Combining Bayesian and Rule Score Learning: Automated Tuning for SpamAssassin

Alexander K. Seewald
Austrian Research Institute for Artificial Intelligence
Freyung 6/6, A-1010 Vienna, Austria
alexsee@oefai.at, alex@seewald.at

Abstract

Spam (Unsolicited Bulk Email), has become a global problem of high economic impact. In this paper, we discuss an applied approach to automatically adjust all parameters of a hybrid spam recognition system, SpamAssassin. We investigate both learning of rule scores and selective training of the integrated bayesian spam filter. We report competitive results concerning ham misclassification rate (i.e. legal mails misclassified as spam) which are comparable to human accuracy, and are able to significantly improve the spam misclassification rate (i.e. spam misclassified as legal mails) by a factor of around twenty versus SA with default scores. A variant of Total Cost Ratio also shows the same trend. The results of the training process can be transformed into a SA preference file. Also, the training process utilizes mainly non-spam mailboxes which are easily obtained.

1. Introduction

Unsolicited Bulk Email (Spam) has become a problem of global impact. For example, according to a study undertaken for the European Commission, Internet subscribers worldwide are wasting an estimated 10 billion euro per year just in connection costs due to Spam [11]. Economic impact is only part of the problem - waste of time, resources and the gradual erosion of trust in Email communications should also be considered. Within the scientific community these effects are felt strongly. For example, at our institute the proportion of Spam now exceeds 94%. This means that for every nonspam mail, we receive around 17-25 spam mails.

Several approaches exist to deal with spam [6]. Filtering approaches based on message features are most widely used and also the only workable solution at present. In fact, most email clients already allow their users to manually build email filters. However, the manual approach is time-consuming and much expertise is needed to create useful filters from scratch. Another option for filtering is to collect large samples of spam and nonspam (=ham) and train a classifier on it. This has been proposed e.g. by [3] and works surprisingly well even with simple statistical classifiers such as Naive Bayes. Most state-of-the-art spam filters now include similar learning systems, e.g. SpamBayes (`spambayes.org`), CRM114 (`crm114.sourceforge.net`) and SpamAssassin (`www.spamassassin.org`).

SpamAssassin is an open-source hybrid spam mail filter incorporating a state-of-the-art bayesian learner as well as a set of 500+ human-created heuristic rules for spam recognition. SpamAssassin thus incorporates background knowledge on spam in the form of heuristic rules as well as a bayesian classification system. Contrary to a pure bayesian approach, this makes adapting the system harder – it not clear when to adapt the scores, train the bayesian filter, or both – but gives advantages in terms of out-of-the-box availability, reliability and diversity. It may even be argued that theses two levels of training correspond to a multi-view learning system and that its diverse representation makes the full system more robust. However, the latter investigation is beyond the scope of this paper.

During the last six months we have successfully adapted SpamAssassin to the author's mailbox, reducing the proportion of spam from 94% to 4% with a false positive rate of <0.2%. The average user is not as lucky, since the default values of SA do not work as well and delete less than half of incoming spam (see also Table 2, row SA), which reduces the proportion of spam from 94% to a mere 89%. Even training the bayesian filter properly alone is not sufficient to reach a 1:1 spam/nonspam ratio - it is also necessary to adjust score weights.

So we have reformulated the problem of adjusting score weights and bayesian learning for SpamAssassin as a learning problem. Given that collection and especially verification of spam is time-consuming and error-prone (error rates as high as 0.16% have been reported even under optimal conditions by [13]), while ham-only collections are easily obtainable, we have based our training on pure nonspam (=ham) collections from other users at our institute. Our approach can either be viewed as an application of machine learning techniques to the problem of optimal score assignment and bayesian learning within SpamAssassin from a mainly empirical viewpoint; or as an investigation into approaches to multi-view learning (one view is the bayesian learner; another is the score assignment) within SpamAssassin.

2. Evaluation measures

The effectiveness of spam filtering systems is measured in terms of correct and wrong decisions. For simplicity, we restrict ourselves to two classes: ham (+ aka nonspam) and spam (- aka Unsolicited Bulk EMail). For a given number of documents, the classification of a spam filtering system can be summarized in a contingency table, see Table 1. The abbreviations have the following meanings: **T**rue**P**ositives and **T**rue**N**egatives are the ham resp. spam mails which are correctly predicted by the system. **F**alse**P**ositives are errors where spam mails have been misclassified as ham, and **F**alse**N**egatives are ham mails which have been misclassified as spam. Notice that our naming conventions are opposite to those normally used. This is because we defined ham (+) as positive class since this seemed to make more sense.

