Batch and On-line Spam Filter Comparison

Gordon V. Cormack
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
gvcormac@uwaterloo.ca

Andrej Bratko
Department of Intelligent Systems
Jozef Stefan Institute
Jamova 39, Ljubljana, Slovenia SI-1000
andrej.bratko@ijs.si

ABSTRACT

In the TREC 2005 Spam Evaluation Track, a number of popular spam filters – all owing their heritage to Graham's *A Plan for Spam* – did quite well. Machine learning techniques reported elsewhere to perform well were hardly represented in the participating filters, and not represented at all in the better results. A non-traditional technique - Prediction by Partial Matching (PPM) – performed exceptionally well, at or near the top of every test.

Are the TREC results an anomaly? Is PPM really the best method for spam filtering? How are these results to be reconciled with others showing that methods like Support Vector Machines (SVM) are superior?

We address these issues by testing implementations of five different classification methods on the TREC public corpus using the online evaluation methodology introduced in TREC. These results are complemented with cross validation experiments, which facilitate a comparison of the methods considered in the study under different evaluation schemes, and also give insight into the nature and utility of the evaluation regimens themselves. For comparison with previously published results, we also conducted cross validation experiments on the Ling-Spam and PU1 datasets.

These tests reveal substantial differences attributable to different test assumptions, in particular batch vs. on-line training and testing, the order of classification, and the method of tokenization. Notwithstanding these differences, the methods that perform well at TREC also perform well using established test methods and corpora. Two previously untested methods – one based on Dynamic Markov Compression and one using logistic regression – compare favorably with competing approaches.

1. INTRODUCTION

The TREC 2005 Spam Evaluation TREC[10] – the largest laboratory evaluation of spam filters to date – models spam filtering as an on-line learning task in which messages are presented to the filter, one at a time, in chronological order. For each message, the filter predicts its class (*spam* or *ham*[1]) by computing a score s which is compared to a fixed but arbitrary threshold t. Immediately after the prediction, the true class is presented to the filter so that it might use this information in future predictions.

Previous studies typically model spam filtering as an off-line (batch) supervised learning task in which a hard binary classifier is induced on a set of labelled training messages and then used to predict the class of each of a set of unlabeled test messages [30]. Many of these studies further abstract each message to a bag or sequence of lexically derived features.

It is difficult to compare studies using these contrasting models; furthermore, it is a challenge to adapt a filter designed for one model to an evaluation method designed for the other. Yet compare we must, if we are to advance our understanding of spam filtering techniques, to determine whether or not the two models yield similar results, and to determine which, if they do not, is more appropriate. Comparison is given particular impetus by the fact that there is little intersection between the well-performing methods evaluated to date using the two models.

At TREC 2005, arguably the best-performing system was based on adaptive data compression methods which have shown mediocre performance for general off-line text classification. The other well-performing systems were derived from currently deployed spam filters using a variety of techniques colloquially referred to as 'Bayesian' – techniques that have not been shown to perform well under the batch model.

Support Vector Machine (SVM) techniques [17] are widely reported to be among the best for supervised learning, text classification, and batch spam filtering. Such techniques are not represented at all at TREC 2005. Logistic Regression (LR) is similarly reported to be effective for classification [19] but has not, to our knowledge, previously been evaluated[2] as a spam filtering technique.

Our objective is to investigate the differences between the two models, the test methods for each, and the transferability of filters and results between them. To this end we prepare reference implementations of five well-performing filters and evaluate them using the two models and several public corpora. We compare the results to each other as well

[1]non-spam

[2]At this conference, Goodman [14] presents an adaptive logistic regression method and results on the TREC corpus which compare favourably to our logistic regression method; these results are reproduced in section 9.

as those in the literature based on the same methods and corpora.

Two of our reference implementations embody methods whose use for spam filtering has not been reported in the literature. One, based on the adaptive Dynamic Markov Compression method [9, 4], was inspired by the compression-based methods that performed well at TREC [5] but was implemented independently within the context of this study. Two implementations previously tested at TREC are also used: PPM, the above-mentioned compression-based method, and Bogofilter [25], an established open-source filter. Our final two reference implementations use established implementations of well-regarded machine-learning techniques: support vector machines [17] and logistic regression [19]; the latter, to our knowledge, has not previously been evaluated as a technique for spam filtering.

Our results show that all the reference implementations perform well overall in comparison to published results, using both batch and on-line tests. Substantial differences may be attributed to the nature of training and testing, message ordering and tokenization.

2. BATCH AND ON-LINE MODELS

The batch model requires that a training set and a test set be identified prior to testing; these sets are assumed to be random samples of a common source population – the population of email messages to which the filter might be applied. The validity of the model is limited by the fact that, in reality, there exists no such common population; the classifier must be induced from existing messages and applied to future ones. If one assumes that the characteristics of messages do not change much with time, one may consider the population of existing messages to be a proxy for those yet to come. As a practical matter, it may be difficult to come by a sufficiently similar sample of messages prior to deploying a spam filter.

