more extreme, under the assumption that the null hypothesis is true.

Note that this is, in a sense, a one-sided test. It is more sensitive to f(*w*)'s very near 0 than very near 1. Eq. 2 is therefore more sensitive to hamminess as made manifest by the presence of f(*w*)'s very near 0 than it is to values very near 1. In the presence of an unusual number of f(*w*)'s near 0, *H* is very near 0.

To maximize our ability to detect spam as well as ham, we do this calculation twice: once in a way that is sensitive to extremely hammy values of f(*w*), and once in a way that is sensitive to extremely spammy values. The spam-sensitive version is

$$S = C^{-1}\left(-2\ln\prod_{w}\left(1-\mathrm{f}(w)\right),\ 2n\right) \qquad 3$$

We then combine these p-values in a way that compares the certainty with which we can reject the null hypothesis in each direction. If we can reject it with more certainty in the spammy direction we may classify the email as spam; otherwise we may classify it as ham. A typical calculation to generate an indicator for this purpose is:

$$I = H/\left(H+S\right) \qquad 4$$

where *I* is near 0 for hams and near 1 for spams.

Note that some indicators, such as the one used by SpamBayes [SB], are designed to provide special "middling" values. It is worth taking some time to discuss the differences between these indicators because while SpamBayes-like indicators have convenient properties, for purposes of the technique described in the present article the evidence is that we are better off with Eq. 4.

The equation used in SpamBayes is essentially *H-S*, which can be brought onto the (0, 1) interval with simple tweaking:

$$I = \left(1+H-S\right)/2 \qquad 5$$

In particular, if H and S are both very close to 0, consistent with a great deal of evidence of hamminess as well as a great deal of evidence of hamminess, Eq. 5 ends up in the middle of its range to allow for such emails to be put in an "Unsure" box so that the user can conveniently choose to manually examine them. This is true even when there are orders of magnitude in difference between *S* and *H*, such as *S*=.0001 and *h*=.000001. Although it might not seem obvious why cases where *S* and *H* are orders of magnitude different should be treated as if *S* and *H* were equivalent, doing so has been found to be very useful. The reason is that, as Tim Peters of the SpamBayes project has pointed out, such a state indicates confusion. There is enormous evidence in each direction, even if the order of magnitude of that "enormity" is different in each case. Practical experience has indicated that such emails often cannot be reliably classified even when the order-of-magnitude difference is great. So Eq. 5 and its variants have been very useful in practice.

The Eq. 4 indicator is does not handle such cases the same way. Because it is focused on proportional difference between *S* and *H* rather than simple size differences, it does not necessarily place emails with *S* and *H* very near 0 into the middling Unsure range. It only does so if those values are close to one another, and whether or not they are near 0 makes no difference. Thus for applications where an Unsure category is desired, an additional test may be desired along with Eq. 4. Recalling that values of S and H that are near 0 are the critical ones indicating spamminess and hamminess respectively, we suggest simply checking for cases where *S* and *H* are both smaller than some threshold *q*, where *q* is chosen to control the number of emails that will be categorized as Unsure. .001 would be a good candidate value. Emails for which *I* is very near .5 would also be labeled Unsure.

However our task in this article is to improve basic binary classification. For purposes of binary classification, a "cutoff" value is chosen and the email is simply considered to be spam if *I* is above the cutoff and ham if it is below it. Sometimes, a value of .5 will be used. However, there is no theoretical reason why the cutoff should be exactly .5; rather the cutoff should be chosen for maximum efficacy. One reason why the ideal cutoff may not be .5 is that there may usually be a higher proportion of very spammy words in a spam email than very hammy words in a ham email.

REDUNDANCY

For the chi-square-based approach to have the most value, we would like email characteristics that result in more extreme values for *S* and *H* to always be characteristics that are indicative of spamminess or hamminess. However in reality, *S* and *H* are influenced by a combination of characteristics, some of which aren't helpful to us. In particular, there is a redundancy in data because spammy and hammy words usually come in groups; if an email contains the word 'java' there is a significantly higher probability that it also contains 'source' and 'code', for example.

Note that we are not calling this effect "correlation" because the calculation isn't working with probabilities corresponding to a fixed set of tokens; that is there is no fixed dimensionality. Because of this we cannot run the token probabilities through a standard multivariate statistical calculation. Rather, from the standpoint of Eq. 2, we are dealing with sets of anonymous numerical values. Without pairing the values based on their identities such that a correlation matrix can be built, it doesn't seem to make sense to talk about correlation.