Traditional text categorization approaches are usually evaluated in terms of recall, precision and F-measure, i.e.

$$r = \frac{tp}{tp + fn},\ p = \frac{tp}{tp + fp},\ F = \frac{2rp}{r + p}$$

Other works make use of standard machine learning metrics, e.g. accuracy and error rate. We mainly report error rates here because it makes comparison easier: for example, the difference between 99.8% and 99.9% accuracy is actually a reduction of error by factor of two. This is more easily visible by comparing 0.2% and 0.1%. For ham-only datasets, the error rate is equivalent to the proportion of ham that would be misclassified of spam – this should be on the order of human error (around 0.16%) for the best systems. For spam-only datasets, this is equivalent to the proportion of spam which gets through, i.e. which is not recognized by the system, and should also be quite small ($<$5%). For mixed datasets, i.e. those including ham and spam mails, we use customary symmetric unit error costs to compute the error rate instead of asymmetric misclassification costs which would be more appropriate but harder to interpret. However, the last is only reported for our first exploratory experiment.

We will introduce two other measures: the ham error (misclassification) rate and the spam error (misclassification) rate. These are simply the proportion of erroneous mails in each category; i.e. the estimated probability of a misclassification from ham to spam (ham_e) or vice versa ($spam_e$), under the assumption that these are independent.

$$ham_e = \frac{fn}{fn + tp}, spam_e = \frac{fp}{fp + tn}$$

Since we evaluate these two terms on ham-only and spam-only datasets, it makes sense to report them as simple error rates on the appropriate datasets.

It is also desirable to compute a single value for comparison between approaches. For this, we chose a variant of Total Cost Ratio (TCR) with message specific costs inspired by [7]. They introduced the following message-dependent costs for misclassifying ham mails as spam, which were used for training purposes – we will use them only for evaluation.

- **Sensitive personal messages** = 1000
  Misclassifying this type of email could be potentially most harmful, so the highest cost (equivalent to correctly recognizing 1000 spam mails) is justified.
- **Business related messages** = 500
  The cost of losing business related emails is hard to establish, especially for a research institute. We have opted to put all emails here which are related to administrative or research projects. If the mail was considered urgent or sufficiently important, we upgraded it into the first category.
- **e-commerce related messages** = 100
  These messages include registration, order, shipment confirmation et al. However, we did not encounter any of these mails as an error.
- **mailing lists / discussion forums** = 50
  These messages are related to less important mailing lists and discussion forums, other information sources etc. We have also chosen to put forwarded stereotyped funny mails, chain letters, get rich schemes and the like

	ham (+)	spam (-)
ham (+)	tp	fp
spam (-)	fn	tn

Table 1. A set of classification decisions for mails belonging to either ham (+) or spam (-) category can be succintly represented by a contingency table. The rows represent the decision/prediction of the system while the columns represent the true category.

into this category, since they would most probably not be missed. Funny mails which look sufficiently original are placed into the *sensitive personal messages* category to err on the side of caution.

- **promotional offers** = 25
  This category concerns promotional offers, commercial offers, additional unsolicited information from companies which we are in contact etc. However, we did not encounter any of these mails as an error.

We have decided manually for each false negative error in which category it belongs and summed the costs accordingly. The costs for misclassifying a spam mail as ham is assumed to be 1, as is the cost of removing a spam mail manually. We will report results in terms of a variant of Total Cost Ratio for message dependent misclassification costs, i.e.

$$TCR = \frac{fp + tn}{\sum_{x \epsilon fn} C(x) + fp}$$

The TCR measures how effective the system performs. We compare the effort to delete all spam mails manually (fp+tn) versus deleting just the ones which were misclassified (fp) plus recovering from the cost of misclassified ham mails ($\sum_{x \epsilon fn} C(x)$). The cost of each misclassified ham mail $C(x)$ is computed according to above categories. Greater TCR means better performance; a TCR$<$1 means that the system performs worse than manual deletion of all offending mails, under the given assumption on action costs.