The on-line model presents to the filter a chronological sequence of n messages, m_0 through m_{n-1}. For each message m_i a classifier is induced on m_0 through m_{i-1}, the subsequence of messages prior to m_i. This classifier is used to predict the class of m_i. The on-line model is similar, but not equivalent, to n instances of the batch model:

- the training messages m_0 through m_{i-1} form a sequence rather than a set; the order of the sequence may be useful (or deleterious) to the classifier;
- the training sequence may be very small, or empty, or lack positive or negative examples;
- implementations requiring n invocations of a batch algorithm are unlikely to be efficient enough for practical use – even a linear-time induction algorithm would yield a quadratic-time on-line classifier.

A filter designed according to one model – batch or on-line – may be adapted for the other. A batch filter might simply treat $\{m_j \mid j < i\}$ and $\{m_i\}$ as training and test sets, inducing a new classifier for each message. Such an approach would be prohibitively slow, and could not harness information present in the sequencing of m_0 through m_{i-1}. The filter might break the message sequence into batches of size k; for each batch b ($0 \leq b \leq \lfloor \frac{n}{k} \rfloor$) treating $\{m_j \mid j < bk\}$ as the training set and $\{m_j \mid bk \leq j < (b+1)k\}$ as the test set. This approach improves efficiency by a factor of k or more, but may compromise effectiveness by reducing the effective amount of training data by up to b messages; in particular the b most recent prior messages which may most resemble those to be classified. Finally, one might define a training window of size w and use $\{m_j \mid bk-w \leq j < bk\}$ as the training set for each batch. This approach yields linear-time efficiency (as opposed to quadratic or worse depending on the efficiency of the inducer) at the expense of a reduced training set size. On the other hand, this approach facilitates – perhaps by accident rather than by design – adaptation in that only recent prior messages influence the classifier. For our experiments, we used $w = 10000$ and $k = 1$ for the first 1000 messages, $k = \lfloor \frac{i}{20} \rfloor$ for the next 9000 messages, and $k = 500$ thereafter. A number of other approaches might be used. For example, one might preserve the 'best' examples as training data, or one might implement (if known) an incremental training algorithm.

An on-line filter might similarly be adapted to batch testing. The batch training set may be converted to a sequence by the application of some order. The order might be arbitrary (such as the order in which the messages are stored in a file), randomized, or determined by design. While order may have a dramatic effect on performance, determining the optimal order presents a challenge. To effect batch testing, it is necessary to disable adaptation or any other change in the classifier. Assuming this is done, the order of the test sequence is irrelevant. In summary, an on-line filter may be tested by running it on the training set (in some order), disabling adaptation, and running it on the test set (in any order). In our experiments, we used a random order unless otherwise stated. A number of other approaches might be used. For example, the filter may use multiple passes, or train on the 'best' examples, or use less efficient approaches that are impractical in an on-line setting.

We note that the same issues apply to hybrid schemes in which training is batch but testing is on-line, or vice versa.

3. TOKENIZATION

Many spam filters use simple textual patterns to identify features from which a classifier is induced. Several corpora presuppose a particular set of patterns and provide only the features. Ling Spam [1] removes all but the subject line of message headers, converts all letters to lower case, and replaces all strings of non-alphanumeric characters by a single white space. PU1 [2] further converts each distinct alphanumeric string to a unique integer represented as a sequence of decimal digits. The 2006 ECML Discovery Challenge[3] goes further: strings that occur fewer than four times in the corpus are eliminated, and each message is represented by a vector indicating the number of occurrences of each feature in the message. The format is exactly that required as input by SVM^{light} [17] and *TR-IRLS* [19].

Some filters – PPM and DMC in particular – treat each message as a sequence of characters or bits, and automatically discover patterns of interest. Others – like Bogofilter – match header information and also the implicit semantics of words and other tokens in identifying features. Such filters are best suited to corpora that contain raw messages with complete headers.

PPM and DMC required no adaptation for use on the

[3]http://www.ecmlpkdd2006.org/challenge.html

Ling Spam and PU1 corpora. Although the messages were processed, the result was still a string of bytes (bits) thus meeting their input requirements. The processing may have removed beneficial information, and the arbitrary representation of features in PU1 may compromise effectiveness, but but we have no way to recover the lost information. Bogofilter required no adaptation to achieve good performance on Ling Spam. However, we needed to transliterate decimal digits to letters in order to achieve acceptable performance on PU1. Obviously, Bogofilter treats numbers differently from 'words'!

SVM and LR require feature vectors as input. For the Ling Spam and PU1 corpora, we treat each distinct token in the corpus as a feature, with a value of 1 if the token occurs in the messages; 0 otherwise. We did not investigate, for example TF-IDF weights as feature values, but we note that the performance of this simple scheme is as good as any reported in the literature for the same corpora.

To apply SVM and LR to the TREC corpus, we implemented a feature extraction scheme similar to that for Ling Spam and PU1 – we first decoded base-64 content, and discarded non-text attachments. Full headers were preserved (in contrast to Ling Spam and PU1). We identified and uniquely labelled each string of alphanumeric characters in the corpus. A binary-valued feature vector was constructed for each message. We also generated preprocessed versions of the messages – in Ling Spam and PU1 formats – to measure the impact of this style of preprocessing on the other filters.

