A TREC along the Spam Track with SpamBayes

Tony Andrew Meyer
SpamBayes Development Team
tony.meyer@gmail.com

Abstract

This paper describes the SpamBayes submissions made to the Spam Track of the 2005 Text Retrieval Conference (TREC). SpamBayes is briefly introduced, but the paper focuses more on how the submissions differ from the standard installation. Unlike in the majority of earlier publications evaluating the effectiveness of SpamBayes, the fundamental 'unsure' range is discussed, and the method of removing the range is outlined. Finally, an analysis of the results of the running the four submissions through the Spam Track 'jig' with the three private corpora and one public corpus is made.

1 SpamBayes

SpamBayes [1] was born on August 19th 2002, soon after publication of *A Plan for Spam* [2]; Tim Peters and others involved with the Python development community developed code based on Graham's ideas, with the initial aim of filtering python.org mailing-list traffic, although this quickly progressed to also filtering personal email streams (today, although most python.org mailing lists do use SpamBayes, the overwhelmingly most common use of SpamBayes is for personal email filtering). Although the project initially started with Graham's original combining scheme, it currently uses the chi-squared combining scheme developed by Robinson.

The SpamBayes distribution includes a plug-in for Microsoft™ Outlook™, a generic POP3 proxy and IMAP4 filter, various command-line scripts, and a suite of tools for testing modifications of the tokenization/classification systems. The software is free[1], released under the Python Software Foundation Licence. A basic introduction to the distribution can be found in [3]. The testing outlined in this paper used version 1.1a1; the only modifications were adjustments of the client/server architecture to improve the speed of the TREC testing.

2 TREC 2005 Spam Track Corpora

This paper covers results from testing SpamBayes with the three 2005 TREC spam track private corpora, and the 2005 TREC spam track public corpus (the full corpus, and four subset variants). As the author had access to the *TM* corpus prior to the 2005 TREC spam track submission, no consideration will be made of the results of the runs against this corpus in this paper, although these runs were performed (results were generally similar to those of the *Public* and *MrX* corpora). Table 1 outlines the vital statistics for each of the copora; more details about the composition of the eight corpora, including a detailed explanation of the creation of the *Public* corpus and a breakdown of the *SB* corpus into both ham and spam genre can be found in the overview of the spam track for TREC 2005 [4].

[1] Beer and speech.

3 Handling the 'unsure' range

3.1 The unsure range in practice

SpamBayes uses Robinson's chi-squared probabilities combining scheme [5]. A chi-squared test calculates the probability that a particular distribution matches a hypothesis (in this case that the message is spam and, separately, that the message is ham). The results of these two chi-squared tests are then combined and scaled to give an overall message spam score in the range 0 to 1. The 'unsure' middle ground is defined as any message with a final score falling between an upper and lower bound. In the SpamBayes distribution, the default unsure range is a message with a final combined spam score between 0.20 and 0.90; this lack of symmetry reflects an aversion to false positives. Messages tend to score at the extremes of the range, apart from difficult to classify messages (those that strongly resemble both ham and spam, and those that do not resemble either) which fall near 0.5.

Corpus	# Ham	# Spam	Ham::Spam Ratio
MrX	9038	40048	0.23::1
SB	6231	775	8.04::1
TM	150685	19516	7.72::1
Public (Full)	39399	52790	0.75::1
Public (Ham 25)	9751	52790	0.18::1
Public (Ham 50)	19586	52790	0.37::1
Public (Spam 25)	39399	13179	2.99::1
Public (Spam 50)	39399	26283	1.50::1

Table 1: Breakdown of testing corpora.

A remarkable property of chi-combining is that people have generally been sympathetic to its 'unsure' ratings: people usually agree that messages classed unsure really are hard to categorize. For example, commercial HTML email from a company you do business with is quite likely to score as unsure the first time the classifier sees such a message from a particular company. Spam and commercial email both use the language and devices of advertising heavily, so it is hard to tell them apart. Training quickly teaches the system how to identify desired commercial email by picking up clues, ranging from which company sent it and how they addressed you, to the kinds of products and services it offers.