Since our set of mailboxes for training consist only of ham mails, we need an additional way to estimate spam error rate. We have chosen to estimate spam error rate of these systems by evaluating them on the spam mails of as_test. This gives a rough estimate on how well spam mails will be recognized. We scaled the number of true spams to 17 times the number of true hams before computing TCR to best approximate the real-life situation at our institute.

3. Experimental setup

We first created our own mailbox of 1176 hams and 1611 spam mails from past mails received at our own mail account. This set is denoted as as_train and was arbitrarily sampled to a somewhat balanced class distribution. We also collected 109 hams and 1011 spam mails for evaluation purposes. These are more current and the class distribution is also more realistic, but the small number of hams makes them unsuitable for training. We denote this set as as_test, and will use it later to estimate the spam error rate for our models.

We also collected ham-only datasets from our colleagues: 475 (*hx*), 2163 (*ix*) and 363 (*ux*) mails, respectively. These datasets were taken from archived mail files; however, during the course of our investigations we found four spam mails wrongly classified as ham, although they had been looked through at least once. At this point we removed all four wrongly classified hams and had to repeat all previous experiments. From this involuntary experiment we can roughly infer an empirical error estimate of 0.12±0.11% for overlooking minority mails in a large mail collection during manual investigation. This agrees reasonably well with the human error rate of 0.16% measured in [13].

	as_train	as_test	hx	ix	ux
SA	36.9%	58.5%	0.2%	0.3%	0.8%
SAnb	*7.79%	*8.6%	0.4%	0.1%	1.4%
Log	0.65%	*1.7%	2.7%	1.8%	1.1%
SMO	0.68%	*1.2%	2.7%	0.9%	1.1%
MLR	*0.65%	*1.9%	2.7%	1.3%	1.1%
J48	0.61%	*1.6%	2.9%	0.9%	1.1%

Table 2. First experiments: two-fold cv on as_train (except SA and SAnb); test of the trained model on as_test plus *hx*, *ix*, *ux*. All reported results are ham error rates in percent. A star (*) indicates that no true ham mails were misclassified (fn=0).

We used version 2.63 of SpamAssassin in all our experiments. SA is a hybrid classifier with a set of 500+ heuristic rules, and a bayesian learner. Each heuristic rule has a weight (score) attached. Rule matching is binary and based on perl regular expression matching. The sum over all matching rules is the full score for the mail. A user-definable threshold is used to determine if a mail is to be classified as spam or ham. The bayesian learner is integrated into the ruleset as a small set of pseudo-rules (e.g. BAYES_00 matches when bayes spam probability is between 0% and 5% etc.), also with an attached user-definable score. A genetic algorithm has been used to optimize the scores for all the rules and the bayesian pseudo-rules on a large corpus of spam and ham mails. These defaults were used for the SA and SAnb runs. The default threshold of 5.0 was used, and autolearn was switched off for these experiments.

The bayesian learner for SpamAssassin was taken from a recent model which has been sporadically trained for six months. It was restored at the beginning of each experimental run. Bayesian learning was prevented during the experiments, except where required by the experiment.

Our meta-datasets consist of the mail classification (spam or ham) and the set of SA rules (including bayes pseudo-rules) which match the corresponding mail. Each rule is represented by a binary attribute.

Four classifiers were chosen for the initial comparison: MLR (*Multi-Response Linear Regression*) which learns a linear regression function to predict class membership; Logistic regression, a more elaborate approach than MLR [2]; SMO (a support vector machine classifier [8]); and J48 (a C4.5 decision tree learner clone, [9]). All classifiers were taken from the WEKA machine learning workbench (`www.cs.waikato.ac.nz/~ml/weka`). Table 2 shows the results. We report SA's performance without (SA) and with (SAnb) our recent bayesian model and default scores for all rules. Both of these are intended as baseline comparison. We also report error rates for as_train as estimated by two-fold crossvalidation, and the performance of the as_train full training set model on as_test, as well as *hx*, *ix* and *ux*. Since the latter three are ham-only mailboxes, what we report here is essentially an estimate for the probability of misclassifying a ham-mail as spam – the ham error rate. The number of ham mails misclassified was usually quite low for as_train and as_test – a star (*) denotes where it was zero.