Possibly, the redundancy would not be a problem if we always used *I*=.5 as the spam/ham cutoff for classification purposes. Arguably, whether spamminess or hamminess dominated, that measured result for an email would become stronger as the number of tokens in the email increased, and .5 would remain the cutoff point.

However, as noted above, there is no reason to assume that .5 is the ideal cutoff value. And for any cutoff value than .5, it is a problem that there is some skew toward spamminess or hamminess that will be distorted in the direction of being more extreme as the number of tokens in the email grows. The problem is rooted in the fact that the average proportion of redundant values remains constant as the number of tokens grows. As the number of tokens in an email grows, this fixed proportion would be more and more unlikely under the null hypothesis. Our chi-square test is sensitive to the degree of such likelihood, and so the output of the inverse chi-square function become more and more extreme.

The problem with this changing amount of skew is that it makes it impossible to pick a value for *I* that means the same thing as the number of tokens grows. For instance, suppose *S* is .001. We are then saying that only 1 in 1,000 times would that value for *S*, or a more extreme one, have occurred if the null hypothesis were true. Notice that *n*, the number of tokens, does not need to be considered once *S* has been calculated. *S* stands on its own and has the same 1-in-1,000 meaning, regardless of *n*. It has a very concrete meaning grounded in probability. The same goes for *H* and therefore *I*. In fact this is the core reason for using the chi-square technique in the first place; no matter how many tokens we have, *I* has, in a useful sense, the same meaning. This lets us pick a single *I* as the spam/ham cutoff point to use for all our emails.

However, to the extent that there is any distortion effect that predictably makes values more extreme as the number of tokens in an email increases, this valuable property of *I* is diminished. *I* then means something different for differently-sized emails. We can no longer choose a single spam/ham cutoff point for all emails. And to the degree that we lose the ability to meaningfully choose a single cutoff for all our emails, our classification task becomes that much more difficult.

So, it is important to try to take this redundancy into account in a way that brings this cumulative distortion under control.

To see how to do this, let us consider an imaginary case. Suppose we lived in a universe where words were always used twice, as in "the the weather weather is is hot hot." If we examined the f(*w*)'s contained in an email, every value would appear an even number of times due to this redundancy. But this doubled number of tokens would add no useful information for determining whether the email was a spam or a ham. It would have exactly the same amount of information, in fact, as would an equivalent email in our universe where words are only used once.

Let us also suppose that in this imaginary universe, there is no other redundancy than the doubled words. We shouldn't apply Eq. 2 to the f(*w*)'s because then we'd get the redundancy-based distortion we've been discussing. However, if we simply transform the email from one in the imaginary universe to one in a universe without word-doubling by eliminating the duplicate words, we would have half as many f(*w*)'s and the redundancy would be gone, so we could apply Eq. 2.

Of course we wouldn't need to actually make a copy of the email that removed the duplicate words; rather we could equivalently change Eq. 2 to use the square root of the product of f(*w*)'s and half the value of n. We could think of this as saying that the "effective size" of the email is

half its actual size. We can represent this situation by the term "effective size factor" or ESF. In this case the ESF is .5.

Obviously, the redundancy in real-world emails isn't as cut-and-dried as in our imaginary-universe example. However, it seems intuitively possible that different degrees of redundancy could be usefully represented by different ESF's in a similar manner to our handling of the doubled-world universe. In fact, testing on real-world data shows that this is indeed the case.

One example of such testing in the task of protein classification [Baily & Grundy], where use of the "effective size" concept has increased the reliability with which proteins can be properly classified. (Note, that research has no equivalent of our combination of *S* and *H* into a single indicator *I*; it would be interesting to try and adapt this techique to be used in that contect.)

In adapting the effective size concept for the spam/ham classification task, we modify Eq. 2 as follows, where *y* is the ESF:

$$H = C^{-1}\left(-2\ln\left(\prod_{w} \mathrm{f}(w)\right)^{y},\ 2ny\right) \qquad 6$$

When *y* is .5, the result is taking the square root of the product and half the DF, as in our example. But any real-valued *y* on the unit interval may be used. And it turns out that this modification leads to a significant improvement in spam/ham classification reliability.

It may be useful at this to point out that due to the redundancy in our example, $-2\ln\left(\prod_{w} \mathrm{f}(w)\right)$ is not in a chi-square distribution under the null hypothesis, whereas $-2\ln\left(\prod_{w} \mathrm{f}(w)\right)^{y}$ is. So, in a sense, one can look at the ESF as a way to create a chi-square random variable so that we can apply Fisher's inverse chi-square calculation to it.