SpamBayes users typically experience no false positives; this is not from an inherent strength of SpamBayes over similar statistical (or other) filters, but as a result of the unsure range. Essentially, the messages that would otherwise have been false positives are classified as unsure. The advantage of this system is that the volume of mail that the user must scan to find errors (both false positives and false negatives) is greatly reduced; typically between one and five percent of messages are classified as unsure, which is generally much lower than the percentage of mail that is spam.

As a result, users are more likely to take the time to scan the unsure folder than they would be to scan the entire spam folder, more able to identify the correct classification (rather than missing a false positive in a crowded spam folder) and more likely to appropriately train messages therein. The disadvantage of this system is that the percentage of messages that are classified as unsure is typically higher than the combined percentage of false negative and false positive messages obtained when using a classifier that does not include an unsure range. In simple terms, more messages must be manually corrected, but fewer messages must be manually examined.

3.2 Treatment of the unsure range for TREC 2005

Although any spam filter that generates a score (rather than a simple binary decision) for messages could be adapted to include an unsure range, the range is fundamental to the classifier that SpamBayes currently uses; there are three distinct

clusters of messages scores (near 0.0, near 0.5, and near 1.0). Note that the classification is not a true ternary classification, however, such as systems like POPFile [6] and CRM114 [7] may provide, merely a method of distributing the message scores across the potential 0.0 – 1.0 range.

The difficulty arises when attempting to utilise SpamBayes as a simpler binary decision engine, requiring a "ham" or "spam" output. The options are to either change the combining scheme that is used, or to set the ham-unsure and unsure-spam thresholds to the same value; while using a different combining scheme is likely to be more effective, this then considerably changes the method that SpamBayes is using, distancing any test results from results that could be expected from 'real life' usage. As such, for the purposes of the TREC spam track, the single-threshold technique was used – the resulting question is then to decide what value to assign to this threshold.

While 0.5 is the most obvious choice, in practice, the unsure range tends to include more spam than ham; this is most likely the result of both the growing ham::spam imbalance in the average mail stream and the greater homogeneity (particularly over time) of ham as compared to spam. With this consideration, a threshold value around 0.4 would likely produce the best results. However, the lower the threshold is, the greater the chance of incurring additional false positives; since a false positive is generally considered many times worse than a false negative, the threshold for all four variants of the 2005 TREC spam track runs was set to 0.6 (incurring additional false negatives in order to avoid as many false positives as possible).

The figure of 0.6 was chosen fairly arbitrarily; little analysis was done to determine an optimal value. The primary reason for this is that it is the score output (see ROC analysis below) that is of more interest to the author than the simple false negative/false positives results; in addition, it appears that the balance of ham and spam in the unsure range is highly sensitive to the balance of ham and spam in the entire mail stream, which was an unknown factor in the tests. Automatic adjustment of the thresholds during the run was considered unnecessary, due both to the greater interest in the ROC analysis and the desire to keep the simulation as similar to 'real use' of SpamBayes as possible.

4 Run Variants

Four variants of the SpamBayes distribution were submitted to the 2005 TREC spam track; all four variants included the same files, with selection of the variant completed by a simple script that selected the training script and SpamBayes configuration file to use. All four variants are possible using the standard SpamBayes distribution.

4.1 tam1 – train-on-everything/defaults

The first submission, tam1, uses all the default settings in the 1.1a1 SpamBayes distribution, other than the 0.6 value for the ham and spam thresholds (as above); this is also identical to the SpamBayes sample script provided in the initial 2005 TREC spam track 'jig', other than the threshold values, which were there set to 0.9. A brief outline of the tokenization methods used can be found in [3]. The training regime used in tam is train-on-everything. Every message is trained with the correct classification immediately after classification, and before any further message is classified.

This technique provides the classifier with the most information, but not necessarily the highest quality information; previous testing [8, 9] has shown that more minimalist training regimes are not only faster and result in smaller databases, but deliver superior results. The expectation was that *tam1* would be the least effective of the four submissions, and was primarily intended as a base with which to compare the other results.