4. EVALUATION MEASURES

A hard classifier yields a categorical ham or spam judgement for each test message. Although the term 'soft classifier' appears not to be widely used, we shall take it to mean a classifier that returns a score s that increases monotonically with the classifier's estimate of the probability that the message is spam. Within the context of spam filtering, soft classification has several uses:

- the soft classifier is easily used as a hard classifier by comparing s to a fixed threshold t – a higher value of t will reduce false positives (misclassified ham messages) at the cost of increased false negatives (misclassified spam messages) while a lower value of t will have the opposite effect;
- t may be adjusted by the user to account for different sensitivities to misclassified ham and spam messages;
- t may be tuned automatically to optimize a particular utility function [20];
- email may be separated into several ordinal categories such as ham, *unsure*, and spam [22];
- quarantined email categorized as spam may be ranked by s to facilitate occasional searches for mis-quarantined ham [10];
- s may provide useful input to a stacking metaclassifier [3, 21].

Within the context of a batch model, a soft classifier is exactly what Sebastiani terms a *ranking* classifier [30]. In a sense, the same is true of an online classifier, except that the ranking (by s) is implicit and incremental – each message takes its place among previously-ranked messages which are, once placed, never reordered.

Many hard classifiers compute an internal score s which makes a suitable soft classifier. SVM and LR fall into this category. Other hard classifiers may take as input parameters for a utility function (equivalent to t) which alters the behaviour of the classifier. Such a classifier could be used as a soft classifier by running it for several values of t and reporting as s the maximum value for which the hard classification was *spam*.

The efficacy of a hard classifier may be measured by its false positive rate (fpr) and false negative rate (fnr), defined as the proportion of ham (resp. spam) messages that it misclassifies. A classifier with lower fpr and fnr than another is superior[4]. Whether a classifier with a lower fpr and higher fnr is superior or inferior depends on the user's sensitivity to each kind of error. A plethora of measures – including accuracy, weighted accuracy, total cost ratio, F-measure, and utility – attempt to quantify this sensitivity and to use this quantification to combine fpr and fnr, along with the corpus ham-to-spam ratio, into a one-dimensional measure. We argue that it is better to report the two dimensional (fpr, fnr) pair which may be combined post-hoc with deployment-specific estimates of the relative costs of misclassification (cf. Cormack and Lynam [11]). We transform published results for hard classifiers to (fpr, fnr) which we plot as points in ROC space (cf. [13, 11]).

The efficacy of a soft classifier may be characterized by the set of all distinguishable (fpr, fnr) pairs for different values of t. This set of points defines a curve in ROC space – a filter whose ROC curve is strictly above that of another is superior, while a filter whose ROC curve crosses that of another is superior for some cost parameters and inferior for others. The area under the ROC curve (AUC) provides an estimate of the effectiveness of a soft classifier over all threshold settings. AUC also has a probabilistic interpretation: it is the probability that the classifier will award a random spam message a higher value of s than a random ham message. In the spam filtering domain, typical AUC values are of the order of 0.999 or greater; for clarity, we report (1-AUC)% , the area above the ROC curve, as a percentage.

5. FILTERS, METHODS AND CORPORA

We have implemented or adapted five spam filtering algorithms for both batch and on-line evaluation:

- PPM – Prediction by Partial Matching – the adaptive method labelled *ijsSPAM2* at TREC 2005 [5], based on the PPM data compression method [8];
- DMC – Dynamic Markov Compression – a new adaptive method [4] inspired by but implemented independently of *ijsSPAM2*, based on the DMC compression method [9];
- *Bogofilter* – version 0.94.0 [25] – a popular open-source 'Bayesian' spam filter that performed well at TREC;
- SVM – SVM^{light}[17] – an established free-for-scientific-use support vector machine classifier;

[4]Under the assumption that all messages have equal misclassification cost. See Kolcz et al. [18]

- LR – *TR-IRLS* [19] – an established open-source logistic regression classifier.

We have evaluated these implementations, and where appropriate compared the results to the literature, using the following techniques and corpora:

- 10-fold cross-validation using the Ling Spam [1] corpus;
- 10-fold cross-validation using the PU1 corpus [2];
- TREC on-line methodology using the TREC 2005 public corpus[5];
- 10-fold cross-validation using new random splits of the TREC 2005 corpus[6];
- 9:1 chronological split[7] of the TREC 2005 corpus, batch test;
- 9:1 chronological split of the TREC 2005 corpus, online test.

We have, where meaningful, repeated the tests with two preprocessed variants of the TREC corpus, in which the vocabulary of each message corresponds to the feature set extracted for use by the SVM and LR methods:

- a tokenized version in the style of the Ling Spam corpus, in which contiguous sequences of alphanumeric characters are preserved and other sequences of characters are replaced by a single white space character;
- an obfuscated version in which each sequence of alphanumeric characters is further replaced by a unique integer represented by a sequence of decimal digits.