We chose MLR for further investigation because of these three reasons: MLR is the only classifier to never misclassify any ham in as_train or as_test; is quite competitive to the other learners on *hx*, *ix* and *ux*; and it is also the only classifier which generates a model that can directly be transformed into a weighted score file for SA, facilitating future deployment of the system. The last reason is probably the most important one to us. It should also be noted that SA with default rule scores – both with and without our pre-trained bayesian model – already performs quite well on *hx*, *ix* and *ux* without any further adaptation. As we shall see later, the difference lies in the spam misclassification rate rather than the ham misclassification rate.

In this simple setting, the performance on *hx*, *ix* and *ux* is not yet satisfactory. Clearly it is necessary to train on part of the specific hams which we try to recognize. So we have opted for a 1:1 split for all hams into ham-F1 and ham-F2 (using odd and even-numbered mails), separately for *hx*, *ix* and *ux*. One of these is used for training, the other for test and afterwards they are swapped (two runs), reminiscent of a two-fold crossvalidation. The main difference is that we reuse parts of the training set for economical reasons, e.g. we use the same spams from as_train in both folds which would not be possible with a normal CV. A two-fold CV gives us more opportunity to look at each error in turn. Lastly, it is also a harder test for our approach, which makes sense since we want to apply this system in practice.

We investigated various different approaches to training which we called V0-V6. Testing is always on the other ham file (i.e. test on ham-F1, if ham-F2 is used for training; and vice versa). The part of the ham mailbox used for training is denoted ham-Fx here.

- V0: This is the baseline model which was trained only on as_train. No training data from ham-Fx is used at all.

- V1: Train on ham-Fx and all spams from as_train. No hams from as_train are used for training. The intention is to focus the training on the important hams. Since the obtained mailboxes are ham-only, at least a source of spam mails has to be added to enable two-class learning.

- V2: Train on ham-Fx and as_train (ham+spam). The intention of this is to use the hams from as_train as additional data, since two of our mailboxes are quite small and may benefit.

- V3: Learn all ham-Fx via NB learner; then train on ham-Fx and all spams from as_train. The idea here is to use bayesian learning in a simple setting: just learn all hams from ham-Fx and see whether the bayesian learner is able to generalize to the unseen examples. This is roughly equivalent to training method Train Every Thing (TEFT) from [13].

- V4: Train on ham-Fx and as_train (ham+spam), afterwards adapt the training set probability threshold from the default of 0.5 so that false negative training set errors (i.e. hams misclassified as spams) disappear. If misclassified ham mails with probability of 1.0 appear, these remain training set errors. This needs a learner which can output confidence values for the prediction - fortunately, this is the case for MLR. V4 was motivated by a common approach to cost-sensitive learning. This approach is also easy to implement within SA by changing the score threshold.

- V5: Train on ham-Fx and as_train (ham+spam), learn all misclassified hams via bayesian learner; then train on ham-Fx and as_train (ham+spam) again and evaluate. This is roughly equivalent to Train Only Errors (TOE) from [13], with the restriction that only one type of error is learned.

- V6: Train on ham-Fx and as_train (ham+spam), learn all training set errors via NB learner; then retrain on ham-Fx and as_train (ham+spam) and evaluate. This is roughly equivalent to TOE from [13].

4. Results

Table 3 shows the ham error rates and standard deviations for approaches V0-V6. We can directly interpret the error rate as estimated probability that a given ham will be misclassified as spam for the respective mailbox. As expected, there is less variance within the largest mailbox *ix*, and more within the smaller ones. In most cases V1-V3 fail to perform much better than V0 (except V2 on *ux*, which is

	hx	ix	ux
V0	2.53%	1.20%	1.10%
V1	4.63±2.38%	0.79±0.07%	1.65±2.34%
V2	2.53±0.60%	0.79±0.07%	0.55±0.78%
V3	5.68±2.08%	1.16±0.33%	2.20±0.78%
V4	0.63±0.30%	0.14±0.07%	0.55±0.78%
V5	0.21±0.30%	0.32±0.07%	0.55±0.78%
V6	0.21±0.30%	0.23±0.07%	1.10±0.00%

Table 3. Average and standard deviation of ham error rate is computed over the two-fold crossvalidation, for each mailbox separately.

