Filtering Email Spam in the Presence of Noisy User Feedback

D. Sculley
Department of Computer Science
Tufts University
161 College Ave.
Medford, MA 02155 USA
dsculley@cs.tufts.edu

Gordon V. Cormack
School of Computer Science
University of Waterloo
2502 David Centre
Waterloo, Ontario N2L 3GI Canada
gvcormac@uwaterloo.ca

Abstract

Recent email spam filtering evaluations, such as those conducted at TREC, have shown that near-perfect filtering results are attained with a variety of machine learning methods when filters are given perfectly accurate labeling feedback for training. Yet in real-world settings, labeling feedback may be far from perfect. Real users give feedback that is often mistaken, inconsistent, or even maliciously inaccurate. To our knowledge, the impact of this *noisy* labeling feedback on current spam filtering methods has not been previously explored in the literature. In this paper, we show that noisy feedback may harm or even break state-of-the-art spam filters, including recent TREC winners. We then propose and evaluate several approaches to make such filters robust to label noise. We find that although such modifications are effective for uniform random label noise, more realistic "natural" label noise from human users remains a difficult challenge.

1 Introduction: Noise in the Labels

Unwanted or harmful electronic messages, known as *spam*, are a particular problem for electronic mail: it has recently been estimated that 80% of all email traffic is spam (Goodman & Yin, 2006). Automated spam filters based on machine learning methods offer a robust methodology for fighting spam, relying on user to provide feedback labels of *spam* or *ham* (i.e. not spam) for each message for training updates. Recent evaluations have shown this approach to be startlingly effective. Best results from the TREC 2007 spam filtering competition were near-perfect with several filters achieving ROC areas better than 0.9999 (Cormack, 2007a).

However, these promising results were attained in a laboratory setting where *gold-standard* feedback was given to the filters for training. That is, the labels of *spam* and *ham* assigned to each message was carefully vetted, and such labeling was done in a consistent and accurate fashion (Cormack & Lynam, 2005b). In real-world systems involving users, it is unlikely that these (possibly anonymous) humans will consistently give label feedback of gold-standard quality. Instead, real users give *noisy* feedback, and the labels used for training real-world filters may contain errors reducing classification performance.

1.1 Causes of Noise

Label noise may come from a variety of causes. Several different insiders in industrial anti-spam settings have reported that at least 3% of all user feedback is simply mistaken. That is, these labels are objectively wrong, on the level of email lottery scams being reported *ham*. (This 3% user mistake rate is also reported in Yin, *et al.*, 2006.) Sources of labeling mistakes include misunderstanding the feedback mechanisms, accidental clicks, or even users who are actually fooled by such scams.

Inconsistent labels arise when similar messages are perceived differently among users. A common example of this are *gray mail* messages (such as email newsletters) that some users value and others prefer to block (Yin, *et al.*, 2007). In a recent paper studying gray mail, it was found that one sample of 418 messages[1] contained 163 gray mail messages – nearly 40% (Yin, *et al.*, 2007). An additional data point comes from John Graham-Cumming's `spamorham.org` project,[2] in which human users across the internet were invited to manually label messages in the `trec05p-1` data set. In

[1] This was apparently randomly sampled from a large corpus of representative spam emails.

[2] Sadly, this project ended in early 2007 and the `spamorham.org` domain is now apparently controlled by web spammers.

this project, individual human labelers disagreed with the gold standard labels 10.9% of the time, in large part due to inconsistency or human error (Graham-Cumming, 2006).

Finally, maliciously inaccurate feedback is an issue in large free email systems. For example, a spammer may acquire many free accounts, send large amounts of spam to these accounts, and then report such messages as being *ham*. Industry insiders at several different large free email systems have confirmed that this is a common tactic by spammers. Indeed, at least one such system chooses to completely ignore all *ham* labels given by users.

Thus, the possibility of label noise is an important real-world consideration for spam filtering. Spam filters based on machine learning techniques must be robust to a variety of label noise levels, and not be narrowly optimized to perform well only in the noise-free case.