6. LING SPAM AND PU1 RESULTS

As two of the oldest public corpora, Ling Spam and PU1 have been used in many studies. The ones that report statistics from which it is possible to derive ham and spam misclassification rates are listed, with a mnemonic label and a short description, in table 1. The results of each are denoted by one or more points in the ROC graphs in figure 1. The results of our five test filters – labelled DMC, PPM, Bogofilter, Logistic and SVM – appear as multi-point curves on the same graphs. All results are based on 10-fold cross validation using the splits defined with each corpus.

The curves for DMC and PPM dominate the methods in the Ling Spam results, with Bogofilter slightly inferior. LR and SVM are somewhat lower. On the PU1 corpus, which is very small, DMC, PPM, Logistic and SVM appear to have about the same performance, above Bogofilter and the other points and curves. Table 2 lists $(1 - AUC)\%$ with 95% confidence limits for each of the curves, computed using the TREC Spam Filter Evaluation Toolkit. These statistics confirm the same impressions; however the confidence intervals – especially for PU1 – are sufficiently large that these results are not by themselves conclusive. However, we believe it is possible to conclude that our test methods – DMC and PPM in particular – compare favorably with existing results on these corpora.

We note that DMC and PPM perform well in spite of the fact that the results are derived from a batch evaluation and that the messages are tokenized, stripped of headers and, in the case of PU1, obfuscated.

7. TREC CORPUS RESULTS

We used the messages in the TREC 2005 Public Corpus [10] for our primary experiments. Our first experiment used exactly the TREC methodology, presenting to each filter the entire corpus in chronological order. DMC, PPM, and Bogofilter are directly amenable to on-line classification; LR and SVM were run as described previously, with a maximum window size of 10000 and a maximum batch size of 500. Table 3 (left column) and figure 2 (left panel) show the ROC results. These results may be compared directly with those labeled "full" in the TREC proceedings[8] [10]; PPM had the best performance at TREC on this corpus. Bogofilter was 5th best. DMC's performance exceeds that of any TREC participant; our adapted LR and SVM methods, while inferior to Bogofilter, would have ranked with the better systems.

Table 3 (third column) and figure 2 (right panel) show the result of 10-fold cross validation on the same messages. The training set was presented to the adaptive methods in random order. All filters – except DMC – achieve substantially better results with cross-validation. PPM shows the most pronounced effect, the net effect being vastly superior performance at all threshold levels. Superior performance under cross-validation is not surprising as the filters have the benefit of a substantial number of training examples throughout the test, as opposed to a very limited number for the initial messages in the on-line test. Also, the presence of 'future' messages in the training sets may provide an oracle to the classifiers[9]. DMC's lack of improvement may be due to the fact that DMC adapts quickly, taking advantage of the temporal locality in message characteristics; locality that is absent with randomly ordered messages. The advantage of cross-validation may be offset by the lack of locality. This hypothesis is investigated further in the following experiments.

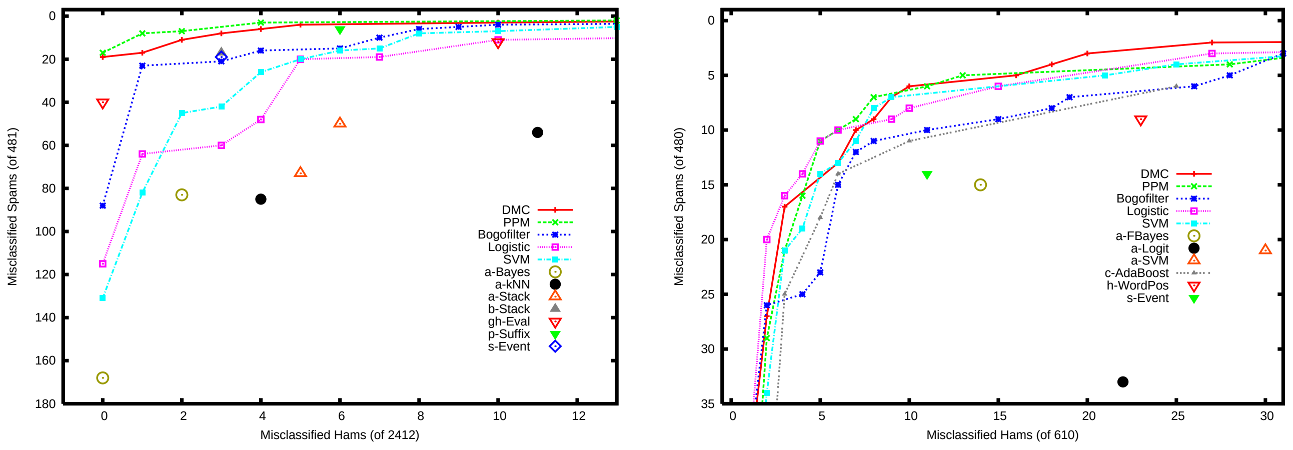
We performed two tests – one on-line and one batch – using the first 90% of the corpus for training and the last 10% (9219 messages) for testing. Effecting the on-line test was easy; we simply discarded the first 82970 of our full-corpus results. For the batch tests we re-ran the systems. DMC was trained on chronologically ordered messages (as were the other systems, but they are insensitive to training order). If the messages in the corpus were homogeneous, we would expect the results for the chronological batch run to be about the same as for cross-validation, and the on-line results to be insubstantially better as a result of slightly more training data. The results of the batch test (table 3, 4th column; figure 3, right panel) indicate that the chronological split is quite different from the 10-fold splits 'easier' for the adaptive methods and much 'harder' for the batch methods. LR and SVM are at a considerable disadvantage; the high

[5]http://plg.uwaterloo.ca/~gvcormac/treccorpus
[6]http://plg.uwaterloo.ca/~gvcormac/trecsplits.gz
[7]Training: first 82970 messages in index; test: last 9219.