	hx	ix	ux
SA		64.7%	
SAnb		9.5%	
V4	7.3%	45.5%	2.1%
V5	3.2%	4.9%	2.1%
V6	2.1%	2.7%	1.5%

Table 4. This table shows the spam error rate on all spams from as_test, for SA, SAnb and V4-V6.

	hx	ix	ux
SA	1.53	1.39	1.12
SAnb	9.31	7.29	2.02
V4	10.92	2.07	5.46
V5	26.18	10.50	5.46
V6	36.42	17.25	2.92

Table 5. This shows the Total Cost Ratio for SA, SAnb and V4-V6.

a special case – see below). Generally, V2 performs better than V1 except for the largest mailbox *ix* where it performs equally well, which confirms our suspicion that adding ham mails from as_train is beneficial for smaller datasets.

The worse performance of V3 took us by surprise. However, it can be explained as follows with the benefit of hindsight. Training all hams prior to MLR learning means that most of them will match a bayes-pseudo rule. Since no other hams are present in the training set, this means that this rule will dominate the MLR model, reducing the complexity of the model. It now seems as if the bayesian learner is then unable to compensate for this reduction in MLR model complexity by a better generalization to the test set. This results in an overall reduction in performance. Consistent with this assumption, the approaches V5 and V6 which disturb the bayesian model less work much better.

V4 works quite well and for *ix* even has the best model overall. However, by adapting the threshold we change both fp and fn at the same time. While we are in most cases able to reduce the ham error (fn) by changing the threshold for classification from the default value of 0.5 (we could always reduce it to zero by changing it to 1.0 – which gives us the trivial all-ham model), this is also accompanied with an increase in the misclassification rate for spam (fp) – for *ix*, the spam error rate is 45.5%! (see Table 4)

V5 and V6 give competitive results to V4. The worse performance on *ux* is likely to be due to insufficient training data – *ux* is the smallest dataset. *ux* is also the only mailbox where V2 is already competitive, since no training set errors are found – therefore neither V4 nor V5 can improve on the model, while V6 fails to improve it. The last may be due to overfitting. Notice that the error of 0.6% corresponds to only two misclassified hams for *ux*, and even in this case TCR is greater than 1 (see Table 5).

V5 and V6 follow the premise to change the bayesian model only where necessary. This approach of showing only those mails to the bayesian learner where information of the other heuristic rules is insufficient to determine its categorization correctly seems to work well in practice. Conceptually, we may also describe this as related to multi-view learning - if the information in one view (=the heuristic rules) is insufficient, the examples are enhanced by adapting their second view representation (=the bayesian model) through adding additional information, which again changes the first view representation to allow a better model there.

V5/6 and the thresholding approach from V4 cannot be combined, because both of these approaches practically ensure that fn on training is zero afterwards, leaving no handle for the other method. V6 achieves near-human-level performance concerning false negative errors, and also yields the lowest spam error rate on two out of three mailboxes (all but the smallest one).

In Table 4, we see that SpamAssassin with our pretrained bayesian model is already quite good, but an additional improvement by factor four is still possible. SpamAssassin without bayesian model performs much worse, so a well-trained bayesian model seems to be essential for significant reduction in visible spam mail volume.

Now we will take a short look at the three best systems V4-V6 plus SA and SAnb in Table 5. The Total Cost Ratio has been computed as previously defined, i.e. estimating the ham error rate via two-fold CV on the spam as described previously and estimating the spam error rate on spams from as_test averaged over the trained two-fold CV models. The cost has been estimated for each misclassified ham mail separately by assigning it to the most appropriate cost category from Experimental Setup – for more details on this, see Discussion. Our high proportion of spam at 94% (around 17:1) was also accounted for by scaling the number of true negative and false positive values accordingly.

As can be seen, the Total Cost Ratio is usually above 1.0 as it should be, even for the simplest baseline SA. Generally, TCR increases from top to bottom, with few exceptions, so we conclude that our models from V4-V6 are improving and we propose approach V6 for further experiments, followed by V5 which is slightly worse but seemingly less risky. We will thus deploy V6 to *hx* and *ix*, plus V5/V4/V2 (which are all the same model for lack of training set errors) to *ux*. Multiple runs of retraining similar to V6 may improve performance further, we leave this for the future.