1.2 Contributions

This paper makes three main contributions. First, it is shown that even uniformly random noise in label feedback significantly harms the performance of state of the art spam filters. Second, several modifications are proposed for making filters robust to label noise, including making less aggressive updates, label cleaning, label correcting, and various forms of regularization. It is shown that the best of these methods make filters siginificantly more robust to uniform label noise at a small cost in classification performance in the noiseless case. Third, it is found that natural noise from real users is more challenging than uniform noise.

2 Related Work

There is relatively little published work on the impact of label noise in spam filtering. However, the problem of noisy class labels and avoiding overfitting is well studied in general machine learning literature.

2.1 Label Noise in Email Spam

To our knowledge, this is the first paper to explicitly explore the problem of label noise in email spam filtering. However, label noise has been considered as an issue by industry experts in spam filtering, and has been given as a reason to prefer filtering methods that do not rely on user feedback. Furthermore, the published acknowledgment of 3% labeling errors in data gained by the Hotmail Feedback Loop implies that this problem has been previously studied internally (Yin *et al.*, 2006). Additionally, Yin *et al.* (2007) studied gray mail detection as a subset of this general label noise problem. They found that adding a gray mail detector as the first stage of a two-stage filtering process reduced false negatives between two and six percent.

2.2 Avoiding Overfitting

In machine learning, it has long been known that *overfitting* is a potential problem when applying optimization methods to noisy training data (Mitchell, 1997). For iterative methods such as gradient descent, *early stopping* is a common and effective practice for reducing overfitting (Mitchell, 1997). In the online filtering scenario for streaming data the effect of early stopping may be approximated by selecting a conservative learning rate. Another common approach is *regularization*, in which a penalty for model complexity is added to the optimization problem (see, for example, Scholkopf & Smola, 2001).

The problem of label noise in data has also been well studied. Zhu and Wu (2004) observed that class label noise can be even more harmful in training classifiers than feature noise.[3] Several methods have been proposed for *cleaning* data containing class label noise by discarding training examples that are suspected to have incorrect labels (see Zhu and Wu (2004) for a comprehensive overview.) A more aggressive approach is label *correcting*, in which instances suspected to be mislabeled are automatically re-labeled (Zheng & Martinez, 2001). Rebbapragada and Brodley (2007) showed that correcting and cleaning can be considered within the same unifying framework, but also note that label correcting is a more difficult task that may introduce additional errors into the data.

3 Label Noise Hurts Filters

We first wished to investigate how well top performing filters from recent TREC evaluations fared with respect to label noise. This section describes our basic experimental design, the filters under consideration, and results from this first exploration.

3.1 Evaluation

We use the standard online filtering scenario as our experimental framework, in which messages are presented to the filter one at a time. For each message, the filter is asked to give a prediction score of *ham* or *spam*. After the prediction is made, the message's label is revealed to the filter, and this information may be used for a training update (Cormack & Lynam, 2007).

For evaluation, we use the (1-ROCA)% measure that

[3] In the spam setting, feature noise is typified by the *good word attack* (Lowd & Meek, 2005).

has become standard in spam filter evaluation (Cormack & Lynam, 2007). This measure reflects the area *above* the ROC curve, expressed as a percentage, and may be statistically interpreted as the percent chance that a randomly selected ham message will be erroneously predicted by the filter to be more "spammy" than a randomly selected spam message (Cormack & Lynam, 2007). The (1-ROCA)% score is computed over all messages seen by the filter in the online filtering test.

Note that in our experiments the labels given to the filter during online filtering may be noisy, as described above. However, the (1-ROCA)% evaluation score is computed with respect to the gold standard labels supplied with the original data set, which have been carefully vetted for accuracy and consistency (Cormack & Lynam, 2005). Thus, our goal is to assess the impact of noisy training labels on the filters' ability to predict true gold-standard labels. In an ideal world, a small amount of label noise would do only minimal harm to the filters' classification performance.