Now, in real-world use, there is no reason to assume that the redundancy is of the simple form it takes in our example that could be fully compensated for some value of *y*. In general, it probably does not. Nevertheless, testing on real-world shows that enough compensation occurs that significant benefit in classification can be obtained by using this procedure.

Because real-world data is not as simple as we would like, we do not presently know of a direct way for calculating *y*. Instead, current procedure is to tune *y* for best overall performance in the context of the spam classification process. An important ramification of that procedure is that we can't really say that we are necessarily tuning *y* so that *H* is a chi-square random variable under the null hypothesis. The idea that we are doing so is the basic motivator of the approach. But the real-world data is complex in structure, and it is quite possible that the *y* that performs best will have other implications which improve classification performance other than obtaining a chi-square distribution from Eq. 6. We are interested in spam classification, not the abstract goal of creating a chi-square distribution, so whatever *y* works best for the end-task is fine with us. That is, we see that the aspect of *y* that moves Eq. 6 toward having a chi-square distribution under the null hypothesis should make the classifier work more consistently with the underlying theory upon which is based, and so should improve performance. But whatever other implications it may have are fine as long as in the end, performance is improved – and testing shows that it is.

It is instructive at this point to note another interesting and useful fact. The ESF for the calculation for *S* is not necessarily the same as the ESF for the calculation of *H*. This is due to the phenomenon, revealed by testing, that spams and hams have different levels of redundancy, and therefore different ESF values can be used to compensate for them.

This dovetails very nicely with the circumstance that the basic chi-square technique already calculates the inverse chi-square value separately for sensitivity to spamminess and hamminess. Because we are already doing these two separate calculations, we are well-prepared to make good use of any different levels of redundancy that appears in spams vs. hams for the purpose of improved spam detection.

This ability to take advantage of such different levels of redundancy being associated with spam and ham creates another motivation for using the chi-square technique.

One technical detail that emerges in implementing an ESF-based approach is that since *y* in Eq. 6 is a real number on the unit interval, the degrees of freedom to pass into the inverse chi-square function is not necessarily an integer, and in fact may even be less than 1 for small emails. Following the above-mentioned protein research, we can handle this by interpolation. (Example code: [Robinson Chi]. There is also an inverse chi function capable of handling non-integer DF in the GNU GSL library [GNU GSL].)

TESTING

The basic chi-square technique has been the subject of a substantial amount of public testing leading to its adoption in such filters as SpamAssassin, SpamBayes, and Bogofilter. See the documentation accompanying those projects for more information on that testing. Our current task is to determine whether the ESF concept can improve the performance of chi-square-based spam filters.

Note that our test setup does not attempt to recreate an entire spam filter. Real spam filters have many bells and whistles that we do not include; we do no processing other than computing token probabilities and using our chi-square-based techniques to generate *I* and compare it to a cutoff value. So the results reported here should not be compared with extant spam filters, but should only be used to determine whether the ESF concept improves upon the basic chi-square-based technique.

Our test uses the SpamAssassin public corpus [SA]. This corpus is divided into separate files with different classification characteristics (such as "hard" and "easy"). These different characteristics were not needed for present purposes, so we started by concatenating the files into one large spam file and one large ham file.

We then performed five-fold cross validation. In each of 5 runs, one fifth of the data was used for training purposes and the rest for testing purposes. No training data was reused for training in different runs. In each of the five "folds" the data was drawn uniformly from the entire concatenated SpamAssassin corpus, by taking every 5^{th} email starting with an offset into the database between 0 and 4 inclusive.

Each run determined the number of misclassifications with and without use of ESF's.

We assumed all parameters were unknown at the outset of training. In the course of training we determined the f(*w*)'s, *s* (from Eq. 1) and one other parameter which we call the "exclusion radius" (explained below). Additionally, training in the test of ESF efficacy also determined the two ESF values.

Thus, the null hypothesis being tested is: a run consisting of first training all parameters including ESF's, and using those parameters on test data, leads to the same mean number of misclassifications as following the same procedure but excluding ESF's in the training and testing. We then compute a p-value representing the confidence with which we can reject that null hypothesis in favor of the alternative hypothesis that there are fewer misclassifications when ESF's are used.

Testing conducted by the SpamBayes and Bogofilter teams, as well as elsewhere, has shown that excluding the f(*w*)'s near .5 is useful. Values near .5 are either of little value because they either represent tokens that aren't significantly hammy or spammy, or else are of little practical use because there aren't enough historical occurrences of them for their associated probabilities to be reliably calculated. (In the latter case, the values are near .5 because we assign *x* a value of .5 in Eq. 1.) So training includes finding the best exclusion radius; for instance if the exclusion radius is .25, tokens with associated f(*w*)'s between .25 and .75 are excluded from all further calculations.