4.2 tam2 – fpfnunsure/bigrams

While early testing [10] showed that using either character or word n-grams was less effective than simple split-on-white-space unigrams, in late 2002 Robinson and Robert Woodhead independently came up with an idea for using both

unigrams and bigrams, with a twist to avoid generating highly correlated clues. This idea was cleaned up and implemented by Peters, although it was only added to the SpamBayes code in late 2003; the technique is further outlined in [3].

The *tam2* submission differs from *tam1* in only two respects: it enables the *use_bigrams* scheme outlined above (retaining default values for all non-threshold options as in *tam1*), and it uses the *fpfnunsure* training regime (train on all false positives, false negatives, and unsure messages[2]) rather than train-on-everything. This is the training regime that SpamBayes recommends users employ. Previous testing has indicated that the tiled unigram/bigram scheme delivers superior results in almost (but not all) corpora [3]; as above, testing has also indicated that a *fpfnunsure* training regime is always superior to train-on-everything.

As a result, the expectation was that *tam2* would considerably outperform *tam1*; one complication is that the tiled unigram/bigram scheme does often increase the number of messages in the 'unsure' range, which is treated unusually (for SpamBayes) in the TREC submission. In addition, some previous testing [3] has shown the tiled unigram/bigram scheme to decrease accuracy when used in an 'incremental' testing setup (e.g. the incremental setup in [3], and the 2005 TREC spam track jig), rather than in cross-validation testing; this is at odds with the anecdotal evidence of use of the scheme in practice, however.

4.3 tam3 – train-to-exhaustion/bigrams

The third SpamBayes submission, *tam3*, uses identical tokenisation and classification options to *tam2*, but uses the train-to-exhaustion training regime. 'Training to exhaustion' [11] is a regime developed by Gary Robinson that resembles the perceptron algorithm [12]. In this regime, a collection of ham and spam are repeatedly trained (using any training method, although typically not train-on-everything; here *fpfnunsure* is used) until all messages in the collection are able to be successfully classified, or an iteration limit is reached. The regime aims to find the smallest subset of messages that, when trained on, will correctly identify the largest subset of messages.

Although previous testing [8] has indicated that train-to-exhaustion delivers results that are superior to train-on-everything, non-edge training, *fpfnunsure*, and *fpfn*, the training regime is much more resource intensive than any of the other regimes. In each iteration, all messages in the collection must be reclassified (whereas in the other regimes, no reclassification is done), and multiple messages may be trained (whereas in the other regimes, a maximum of one message is trained). In addition, more than one iteration is likely to be required (for *tam3* and *tam4* a maximum of ten iterations was permitted). The regime is therefore more suitable to batch training than training after every message.

The initial concept for using train-to-exhaustion with *tam3* and *tam4* was to include the entire known corpus in the training collection (i.e. after classifying 5,000 messages, there would be 5,000 messages in the training collection). This proved to be far too slow with large corpora, however, and so the final submission used the most recent 1,000 ham and 1,000 spam (for the first 1000 ham and 1,000 spam, all available messages were used). Although a batch training mode could have been used (e.g. only training every hundredth message, or at the end of every 'day'), the submissions used the simpler method of executing a full, 2,000 message, 10-iteration-maximum, train-to-exhaustion cycle for every incorrectly classified message.

This technique proved far too slow in practice, far exceeding the two-second-per-message intended limit for the spam track jig. As a result, the larger corpora (the public corpus, in all but one variation, and the TM corpus) did not complete

[2] With typical SpamBayes use, where the unsure range is present, this regime differs from *fpfn* (false positives, false negatives, but not unsure messages); both, however, are commonly referred to as train-on-error and other similar names. For the 2005 TREC spam track, these two regimes are, in fact, identical, of course, since the unsure range was eliminated.

the *tam3* and *tam4* runs. If the train-to-exhaustion regime is used in future 'incremental' style simulations, such as the 2005 TREC spam track jig, then implementation of a batch version of this regime would be necessary. A potential method would be to batch-train once per simulation day, including only messages received during that 'day' (rather than the most recent 1,000 ham and 1,000 spam), and to add to the existing database, rather than replace it.