3.2 Data Sets with Synthetic Noise

For these initial tests, we built noisy data sets from two large, publicly available benchmark data sets for spam filter evaluation: the `trec06p` data set of 37,822 emails (Cormack, 2006) and the `trec07p` data set of 75,419 emails (Cormack, 2007), both of which were originally constructed for the TREC spam filtering competitions. (All of the noisy data sets used in this paper are publicly available for research purposes; contact the authors.)

For this initial evaluation, we chose to investigate the effect of uniform, random label noise. (Natural label noise from actual human feedback is tested in Section 5.2.) Synthetic noise was added to each data set as follows: for each message, the label of that message was flipped with uniform probability p. We created one test set for each data set and each of seven noise levels, with $p = \{0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25\}$. Note that when $p = 0$, the test set is identical to the original, unaltered TREC data set.

3.3 Filters

We tested a range of current statistical spam filtering methods, as follows. Where noted, parameters were set by tuning on the separate `spamassassin` data set[4] of 6,032 messages using (1-ROCA)% as the evaluation measure. Other filters were tested with parameters set as given by default, or as given in the reference describing that filter. Unless noted otherwise, the filters tested used a feature space of binary 4-*mers* drawn from the first 3000 characters of each message and all feature vectors were normalized with the Euclidean norm (see Sculley and Wachman (2007) for details).

- **Multi-Nomial Naive Bayes (MN NB).** Variants of the Naive Bayes classifier (Mitchell, 1997) have been popular for spam filtering since they were proposed for this task by Paul Graham (2002, 2003). Metsis *et al.* (2006) tested a number of Naive Bayes variants, and found Multi-Nomial Naive Bayes with binary feature values to be one of the top performing methods.

- **Logistic Regression.** To our knowledge, logistic regression was first proposed for spam filtering by Goodman and Yin (2006). When coupled with binary 4-*mers*, this machine learning method gave best results on several tasks at TREC 2007 (Cormack, 2007a, 2007b). Online logistic regression has a parameter η that controls learning rate, determining how aggressively to update the model on each new example. We set to $\eta = 0.1$ after tuning on `spamassassin` data.

- **Perceptron with Margins (PwM).** This variant of the classical perceptron algorithm employs a fixed update-margin (W. Krauth & M. Mézard, 1987) that functions as a cheap approximation of the large-margin principle, giving good tolerance to noise in data (Khardon & Wachman, 2007). This algorithm was the base learner in an approach that gave strong results at TREC 2006 (Sculley, *et al.*, 2006). After tuning, we set the fixed margin $m = 8$ and learning rate $\eta = 0.5$.

- **Relaxed Online Support Vector Machine (ROSVM)** This sliding window variant of the soft-margin Support Vector Machine (see Scholkopf and Smola, 2001) was recently developed and proposed for online spam filtering (Sculley & Wachman, 2007). In this work, it was found that high values of the cost parameter C, encouraging little regularization, gave best results for spam filtering tasks. This result agreed with a prior finding by Drucker *et al.* (1999), which also found that SVMs gain best performance on spam data with high values of C. Using $C = 100$, the ROSVM method gave best results on several tasks at TREC 2007 (Cormack, 2007a).

- **Dynamic Markov Chain Compression (DMC).** Perhaps the best performing of the compression-based spam filters (Bratko *et al.*, 2006), DMC was tested at TREC 2007 as `wat2` with strong results (Cormack, 2007b).

[4] Available at: `www.spamassassin.org`

Table 1: Results for prior methods on `trec06p` data set with uniform synthetic noise. Results are reported as (1-ROCA)%, with 0.95 confidence intervals. Bold numbers indicate best result for a given noise level, or confidence interval overlapping with confidence interval of best result.