5. Discussion of False Negatives

On *hx*, only a single ham mail appears as error in the experiments V4-V6. This is a newsletter on Italian Cuisine from About.com. The newsletter only appears twice in the test set and is correctly recognized as ham once. When training on the full data, both instances of the newsletter are correctly recognized as ham. We have assigned a cost of 50 to this misclassified mail after obtaining feedback from the mailbox's user.

On *ix*, multiple diverse mails appear as errors. There is no clear pattern to the errors. If we take a close look at the five mails from V6, we see that most of them are from an ongoing email conversation. Only one of the mails is unique; the others come from people who appear 3, 7, 9 and 60 times in the whole mailbox. We can assume that the auto whitelist feature (which averages mail scores over a longer conversation) is able to smooth one error in 7, 9 and 60; and probably even one error in 3. The diverse nature of this dataset makes further optimization challenging. A training on the whole dataset indicates that a consistent model cannot yet be learned: the person from which mail appears 3x in the whole mailbox is still once misclassified. We should note that the mentioned mail is completely empty, with a large binary attachment which is a microsoft executable, and additionally has some twice encapsulated mime structure. As a (supposedly?) funny mail we have assigned it a cost of 50 (after checking with the mailbox's user), along with three other similarily misclassified mails: forwarded chain letters and other mails which are not likely to be strongly missed. Generally, mails from freemail systems (GMX, Hotmail, Yahoo) seem more likely to end up misclassified. We also assigned a cost of 500 to each of the four business mails which were misclassified.

On *ux*, two mails from the same person appear as errors in experiments V4-V5. Almost a dozen rules in SpamAssassin match both mails; and both of them are in the same fold, so they cannot be learned in our two-fold CV reminiscent setup. In V6, two other mails also appear as errors. All these errors disappear when trained on the whole data, indicating that a consistent model can be learned - however, the training data within the CV seems to be too small to achieve this. All erroneous mails have been assigned a category of business mail and cost of 500.

6. Related Research

[5] reports a cost-sensitive evaluation of current approaches to unsolicited bulk email categorization. A large set of papers which apply machine learning techniques to spam categorization is mentioned. However, all of these papers are built on the term vector space (bag of words) input and do not use a simple binary representation of matching rules like our approach.

[4] reports a comparative evaluation of several machine learning algorithms on the text of messages and also a set of 9 heuristics. The reported improvement due to the use of heuristics is modest. Our approach uses 500+ heuristics from SpamAssassin which may explain why the observed improvement in our case is higher. On the other hand, no direct content features such as word or phrase occurrence are available to our system. It should be noted that a well-trained bayesian model is still at the core of a good spam filter even in our case, so heuristics alone are in any case insufficient.

[10] reports that a scheme for combining classifiers known as stacked generalization improves the performance of spam categorizers. While the use of ensembles is similar in spirit to SpamAssassin, their ensemble is a set of classifiers trained on bag-of-word representations of mails while SA's ensemble is one of human-created rules and a bayesian learner which is expected to be more diverse than their ensemble which differs only in the choice of learning algorithms at the base level. Since diversity is one of the key elements for successful ensemble learning, it is not unexpected that our approach is competitive.

Our results disagree with [1] who found that current spam filters are not suitable for deleting messaged classified as spam. However, since then things have changed: For once, our spam mails account for 94% of mails, while their corpus had only 16.6% spam mails. It seems that the last three years have seen an exponential increase in spam volume. Also, they used Total Cost Ratio with a simple cost of 1000 for all false negative errors. In reality, the cost for losing a ham mails varies with the type of mail, which has been addressed by [7] with a variable cost measure. We believe this to yield a more accurate picture of spam filtering than a constant cost approach and have therefore decided to use it for our evaluation.

[7] propose a SVM classifier which explicitly uses the per-message cost values during training, or for post-

processing. They found that their use during training was most helpful. We on the other hand have found the given variable cost measures also very useful for realistic evaluation of ham misclassification errors.

[12] has described an approach to use Genetic Algorithm techniques to optimize rule scores within SpamAssassin. Their approach differs from ours in that they do not explicitly trigger bayesian retraining, but instead assume that bayesian model to be fixed. Our approaches V5 and V6 aim to mesh rule score training and bayesian training within a simple setting reminiscent of multi-view learning.