[8]However, note that the TREC ROC curves are plotted on a logit scale; ours are linear.
[9]Fawcett [12] draws attention to this and other anomalies in batch filter evaluation.

Label	Description
a-Bayes	Naive Bayes, multi-variate Bernoulli model on binary features [1]
a-kNN	k-nearest neighbors with attribute and distance weighting [27]
a-Stack	Stacking of naive Bayes and k-nearest neighbors [26]
b-Stack	Stacking of linear support vector machine classifiers built from different message fields [6]
gh-Eval	Naive Bayes with weighting of training instances according to misclassification cost ratio [15]
p-Suffix	Pattern matching of character sequences based on suffix tree data structure and heuristic scoring functions [23]
s-Event	Multinomial naive Bayes [28]
a-FBayes	Flexible naive Bayes – uses kernel density estimation to estimate class-conditional probabilities of continuous valued attributes [2]
a-Logit	LogitBoost with decision stumps as base classifiers [2]
a-SVM	Linear kernel support vector machine [2]
c-AdaBoost	Boosted decision trees with real-valued predictions [7]
h-WordPos	Multinomial naive Bayes [16]

Table 1: Results of Previous Studies



Ling Spam Corpus PU1 Corpus

Figure 1: Ling Spam and PU1 Corpora

Method	Ling Spam	PU1
DMC	.07 (.01-.49)	.26 (.09-.80)
PPM	**.04 (.004-.39)**	**.18 (.08-.43)**
Bogofilter	.04 (.02-.11)	.21 (.10-.43)
LR	.087 (.04-.17)	.20 (.07-.53)
SVM	.14 (.08-.26)	.22 (.10-.50)

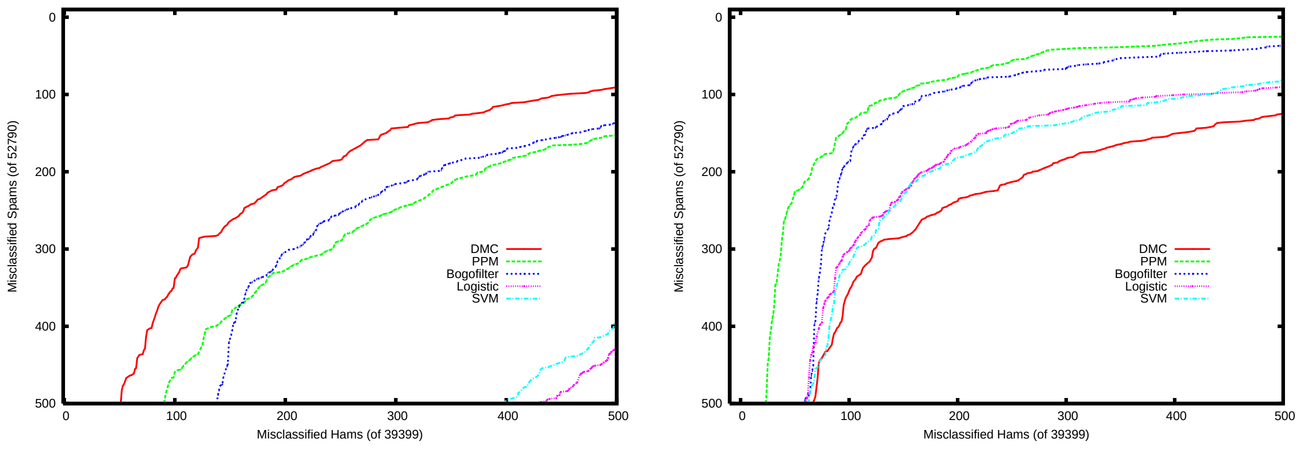
Table 2: Cross-validation Results – Ling Spam and PU1 Corpora (1-AUC)%

	On-line		Batch	
Method	Full Corpus	9:1 Chronological	10-fold C.V.	9:1 Chronological
DMC	**.013 (.010-.018)**	**.0003 (.0000-.003)**	.015 (.012-.018)	**.003 (.001-.006)**
PPM	.017 (.014-.021)	.0007 (.0001-.005)	**.006 (.004-.009)**	.003 (.001-.008)
Bogofilter	.048 (.038-.062)	.002 (.0001-.041)	.020 (.012 - .033)	.009 (.003-.029)
LR	.068 (.058-.079)	.020 (.003-.135)	.016 (.012-.021)	.12 (.001-10.1)
SVM	.075 (.064-.088)	.007 (.0015-.033)	.021 (.015-.029)	.13 (.003-5.6)