The train-to-exhaustion regime includes two, potentially significant, improvements over *fpfnunsure*, in addition to the cyclical, backtracking, nature of the regime. The first is that if a limit is placed on the number of ham and spam included in the training collection, and the database is recreated with each train-to-exhaustion cycle, then messages are automatically expired from the database. With the *tam3* and *tam4* submissions, this expiry was age-based – only the most recent 1,000 ham and 1,000 spam could be used for training; other selection (and therefore expiry) methods are also possible.

In addition to the automatic expiry, the train-to-exhaustion regime automatically balances the database. Each train-to-exhaustion iteration processes the ham and spam collection independently, classifying and potentially training one ham message, then one spam message (and so on). Any messages left after one collection is exhausted are ignored for that training cycle (although for the *tam3* and *tam4* submissions an equal number of ham and spam were provided, so this would never be the case). Like other statistical filters, SpamBayes appears to be fairly sensitive to large imbalances between the number of ham and spam trained (see also below), and so it is expected that results would improve as a result of this automatic balancing.

4.4 tam4 – train-to-exhaustion/all options

The fourth, final, submission, *tam4*, uses the train-to-exhaustion training regime, like *tam3* (above); as a result, it too failed to complete the runs of the larger corpora. The difference between *tam3* and *tam4* is that *tam4* enables nearly all of the optional tokenizer techniques that SpamBayes 1.1a1 includes. The intent behind this submission was to get a general idea of whether the 'turn everything on' attitude that some users have would be beneficial or not; ideally each of these options should be considered individually (as was the case before they were added to the distribution), but this was outside the scope of the TREC evaluation. The hope was that *tam4* would outperform *tam3*; even if one of the additional options was detrimental, it ought to be countered by benefits from other options. *tam4* used the same thresholds as the other three submissions, and used the tiling unigram/bigram scheme as *tam2* and *tam3*. A brief outline of the other various options appeared in the track notebook paper for the submission; the reader is referred to the SpamBayes website[3] for more details.

5 Results

5.1 False positive and false negative rates

Tables 2 through 5 outline simple false positive and false negative results for the two[4] private corpora and (full) public corpus, along with (for reference) the total size and ham::spam ratio for each corpus. At first glance, it appears that SpamBayes performs somewhat better than (the author) expected, given the 'unsure' range difficulty, but that all submissions perform extremely poorly on the *SB* corpus. The *MrX* and *Public* results are reasonably similar across the four submissions.

[3] http://spambayes.org

[4] As above, the TM corpus is excluded from these results.

5.2 Explaining the poor *SB* corpus results

It is difficult to determine why performance was so poor for the *SB* corpus; the two obvious differences are that the corpus is relatively small and is weighted much more towards ham than spam[5]. One possibility is that the ham and spam are unevenly temporally distributed, so that a large number of ham are classified before any spam are seen; this particularly fits with the decrease in false negatives between *tam1* and *tam2* – since *tam2* only trains when a mistake is made, this imbalance would have a much smaller effect. This could also explain the shift from excessive false negatives to high (but not as high) false positives and false negatives when comparing *tam1* and *tam2* with *tam3* and *tam4*. Since the train-to-exhaustion regime reorganises the training order of messages, alternating between ham and spam, and also forces a 1::1 ham::spam ratio, this imbalance would have a significantly different effect.

The *SB* corpus is largely composed of mailing-list messages (48% of ham) and three frequent correspondents (12% of ham) [4]. "List" spam was particularly likely to be classified as ham (40, 8, and 4 messages for *tam1*, *tam2*, and *tam3* respectively); this is not entirely unexpected, as mailing list messages tend to have a large number of tokens that are present in every message. Since around 3,000 mailing-list messages were trained as ham, these tokens end up as very strong ham clues; the best solution to this problem is to install a good spam filter in the mailing-list delivery process (as this filter has access to the raw message, prior to the mailing-list specific tokens being added to it). A similar explanation probably applies to "newsletter" spam (14, 10, and 5 messages for *tam1*, *tam2*, and *tam3*, respectively).