To find the f(*w*)'s, we did the calculations described for Eq. 1. All other parameter values were found by the following process. First we divided each of the two training files into two equal-sized parts. Then, we exhaustively tested all possible permutations of values for the parameters based on a predetermined list of allowable values for each parameter. Note that this "testing" is occurring within the training data; the actual testing data is not used at this stage. The idea is that we want to get good values for all parameters; other than the f(*w*)'s themselves, this can only be accomplished through an optimization process involving testing the efficacy of different combinations of parameters. So we carry out this optimization on the training data, which involves using half the training data to find the f(*w*)'s and the other half to use those f(*w*)'s to test the performance of various combinations of parameters. We divided the data up in this way because we wanted to better simulate the real-world use situation where we are classifying emails based on pre-existing values for f(*w*). We wanted to find values of the parameters that would be robust for situations where the word probabilities among the emails we were classifying were not necessarily identical to the ones used to generate f(*w*)'s.

The combinations of values that most accurately classified the second parts of the training data were then used to classify the testing emails based on a new set of f(*w*)'s, where these new f(*w*)'s were generated from the training data (associated with the current fold of the fivefold cross-validation) taken in its entirety.

Of course, such methodologies as simulated annealing or genetic algorithms could be applied to the problem of finding the best combination of parameters, but for present purposes it was felt that a simple exhaustive search for the best combination was sufficient.

We did two runs of this process. In the first run the ESF's were held fixed at 1.0, and in the second, they were allowed to vary. Also, in the first run we used Eq. 5 to compute *I* so that we were using the technique that is familiar from SpamBayes, and in the second run we used Eq. 4.

The SpamAssassin corpus as used here contains 9,354 emails of which 2,400 are spams and 6,954 are hams.

The tested values for the various parameters were as follows:

Exclusion radius: .45, .4, .25, .1, .05.

s: 1.0, 0.1, 0.01

spam/ham cutoff: .01, .02, …, .99

ESF (both spam-sensitive and ham-sensitive): $.75^0$, $.75^1$, …, $.75^{19}$

All permutations of those parameter values were tested, for a total of 5*3*100*20*20=600,000 tests (and 1,500 tests where the ESF values were held at 1.0).

The same training and testing data were used for the tests with ESF's held constant as for the tests where ESF's varied.

Indicator from Eq. 5 ("SpamBayes"); ESF held at 1.0 (that is, not used)				Indicator from Eq. 4; ESF allowed to vary						
s	Exc. Rad.	Spam/ Ham Cutoff	Errors	*s*	Exc. Rad.	Spam ESF	Ham ESF	Spam/ Ham Cutoff	Errors	P-value
.1	.45	.04	102	.01	.45	.017818	.005638	.23	61	.000823
.1	.45	.35	87	.1	.4	.75	.010023	.91	89	0.589
.1	.45	.34	62	.1	.4	1.0	.017818	,51	64	0.605
.1	.45	.10	81	.01	.45	1.0	.056	.01	74	0.315
1.0	.45	.13	96	1.0	.1	.017818	1.0	.02	88	0.303

The decision was made to use the following procedure to determine the statistical significance of the results. (No alternative approaches were tried for determining significance.)

First we considered each of the five "folds" separately and determine a p-value based on the binomial distribution. That is, if the ESF-enabled runs had the same mean error rate as the ESF-disabled runs, then if we add up the total errors from the two runs (disregarding the identity of the email the error applied to), half of them should be from the ESF-enabled run. Thus, we have a binomial distribution where the experiments are the individual errors, and a failure occurs when an error is from the ESF-enabled run. The null hypothesis is that the failure rate is .5.

We then combine these p-values using R.A. Fisher's chi-square combining technique. This technique was first used by Fisher to combine the results of various studies where a p-value was obtained from each one [Hedges & Olkin]. This meta-analytical technique is the original domain of the Fisher calculation. Fisher's calculation has been adopted to the purpose of spam detection,

but its roots are in this process of combining p-values from multiple studies, and so we use it for that purpose here.

The p-values resulting from the binomial distribution appear in the rightmost column of the table. The combined p-value determined by Fisher's method is .0213. Using the usual standard for testing for statistical significance -- an alpha-level of .05 -- the results are statistically significant.