Table 3: Overall Results – TREC Corpus (1-AUC)%

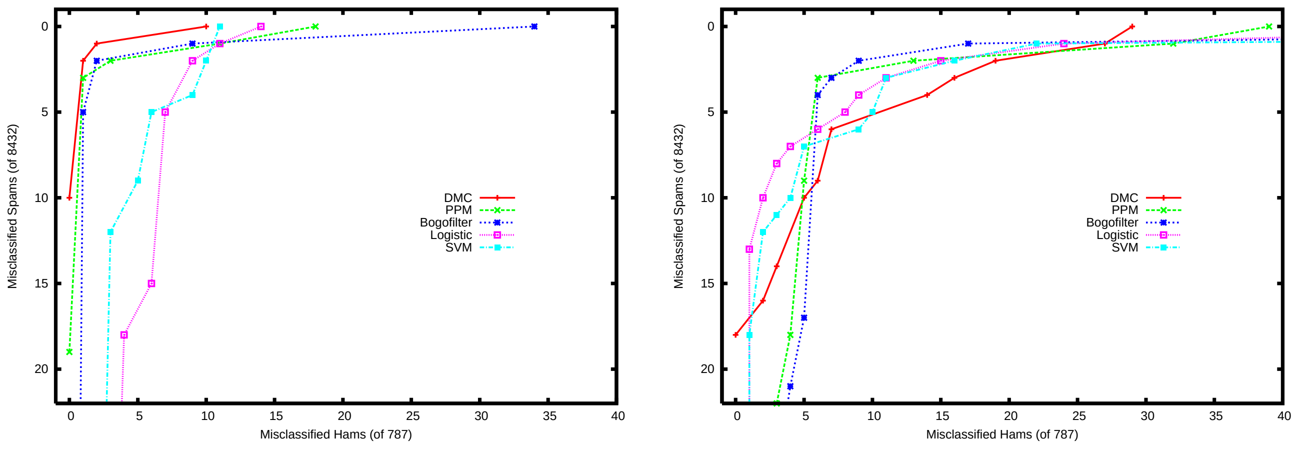
$(1-AUC)\%$ statistics and huge confidence intervals indicate instability. DMC appears to take the most advantage, with results improved five-fold.

The results for adaptive testing are even more striking (table 3, 2nd column; figure 3, left panel). DMC shows a factor of ten improvement over the batch test; a factor of fifty over its cross-validation performance. PPM, Bogofilter, and SVM show a five-fold improvement over batch, while



On-line Evaluation 10-fold Cross Validation

Figure 2: TREC 2005 Public Corpus



On-line Batch

Figure 3: 9:1 Chronological Split

Filter	Training Regimen	Testing Regimen On-line Random Order	On-line Corpus Order	Batch
DMC	Random Order	.01 (.006-.017)	.007 (.004-.011)	.009 (.006-.015)
DMC	Corpus Order	.035 (.026-.047)	.037 (.024-.057)	.31 (.25-.37)
PPM	Batch	.0052 (.003-.01)	.0053 (.003-.009)	.0055 (.003-.01)

Table 4: Effect of Training/Test Order (Fold 0)

SVM shows a factor of twenty.

We performed two further experiments to explore the effect of training and testing order on DMC and PPM. We examined the effect of processing the training and test messages from fold 0 (from the 10-fold cross validation) in various orders. Each row of table 4 corresponds to a different training regimen; each column corresponds to a different test regimen. For DMC, row 1 column 3 corresponds to the strategy used in the cross-validation. Columns 1 and 2 show the effect of on-line training in random and corpus orders. As predicted, on-line testing in random order appears to have little effect (the apparent effect is negative, but could easily be due do chance). On-line testing in corpus (chronological) order may have some positive effect, but this test lacks the statistical power to provide convincing evidence. The results when DMC is trained on ordered data (with the test set excluded) – are much more dramatic. Batch performance is thirty times worse. This degraded performance is mitigated by a factor of ten when on-line testing is used, regardless of testing order. The third column indicates that PPM is oblivious to the ordering effects that affected DMC so dramatically. Training order is irrelevant, and the results are essentially the same as for the cross-validation, regardless of testing regimen.

The results show a large amount of locality in the TREC data; the characteristics of examples appear to change with time. Unlike PPM, the DMC adaptive model is affected by the order in which examples are presented for training,

	On-line		Batch	
Method	Full Corpus	9:1 Chronological	10-fold C.V.	9:1 Chronological
DMC	*.013 (.010-.018)*	*.0003 (.0000-.003)*	*.015 (.012-.018)*	*.003 (.001-.006)*
tokenized	.025 (.020-.032)	.0006 (.0001-.006)	.025 (.019-.033)	.001 (.000-.013)
obfuscated	.037 (.030-.045)	.0004 (.0000-.0042)	.029 (.023-.037)	.002 (.001-.006)
PPM	.017 (.014-.021)	*.0007 (.0001-.005)*	*.006 (.004-.009)*	*.003 (.001-.008)*
tokenized	.038 (.033-.045)	.0016 (.0003-.009)	.012 (.009-.016)	.005 (.002-.012)
obfuscated	.075 (.066-.084)	.0046 (.0016-.013)	.020 (.014-.027)	.015 (.006-.035)
Bogofilter	*.048 (.038-.062)*	*.002 (.0001-.041)*	*.020 (.012 - .033)*	*.009 (.003-.029)*
tokenized	.13 (.11-.15)	.024 (.004-.14)	.055 (.045-.068)	3.1 (2.8-3.4)
obfuscated	.13 (.11-.15)	.024 (.004-.14)	.059 (.048-.071)	3.7 (3.3-4.1)

Table 5: Effect of Tokenization and Obfuscation – TREC Corpus (1-AUC)%

and is most influenced by the more recent examples. This behaviour serves to its advantage in a typical online setting, but may also hurt the method when training and test data are presented in an "unnatural" order.