"Sex" spam was also particularly likely to be classified as ham (147, 48, and 17 messages for *tam1*, *tam2*, *tam3* respectively); this is much more difficult to explain. One possible explanation is that the ham messages included many messages about sex, neutralising those tokens; another explanation is that these messages were primarily image-based. SpamBayes does not yet do any processing of images, so image-based messages generally have very few tokens and end up in the unsure range (which, for TREC, would mean ham). These three spam genre account for 94%, 92%, and 93% of the false negatives for *tam1*, *tam2*, and *tam3* respectively.

Genre classification of false positives offers fewer clues as to the poor performance. "List" and "newsletter" false positives have the greatest increase across submissions, from 3 (*tam1*) to 32 (*tam2*) to 197 (*tam3*); as above, the most likely explanation for these errors is the temporal distribution of the messages, along with the ham::spam imbalance. If the spam "list" and "newsletter" messages arrived much earlier than the ham messages, then the large number of tokens would start out as strongly spam.

5.3 ROCA and LAM%

A difficulty of comparing filters is that the choice of threshold value includes a subjective judgement about the cost of false positives versus false negatives. To compare filters across all threshold values, Receiver Operating Characteristic (ROC) curves [13] were calculated for each run, and the area under the ROC curve was calculated. The area under the ROC curve is a cumulative measure of the effectiveness of the filter over all possible threshold values; for consistency with false negative and false positive values, the track uses the area above the ROC curve, as a percentage ((1 – ROCA)%). Note that the 'unsure' range does not fit particularly well in ROCA analysis, as the effectiveness is a measurement of a binary classification, and is not able to factor in the 'unsure' range.

Another single-figure result provided in the track is the logistic average misclassification percentage (LAM%) [4], which is the geometric mean of the odds of ham and spam misclassification, converted back to a proportion. Note that neither of these measures impose any relative importance on ham or spam misclassification, and reward equally a fixed-factor improvement in the odds of either. While a single-figure statistic is convenient for comparison purposes, and is certainly preferred by the media and general public, it is somewhat misleading, given that there is a recognised difference in cost

[5] However, the TM corpus had an even higher ham::spam ratio, and did not experience these poor results.

between false positives and false negatives (with the former considered much more expensive). However, until valid data quantifying the relative costs is available, determining a single-figure statistic remains difficult.

The *tam1* submission had a ROCA value of 0.172 (the range of all submissions[6] was 0.051 to 33.079; almost all were under 5.0), and the *tam2* submission had a ROCA value of 0.209. These are fairly satisfactory results, and more indicative of actual performance than the raw false positive and false negative results; for example, *tam1* was the 10th most successful filter in terms of ROCA, but 3rd in terms of false positives, and 24th in terms of false negatives. This suggests that the 0.6 threshold was not ideal, and should have been lower in order to minimize the total number of errors, although considering the perceived (see above) cost of false positives, this trade-off could be considered worthwhile.

The *tam1* submission had a LAM percentage of 1.07 (the range of all submissions was 0.62 to 35.88; most were under 2.0), and the *tam2* submission had a LAM percentage of 1.10. Most interestingly, these are very similar results (more similar than the ROCA and, particularly, false positive/negative measures), indicating that the primary difference that the *use_bigrams* option and *fpfnunsure* regime made was to shift errors to/from false negatives to false positives. Comparatively, both *tam1* and *tam2* did not do as well with this measure, with *tam2* ranked at 18th; selecting a threshold that minimized total errors rather than false positives would probably have improved this.