`trec06p`	NOISE 0	NOISE 0.01	NOISE 0.05	NOISE 0.10	NOISE 0.15	NOISE 0.20	NOISE 0.25
MN NB	0.477 (0.425 - 0.535)	0.513 (0.460 - 0.571)	0.517 (0.459 - 0.582)	**0.517** (0.459 - 0.583)	**0.624** (0.557 - 0.698)	**0.665** (0.594 - 0.744)	**0.685** (0.620 - 0.758)
LogReg	**0.032** (0.025 - 0.041)	**0.035** (0.027 - 0.046)	**0.118** (0.099 - 0.140)	**0.615** (0.558 - 0.677)	2.107 (1.985 - 2.236)	4.914 (4.732 - 5.102)	9.077 (8.774 - 9.390)
PwM	**0.049** (0.034 - 0.070)	0.069 (0.050 - 0.094)	**0.181** (0.149 - 0.221)	**0.577** (0.526 - 0.632)	1.517 (1.424 - 1.615)	3.328 (3.173 - 3.491)	6.666 (6.423 - 6.918)
ROSVM	**0.031** (0.021 - 0.044)	0.328 (0.288 - 0.373)	2.430 (2.305 - 2.561)	6.532 (6.297 - 6.775)	11.512 (11.182 - 11.850)	16.852 (16.449 - 17.263)	21.680 (21.262 - 22.104)
DMC	**0.031** (0.024 - 0.041)	**0.053** (0.040 - 0.070)	**0.183** (0.150 - 0.222)	**0.619** (0.542 - 0.706)	1.430 (1.308 - 1.564)	3.044 (2.869 - 3.230)	5.208 (4.986 - 5.439)
Bogo	0.087 (0.066 - 0.114)	0.096 (0.071 - 0.130)	0.277 (0.231 - 0.332)	1.203 (1.110 - 1.304)	3.168 (2.999 - 3.346)	7.336 (7.066 - 7.616)	11.478 (11.148 - 11.818)
WAT1	**0.036** (0.027 - 0.049)	0.075 (0.058 - 0.096)	0.389 (0.347 - 0.435)	1.839 (1.723 - 1.963)	4.548 (4.360 - 4.743)	8.358 (8.073 - 8.651)	13.112 (12.755 - 13.478)
OSBF-Lua	**0.054** (0.034 - 0.085)	0.075 (0.053 - 0.107)	0.316 (0.155 - 0.644)	29.575 (28.622 - 30.546)	35.011 (34.371 - 35.657)	38.486 (37.855 - 39.122)	39.699 (39.046 - 40.356)

Table 2: Results for prior methods on `trec07p` data set with uniform synthetic noise. Results are reported as (1-ROCA)%, with 0.95 confidence intervals. Bold numbers indicate best result for a given noise level, or confidence interval overlapping with confidence interval of best result. Methods unable to complete a given task are marked with DNF.

`trec07p`	NOISE 0	NOISE 0.01	NOISE 0.05	NOISE 0.10	NOISE 0.15	NOISE 0.20	NOISE 0.25
MN NB	0.168 (0.151 - 0.185)	0.181 (0.163 - 0.203)	0.237 (0.216 - 0.259)	**0.249** (0.227 - 0.273)	**0.297** (0.273 - 0.324)	**0.298** (0.272 - 0.326)	**0.342** (0.309 - 0.377)
LogReg	**0.005** (0.002 - 0.017)	**0.006** (0.004 - 0.009)	**0.071** (0.061 - 0.084)	0.550 (0.512 - 0.590)	2.266 (2.184 - 2.351)	5.192 (5.045 - 5.343)	9.501 (9.271 - 9.737)
ROSVM	**0.010** (0.003 - 0.030)	0.031 (0.020 - 0.048)	0.473 (0.436 - 0.512)	2.246 (2.157 - 2.339)	5.604 (5.410 - 5.804)	9.531 (9.260 - 9.809)	14.714 (14.215 - 15.228)
DMC	**0.006** (0.003 - 0.016)	0.021 (0.014 - 0.031)	**0.103** (0.084 - 0.126)	0.242 (0.209 - 0.280)	0.594 (0.533 - 0.661)	1.208 (1.123 - 1.300)	2.484 (2.354 - 2.621)
Bogo	**0.027** (0.017 - 0.043)	0.033 (0.023 - 0.049)	**0.097** (0.077 - 0.122)	**0.264** (0.231 - 0.302)	0.504 (0.454 - 0.559)	3.221 (3.083 - 3.365)	10.294 (10.041 - 10.551)
WAT1	**0.006** (0.002 - 0.015)	0.019 (0.014 - 0.026)	0.428 (0.396 - 0.462)	1.984 (1.904 - 2.068)	5.221 (5.043 - 5.405)	9.226 (9.000 - 9.457)	14.116 (13.851 - 14.385)
OSBF-Lua	**0.029** (0.015 - 0.059)	0.054 (0.016 - 0.184)	0.290 (0.097 - 0.859)	29.478 (27.591 - 31.432)	DNF	DNF	DNF