8. EFFECT OF TOKENIZATION

The Ling Spam and PU1 results led us to ponder the effect of tokenization and obfuscation on filter performance. Since DMC and PPM examine sequences of characters or bits, altering the data encoding might have a significant impact. To measure this impact, we created two tokenized versions of the TREC corpus; one tokenized in the style of Ling Spam and one obfuscated in the style of PU1. Our principal experiments were repeated on these corpora; the results appear in table 5. Baseline results, repeated from table 3, are given in italics.

PPM is consistently hurt by tokenization, and hurt further by obfuscation; approximately by a factor of two in each instance. DMC is hurt by tokenization and obfuscation on the full corpus test, but not so much, if at all, on the chronological splits. In fact, for the batch test, performance apparently improves. This apparent improvement may be due to chance; we can offer no more plausible explanation.

Bogofilter is hurt substantially by tokenization but not much, it appears, by further obfuscation. Note that tokenization degrades Bogofilter's performance on the batch chronological split by two orders of magnitude. Examination of the data reveals that this phenomenon is largely due to the existence of bursts of near-identical spam messages. In the original format, Bogofilter is able to identify features (perhaps headers or obfuscated words) that allow it to correctly classify these messages; our tokenization defeats this ability. The phenomenon does not occur with cross-validation because the training sets contain examples of each burst.

9. DISCUSSION

Our results invite further experiment and analysis. The performance of our reference implementations – both on-line and batch – is limited by our imagination, skill and implementation effort. While comparison with published results indicates that our implementations are at least reasonable, one should not conclude, for example, that SVM and LR are inferior for on-line filtering. One may conclude, on the other hand, that DMC and PPM set a new standard to beat on the most realistic corpus and test available at this time.

Within this context, two recent results are worthy of note. Goodman and Yih [14] use gradient descent in a truly adaptive logistic regression method, achieving superior performance to our logistic regression method in on-line tests on the full TREC corpus. Lynam and Cormack [21] achieve superior overall performance for the on-line tests by fusing the results of the fifty-three filters evaluated at TREC. These results are reproduced in table 6.

The mechanisms behind the strong performance of the compression-based methods appear to be complex, and somewhat different for the two methods. Both take advantage of intra-word and inter-word patterns that distinguish spam from ham. DMC, in addition, harnesses inter-message correlations, as its model is biased toward recent examples. This bias, we think, makes DMC in general more adaptive than PPM. Both models appear to handle heterogeneous data better than the feature-based methods. A better understanding of the mechanisms might help to guide feature extraction for machine-learning algorithms.

The TREC corpus is quite obviously inhomogeneous and demonstrates strong temporal locality. The ham and some of the spam in this corpus is derived from the Enron corporation through a time of upheaval, from business-as-usual through bankruptcy and receivership. So it is not surprising that the nature of messages changes with time. We do not believe that this makes it an unrepresentative corpus – people's and organizations' circumstances change, and a corresponding change in the nature of email is to be expected. Also, spam itself adapts, as the process is adversarial. Further investigation is required to determine the extent to which the TREC corpus is typical in this regard, and whether the observed effects are due to changes in the nature of the ham, the spam, or both.

Our results show that on-line testing and cross-validation are different, and support arguments questioning the validity of cross validation for testing spam filters[10]. Cross-validation excessively penalizes methods with a high degree of adaptation, although they are extremely competitive on real e-mail sequences. Notwithstanding these reservations as to the validity of batch testing, we see that DMC, Bogofilter and, in particular, PPM yield superior results in batch testing; however much of their advantage may be due to better implicit or explicit feature engineering, as evidenced by the fact that the superiority is compromised by the use of obfuscated tokens. Better feature engineering might give feature-based techniques the edge; for example, Peng et al report good performance for language-based features [24] that harness the inter-word patterns that we believe account, at least

[10]See, for example page 40 of:
http://research.microsoft.com/~joshuago/tutorialOnJunkMailFilteringjune4.pdf

Method	On-line Full Corpus	On-line 9:1 Chronological
Adaptive Logistic Regression [14]	.022 (.017-.027)	.066 (.0003-11.9)
TREC Filter Fusion [21]	.007 (.005-.008)	.0003 (.0000-.01)

Table 6: Related Results – TREC Corpus (1-AUC)%

partially, for the compression models' performance. Sculley and Brodley [29] present a general framework for the relationship between compression models and feature selection.