5.4 Comparing submissions

Comparing *tam1* and *tam2*, *tam2* does not perform as well as expected. The *tam2* submission added the tiled unigram/bigrams technique and changed to a mistake-based training regime; both of these have given superior results in the majority of previous testing. The percentage of false negatives did decrease between *tam1* and *tam2* (vastly in the case of the unusual SB corpus), but this decrease came at the cost of an increase in false positives in all three corpora; whether this trade-off is acceptable would depend on the relative cost of false positives versus false negatives. The author's hypothesis is that the change to the *fpfnunsure* training regime improved results (particularly reducing the number of false negatives), but that the unigram/bigram tiling technique may have increased the number of false positives (particularly those that would normally fall in the unsure range). This hypothesis is based on previous research, where the tiling technique has increased the number of messages in the unsure range and has not always given superior results in an 'incremental' simulation, rather than on data provided from the TREC simulations themselves.

The shift from *tam2* to *tam3*, which only changed the training regime from *fpfnunsure* to train-to-exhaustion, was expected to (at the cost of vastly increasing the simulation runtime) significantly improve the classification results. The actual results are unclear; with *MrX* there is a slight improvement in the false positive rate at the cost of a much smaller increase in the false negative rate, with the *Public* corpus there is a larger improvement in the false positive rate but at the cost of an even larger increase in the false negative rate, and with the *SB* corpus the change appears to mostly be a shifting of the error rate from mostly false negatives to both false negatives and false positives.

The failure of the train-to-exhaustion technique to deliver improved results (particularly in the comparison of *tam2* and *tam3*, where nothing else was changed) is puzzling. It is possible that the restriction of the training collections to the most recent 1,000 ham and 1,000 spam had a detrimental effect, but the simple enforced balance of the training data was expected to improve results.

[6] Both ROCA and LAM% refer to aggregate scores over all corpora (including the *SB* corpus, for which SpamBayes results were very poor). *tam3* and *tam4* are not included, as they did not complete on all corpora.

Corpus	FP %	FN %	Total Messages	Ham::Spam Ratio
MrX	0.2774%	2.614%	49086	0.23::1
SB	0.1446%	37.90%	7006	8.04::1
Public *(Full)*	0.2621%	4.270%	92189	0.75::1

Table 2: Simple results summary for *tam1*.

Corpus	FP %	FN %	Total Messages	Ham::Spam Ratio
MrX	1.448%	1.784%	49086	0.23::1
SB	1.054%	10.24%	7006	8.04::1
Public *(Full)*	0.8550%	1.467%	92189	0.75::1

Table 3: Simple results summary for *tam2*.

Corpus	FP %	FN %	Total Messages	Ham::Spam Ratio
MrX	0.8239%	1.883%	49086	0.23::1
SB	6.708%	3.774%	7006	8.04::1
Public *(Full)*	0.2239%	4.667%	92189	0.75::1

Table 4: Simple results summary for *tam3*.

Corpus	FP %	FN %	Total Messages	Ham::Spam Ratio
MrX	0.9720%	0.8944	49086	0.23::1
SB	4.075%	6.602%	7006	8.04::1

Table 5: Simple results summary for *tam4*.

Comparing *tam3* to *tam4*, where many additional tokenization options were enabled, the *MrX* corpus had improved results (a small increase in the false positive rate, but a near halving of the false negative rate), while the unusual *SB* corpus essentially swapped the false positive and false negative rates (overall, a success, since a false positive is considerably worse that a false negative). It is odd that the *tam4* changes had opposite effects in these two corpora, decreasing false negatives with *MrX*, but decreasing false positives with *SB*; unfortunately the size of the public corpus, combined with the lethargy of the *tam3* and *tam4* submissions, meant that results for *tam4* were unable to be obtained.

5.5 Training data ham::spam balance

SpamBayes does not perform well when the ratio of trained ham to trained spam is very large or very small (some users have reported ratios as high as 1::300 and 1::500!). The expectation, therefore, is that the percentages of false positives and false negatives would be lowest where the ham::spam ratio is closest to 1, and as the ratio approached zero and infinity, the false positive and false negative rates would increase. The data is not conclusive (particularly since there are only a small number of corpora represented), but, as shown in Figure 1, there is some evidence of this trend. The shape of the false positive data points does form a very shallow parabola, with the low centre point around the 1::1 point. The false negative data is less clear – there are some fairly high rates near 1::1, and fairly low rates towards 0.1, but many of the data points would fit into a curve of a similar shape to that of the false positive data points.