One very interesting data point occurs in the fifth fold (last row of the table). Instead of an exclusion radius of .45 or .4 as in the other folds, the exclusion radius is .1. Simultaneously, the relationship between Spam ESF and Ham ESF is reversed: in all other folds the Spam ESF is much greater than the Ham ESF, while in the fifth fold the opposite is true. In fact the values of those parameters are exactly backwards from the third fold.

This leads one toward the conclusion that the f(*w*)'s that are closer to .5 contain more redundancy on the spam side than the ham side, and that the opposite is true for the values that are farther from .5.

Another important conclusion is that one can't obtain training parameters by doing this kind of five-fold test and then averaging the parameter values that emerge. Rather the parameter landscape apparently has at last two peaks in very different regions of the landscape, and best results will be obtained by using parameter combinations from one of those peaks. The alternative of arriving at some in-between valley by averaging is unlikely to work as well. (A possible topic for future research would be to apply multiple parameter sets, representing different peaks, to each email and then to combine those results into an overall indicator.)

While five-fold cross-validation is useful for determining whether a statistically significant effect exists, we suggest that for real-world training, all available training data be used in one massive training run, ideally repeated at intervals as more data arrives.

One last note related to testing. As mentioned above, the indicator defined by Eq. 5 is insensitive to the multiplicative or proportional relationship of *S* to *H*, which is useful information to us. Rather Eq. 5 is sensitive only to the simple difference in size. In testing the ESF-based approach in combination using Eq. 5 as the indicator, no statistically significant result was attained. It is surmised that this non-result may have been due to the insensitivity of Eq. 5 to the proportional differences. Intuitively, a few moment's reflection may convince the reader that since the use of ESF's can change the order of magnitude of both *S* and *H*, it might be very useful to be sensitive to proportionate differences. That is, by using ESF's that create a particular proportional relationship between *S* and *H*, useful information may be revealed that is only available when the the proportional difference is considered, instead of being ignored, as it is in Eq. 5. This intuition appears to be borne out in testing.

Further research could clarify this issue. In the meantime, it is suggested that Eq. 5 and its variants not be used with ESF's.

As mentioned previously, a simple additional test that *S* and *H* are both smaller than some constant *q*, used in addition to Eq. 4, should replicate the practical advantages to using Eq. 5 for classifying emails to an Unsure label.

Note also that in another fivefold cross-validation run which compared the results of using Eq. 4 with and without ESF's resulted in a combined p-value that was even more statistically significant in rejecting the null hypothesis that ESF's made no difference. However, that stronger result was due to the fact that without ESF's, the classification scores were a bit worse with Eq. 4 than with

Eq. 5. We are unaware of a statistically significant result comparing Eq. 4 and Eq. 5 without ESF's, but the research described here finds no reason not to use Eq. 5 when ESF's are not used.

Finally it may be worth noting the Greg Louis of the Bogofilter project has also done testing of the ESF procedure [Louis], with a much larger dataset. His test also has significant positive results.

CONCLUSION

The test described here achieved a statistically significant result indicating that the ESF concept can be used to enhance the performance of chi-square-based spam classifiers.

This concept allows us to take advantage of the fact that the chi-square-based technique separately considers evidence for spamminess and evidence for hamminess in such a way that differing redundancy levels in spam and ham emails can be taken into account.

At the time of this writing, the author knows of no testing of the chi-square-based technique in the context of phrase-based spam filtering algorithms such as CRM114. There are clear advantages to the phrase-based techniques because certain phrases can be more strongly associated with spam or ham than most individual words can. However, most such techniques include phrases as well as subphrases in their calculations, which introduces another form of redundancy. The author believes that the ESF technique can be used to reduce or eliminate that redundancy, hopefully improving the accuracy of phrase-based techniques.

BIBLIOGRAPHY

[SA] http://spamassassin.org/publiccorpus/

[SB] http://spambayes.sourceforge.net/background.html

[Graham] http://www.paulgraham.com/spam.html

[Hedges & Olkin] Hedges, L.V. and Olkin, I. (1985). Statistical Methods for Meta Analysis, New York, Academic Press.

[Baily & Grundy] Bailey, T.L. and Grundy, W.H. Proceedings of the Third international conference on computational molecular biology (RECOMB99), April 11-14, 1999. pp. 10-14.

[Robinson] http://www.linuxjournal.com/article.php?sid=6467

[Robinson Chi] http://garyrob.blogs.com/chi2p.py

[GNU GSL] ftp://ftp.gnu.org/gnu/gsl

[Louis] http://www.bgl.nu/bogofilter/esf.html