- **BogoFilter.** BogoFilter has been the best performing open-source spam filter at TREC for several years (Cormack, 2006, 2007a), and employs a fast variant of the Naive Bayes classifier.

- **WAT1**. This is the same filter submitted as `wat1` at TREC 2007 (Cormack, 2007b), which gave best results on several tasks. This filter employs logistic regression and binary 4-*mers*, but does not normalize the feature vectors.

- **OSBF-Lua.** This is the same filter that gave best overall performance at TREC 2006 (Cormack, 2006). OSBF-lua uses an aggressive variant of the Naive Bayes classifier, in which the filter continues to re-train on the header of a given email message so long as the filter scores that message near the classification boundary, or until other stopping conditions such as maximum number of iterations are met (Assis, 2006). We refer to this update strategy as *train until no error*.

3.4 Initial Results

The results of this first experiment reveal a disturbing trend, as shown in Tables 1 and 2. The methods giving best results without label noise give *worst* results with moderate to large amounts of label noise. This is true for Logistic Regression, ROSVMs, DMC, BogoFilter, and even Perceptron with Margins, each of which has given strong performance in the TREC spam filtering competitions. In contrast, the Multi-Nomial Naive Bayes method gives relatively modest results without noise, but is much more robust to increasing levels of label noise.

What causes the steep degradation in performance with the state of the art filters? Each of these methods is tuned to perform aggressive online updates, necessary to attain competitive results in the noise-free TREC evaluations. For example, the Logistic Regression method is tuned with an aggressive learning rate η and uses no regularization. The ROSVM method is tuned with the cost parameter C set to a high value discouraging regularization. Such settings allow the filters to quickly adapt to new spam attacks when user feedback contains no noise, but makes these filters subject to *overfitting* when label noise is present.

As an extreme case, OSBF-Lua, the top performer from TREC 2006, was actually broken by the noisy data, eventually giving results of `nan` on messages in all noisy data sets with $p > 0$. The train-until-no-error approach severely overfit mislabeled instances, resulting in a useless model.

These are a troubling initial results. Together, they call into question the real-world utility of top filters from previous evaluations. Are the strong performance levels from TREC only achievable in laboratory settings, with users willing and able to give perfectly accurate feedback? The remainder of this paper investigates this question.

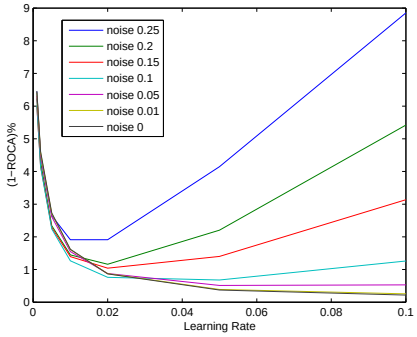

Figure 1: Results for varying learning rate η for Logistic Regression, on `spamassassin` tuning data with varying levels of synthetic uniform label noise.

4 Filtering without Overfitting

In this section, we suggest several strategies for making learning-based filtering methods more robust to noise in feedback. These strategies include tuning parameters to prevent overly aggressive updates, various forms of regularization for logistic regression and SVM variants, and methods that attempt to automatically clean or even correct labels given for training.