7. Conclusion

We have investigated an application of machine learning techniques for spam mail classification. Contrary to other approaches, we have learned from a representation of mails as a set of heuristic rules (partially human-created, partially pseudo-rules which corresponds to the output from bayesian learner) with binary matching; and not from term vector space (bag-of-word) representations. This efficient representation has allowed us to learn models which are as accurate as state-of-the-art systems in classifying spam, and also offer a competitive false positive rate. The given system would be able to reduce the proportion of spam mails from 94% to around 24% for our institute and the ham mailboxes which we considered.

Ham misclassification rates as low as 0.1%-0.3% and overall accuracies in excess of 95% were achieved. A Total Cost Ratio of up to 36.42 was observed, indicating that the system is at least an order of magnitude more efficient than manual deletion of all spam mails. It compares well to statistics for single users (0.1% ham error and 99% overall accuracy, [12]) and to human error rate in classifying spam (0.16% according to [13]; 0.12 ± 0.11 according to our involuntary experiment).

The excellent results are even more surprising because we did not use cost-sensitive learning, and used a two-fold crossvalidation instead of the ubiquitous ten-fold CV used in other papers. This translates into a harder task for learning, since the amount of training data is reduced by factor 1.8.

8. Future Work

We intend to apply this approach to classify harder ham mails, e.g. the public collection by SpamAssassin. Although preliminary experiments look somewhat promising, multiple cycles of learning may be necessary in that case.

We will deploy the final system from this approach to our colleagues who contributed the mailboxes and aim to deploy it system-wide at our institute for further practical evaluation. We will also investigate the feasibility of a single bayes model for the whole institute.

As a long-term goal, we will also aim investigate whether it is possible to transfer some expertise across to another institution with completely different spam mail corpora to speed up the first step: training a well-performing bayesian model.

9. Acknowledgements

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. We want to thank our anonymous colleagues for contributing the ham collections.

References

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. and Spyropoulos, C.D. An Evaluation of Naive Bayesian Anti-Spam Filtering. In Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, p. 9–17. Potamias, G., Moustakis, V., van Someren, M.n (eds.), Barcelona, Spain, 2000.

[2] le Cessie, S. and van Houwelingen, J.C. Ridge Estimators in Logistic Regression. Applied Statistics, Vol. 41, No. 1, pp. 191-201, 1997.

[3] Graham, Paul. A Plan For Spam. `www.paulgraham.com/spam.html`, 2003.

[4] Gómez Hidalgo, J.M., Maña López, M. and Puertas Sanz, E. Combining Text and Heuristics for Cost-Sensitive Spam Filtering, Proceedings of the Fourth Computational Natural Language Learning Workshop, CoNLL-2000, Association for Computational Linguistics, Lisbon, Portugal, 2000.

[5] Gómez Hidalgo, Jose M. Evaluating Cost-Sensitive Unsolicited Bulk Email Categorization, Proceedings of SAC-02, 17th ACM Symposium on Applied Computing, 615–620, Madrid, Spain, 2002.

[6] Hoffman, Paul and Crocker, Dave. Unsolicited bulk email: Mechanisms for control. Technical Report UBE-SOL, IMCR-008, Internet Mail Consortium, 1998.

[7] Kolcz, A. and Alspector, J. SVM-based Filtering of E-mail Spam with Content-specific Misclassification Costs, Proceedings of the TextDM'01 Workshop on Text Mining, IEEE International Conference on Data Mining, 2001.

[8] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf, C. Burges, and A. Smola, eds., MIT Press, 1998.

[9] Quinlan, R. C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[10] Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. and Stamatopoulos, P. Stacking Classifiers for Anti-Spam Filtering of E-Mail, Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Morristown, Pittsburgh, US, 2001.

[11] Serge Gauthronet and Etienne Drouard. Unsolicited commercial communications and data protection. Technical report, Commission of the European Communities, 2001.

[12] Sergeant, Matt. Internet Level Spam Detection and SpamAssassin 2.50. 2003 MIT Spam Conference, Cambridge, Massachusetts, U.S.A, 2003. `spamconference.org/proceedings2003.html`

[13] Yerazunis, William S. The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It. 2004 MIT Spam Conference, MIT, Cambridge, Massachusetts.