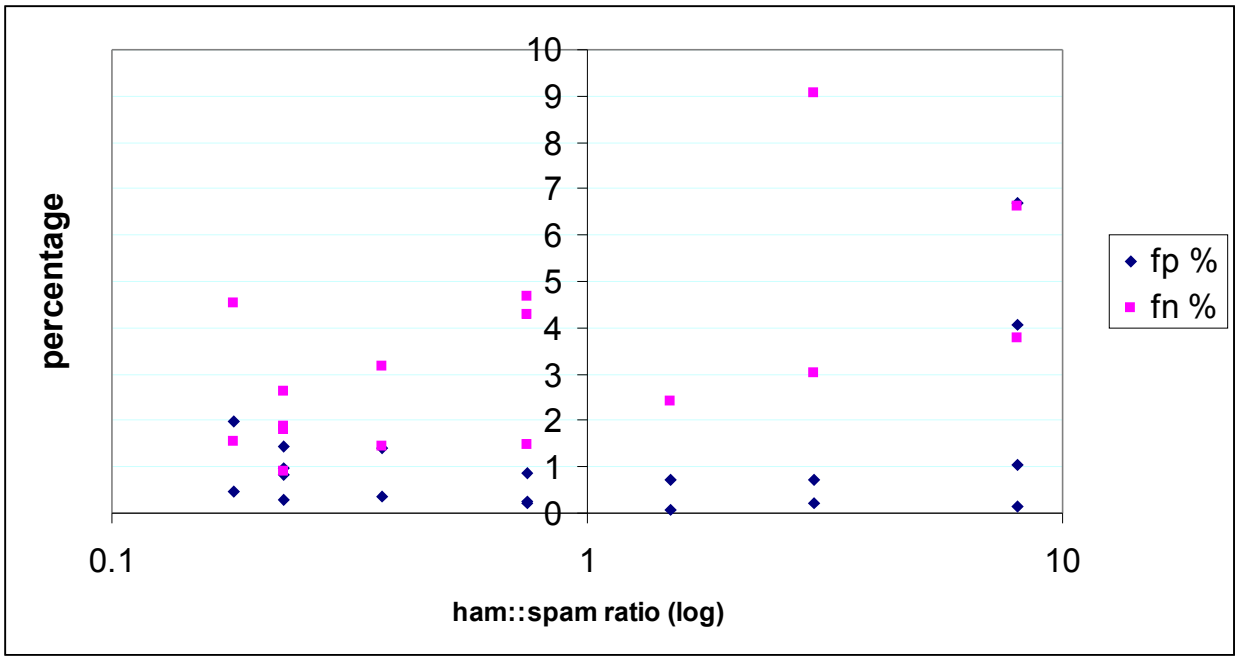

Figure 1: False positive and false negative rates across varying ham::spam ratios[7].

6 Conclusion

For the majority of the corpora tested as part of the TREC 2005 spam track (including the variants of the public corpus, not discussed here), SpamBayes performed better than the author expected, given that a somewhat arbitrary value was chosen for the ham and spam thresholds, in order to remove the unsure range. Anecdotal reports suggest that a false negative rate around 1%, a false positive rate of around 0%, and an unsure rate of around 1% to 5% is common with practical SpamBayes use. Dividing the 1%-5% unsure range (as above) into the false negative and false positive ranges results in an expectation of around a 2% to 4% false negative range, and a 0% to 2% false positive range. The results obtained match this fairly closely, with typical false positive values around 1% and typical false negative values around 2%-4% (with the exception of the SB corpus, where results are abysmal). These results strengthen those obtained in the research of the SpamBayes development team, with their own cross-validation and incremental testing simulations [3, 8, 9].