The TREC results are not an anomaly. Well-performing systems at TREC demonstrated superior performance relative to published results on other corpora. A new method – DMC – inspired by the the best system at TREC – PPM – demonstrates even better performance, although our testing reveals that DMC and PPM achieve their results through somewhat different mechanisms.

References

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., and Spyropoulos, C. An evaluation of naive Bayesian anti-spam filtering, 2000.

[2] Androutsopoulos, I., Paliouras, G., and Michelakis, E. Learning to filter unsolicited commercial E-Mail. Tech. Rep. 2004/2, NCSR "Demokritos", October 2004.

[3] Bennett, P. N., Dumais, S. T., and Horvitz, E. The combination of text classifiers using reliability indicators. *Inf. Retr. 8*, 1 (2005), 67–100.

[4] Bratko, A., Cormack, G. V., Filipic, B., Lynam, T. R., and Zupan, B. Spam filtering using statistical data compression.

[5] Bratko, A., and Filipič, B. Spam filtering using character-level markov models: Experiments for the TREC 2005 Spam Track. In *Proc. 14th Text REtrieval Conference (TREC 2005)* (Gaithersburg, MD, November 2005).

[6] Bratko, A., and Filipič, B. Exploiting structural information for semi-structured document categorization. *Information Processing & Management 42*, 3 (2006), 679–694.

[7] Carreras, X., and Márquez, L. Boosting trees for anti-spam email filtering. In *Proc. of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing* (2001).

[8] Cleary, J. G., and Witten, I. H. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications COM-32*, 4 (April 1984), 396–402.

[9] Cormack, G. V., and Horspool, R. N. S. Data compression using dynamic Markov modelling. *The Computer Journal 30*, 6 (1987), 541–550.

[10] Cormack, G. V., and Lynam, T. R. Overview of the TREC 2005 Spam Evaluation Track. In *Fourteenth Text REtrieval Conference (TREC-2005)* (Gaithersburg, MD, 2005), NIST.

[11] Cormack, G. V., and Lynam, T. R. On-line supervised spam filter evaluation. http://plg.uwaterloo.ca/~gvcormac/spamcormack, 2006.

[12] Fawcett, T. 'In Vivo' spam filtering: A challenge problem for data mining. *KDD Explorations 5*, 2 (December 2003).

[13] Fawcett, T. ROC graphs: Notes and practical considerations for researchers. Tech. Rep. HPL-2003-4, 2004.

[14] Goodman, J., and tau Yih, W. Online discriminative spam filter training. In *The Third Conference on Email and Anti-Spam* (Mountain View, CA, 2006).

[15] Hidalgo, J. M. G. Evaluating cost-sensitive unsolicited bulk email categorization. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing* (Madrid, March 2002), ACM Press, pp. 615–620.

[16] Hovold, J. Naive Bayes spam filtering using word-position-based attributes. In *Proc. of the 2nd Conference on Email and Anti-Spam (CEAS 2005)* (Palo Alto, CA, July 2005).

[17] Joachims, T. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998.

[18] Kolcz, A., and Alspector, J. SVM-based filtering of E-mail spam with content-specific misclassification costs. *TextDM 2001 (IEEE ICDM-2001 Workshop on Text Mining)* (2001).

[19] Komarek, P., and Moore, A. Fast robust logistic regression for large sparse datasets with binary outputs. In *Artificial Intelligence and Statistics* (2003).

[20] Lewis, D. D. Evaluating and optimizing autonomous text classification systems. In *SIGIR'95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle, Washington, USA, July 9-13, 1995 (Special Issue of the SIGIR Forum)* (1995), E. A. Fox, P. Ingwersen, and R. Fidel, Eds., ACM Press, pp. 246–254.

[21] Lynam, T. R., and Cormack, G. V. On-line spam filter fusion. In *29th ACM SIGIR Conference on Research and Development on Information Retrieval* (Seattle, 2006).

[22] Meyer, T. A. A TREC along the spam track with SpamBayes. In *Proc. 14th Text REtrieval Conference (TREC 2005)* (Gaithersburg, MD, November 2005).

[23] Pampapathi, R. M., Mirkin, B., and Levene, M. A suffix tree approach to email filtering. Tech. rep., Birkbeck University of London, 2005.

[24] Peng, F., Schuurmans, D., and Wang, S. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval 7*, 3-4 (2004), 317–345.

[25] Raymond, E. S., Relson, D., Andree, M., and Louis, G. Bogofilter. http://bogofilter.sourceforge.net/, 2004.

[26] Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., and Stamatopoulos, P. Stacking classifiers for anti-spam filtering of e-mail, 2001.

[27] Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., and Stamatopoulos, P. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval 6*, 1 (2003), 49–73.

[28] Schneider, K. M. A comparison of event models for naive bayes anti-spam E-mail filtering. In *Proc. of the 10th Conference of the European Chapter of the Association for Computational Linguistics* (2003).

[29] Sculley, D., and Brodley, C. E. Compression and machine learning: A new perspective on feature space vectors. In *Data Compression Conference (DCC 06)* (Snowbird, 2006), pp. 332–341.

[30] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (2002), 1–47.