However, the comparison of the four submissions is extremely unclear. Those techniques that have, both in simulation and in practice, performed well in the past, have failed to deliver improved results in the TREC simulations. The reasons for this failure are not apparent; it is possible that there will not be enough data available from the TREC simulations for the reasons to be conclusively determined. From limited analysis, it does appear that the expected performance decrease as the trained ham::trained spam ratio diverges from 1::1 is observed in the TREC simulations.

Acknowledgements

As an open-source project, SpamBayes is developed by a team of volunteers, headed by Tim Peters, and all credit for SpamBayes should be directed to them; however, the author assumes responsibility for any errors in this paper. The 2005 TREC spam track public corpus SpamBayes runs were executed on Massey University's helix cluster [14]; many

[7] This figure includes data for the two private corpora (*tam1*, *tam2*, *tam3*, *tam4*), and the public corpus (*tam1*, *tam2*) including the four additional variations. One data point (for the SB corpus) is outside of the range of the graph.

thanks to the Institute of Information and Mathematical Sciences for providing this resource. Finally, many thanks to Gordon Cormack (and his team) for their assistance in massaging the SpamBayes submission so that it would complete the private corpora runs sufficiently fast, and for patiently waiting for the *tam3* and *tam4* runs to complete.

References

1. SpamBayes-Development-Team. *SpamBayes: Bayesian anti-spam classifier written in Python*. [Website] 2003 [cited 2003 10 October]; Available from: http://spambayes.org.
2. Graham, P. *A Plan for Spam*. [Website] 2002 January 2003 [cited 2004 April 12]; Available from: http://www.paulgraham.com/spam.html.
3. Meyer, T.A. and B. Whateley. *SpamBayes: Effective open-source, Bayesian based, email classification system*. in *Conference on Email and Spam*. 2004. Mountain View, California.
4. Cormack, G. and T. Lynam. *TREC 2005 Spam Track Overview*. in *Text Retrieval Conference*. 2005. United States.
5. Robinson, G., *A Statistical Approach to the Spam Problem*. Linux Journal, 2003. **107**.
6. Graham-Cumming, J. *POPFile*. [Website] 2005 [cited 2006 25th January]; Available from: http://popfile.sourceforge.net/cgi-bin/wiki.pl.
7. Assis, F., et al. *CRM114 versus Mr. X*. in *Text Retrieval Conference (Spam Track)*. 2005. United States.
8. Meyer, T.A. *SpamBayes Exhaustion*. [Website] 2004 10th Feburary 2004 [cited 2005 20th October]; Available from: http://www.massey.ac.nz/~tameyer/research/spambayes/exhaustion.html.
9. Popiel, A. *Alex's spambayes testing*. [Website] 2003 25th December 2003 [cited 2006 25th January]; Available from: http://www.wolfskeep.com/~popiel/spambayes/.
10. Peters, T. *The first trustworthy <wink> GBayes results*. [Website] 2002 2 September [cited 2004 April 12]; Available from: http://mail.python.org/pipermail/python-dev/2002-September/028513.html.
11. Robinson, G. *Instructions for Training to Exhaustion*. [Website] 2004 [cited 2004 April 12]; Available from: http://garyrob.blogs.com/garys_longer_rants/2004/02/instructions_fo.html.
12. Ng, H.T., W.B. Goh, and K.L. Low. *Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization*. in *20th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1997. Philadelphia, Pennsylvania, USA.
13. Cormack, G. and T. Lynam. *A Study of Supervised Spam Detection Applied to Eight Months of Personal Email*. [Website] 2004 1st July 2004 [cited 2006 25th January]; Available from: http://plg.uwaterloo.ca/~gvcormac/spamcormack.html.
14. Barczak, A.L.C., C. Messom, and M.J. Johnson, *Performance Characteristics of a Cost-Effective Medium-Sized Beowulf Cluster Supercomputer*. Research Letters in the Information and Mathematical Sciences (RLIMS), 2003.