For preliminary experiments and tuning runs in this section, we created noisy versions of the `spamassassin` data set, adding uniform synthetic label noise at different levels as described in Section 3.2.

4.1 Tuning Learning Rates

As discussed above, both Logistic Regression and Perceptron with Margins utilize a learning rate parameter η that controls the size of the step taken on any given update during online gradient descent. Lower values of η lead to less aggressive updates, giving an online approximation of the early stopping strategy that gives good results when gradient descent is applied in batch mode (Mitchell, 1997).

Figure 1 shows the effect of varying η for Logistic Regression at different levels of label noise on the noisy `spamassassin` data sets. (Similar effects are seen with Perceptron with Margins). Note that when there is little or no label noise, high values of η give best results, but when label noise becomes more prevalent lower η values (centering on $\eta = 0.02$) improve results.

For our final experiments in the next section, we set $\eta = 0.02$ for Logistic Regression and $\eta = 0.02$ with

margin= 2 for Perceptron with Margins, as these values give best results at noise level $p = 0.25$ and near-best results for other noise values at or above $p = 0.1$ on the noisy `spamassassin` tuning data.

4.2 Regularization

Another general strategy for reducing overfitting is *regularization*, requiring that the learned model not only describes the training data well, but also has low complexity (see, for example, Scholkopf and Smola, 2001). One measure of model complexity is the L2-norm (or Euclidean norm) of the weight vector. Thus, L2 regularization seeks to ensure that the Euclidean norm of the weight vector is as small as possible, while still describing the training data well. These goals of fitting the training data and reducing model complexity are often in conflict, and the balance between these goals is controlled by a parameter.

4.2.1 Regularization with SVM variants

The classic soft-margin SVM optimization problem is to minimize:

$$||\mathbf{w}||^2 + C\sum_{i}^{m}\xi_i$$

Here, $\mathbf{w}$ is the weight vector storing the model, each ξ_i is a *slack term* describing the amount of error associated with a particular training example $\mathbf{x}_i$ (Scholkopf & Smola, 2001). Thus, the optimization problem seeks to minimize both model complexity (the L2-norm of $\mathbf{w}$) and training error (given by the sum of the slack terms), and the cost parameter C controls how much emphasis to place on each of these tasks in training.

A high value of C focuses on reducing training error by enforcing little regularization, resulting in the possibility of overfitting. Both Sculley and Wachman (2007) and Drucker *et al.* (1999) found that high values of C gave best performance on spam data for ROSVMs and SVMs, respectively, but these results were gained with no label noise in the data. As shown in Figure 2, lower values of C give much improved performance in the presence of noise. We set $C = 0.5$ for our final experiments, as this gives best results for noise level $p = 0.25$.

4.2.2 Regularization for Logistic Regression

Logistic Regression is often considered to be especially prone to overfitting in the absence of regularization (Mitchell, 2005). For the online gradient descent algorithm commonly used for online logistic regularization, L2 regularization is achieved with a modified update rule (Mitchell, 2005):

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(y_i - f(\mathbf{x}_i))\mathbf{x}_i - \eta\lambda\mathbf{w}$$

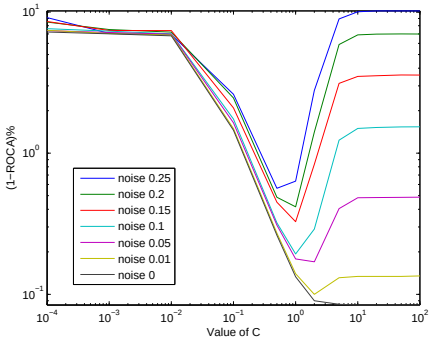

Figure 2: Results for varying C in ROSVM for regularization, on `spamassassin` tuning data with varying levels of synthetically added label noise.

As before, $\mathbf{w}$ is the weight vector, $\mathbf{x}_i$ is an individual training example with label $y_i \in \{0, 1\}$, and the learning rate is given by η. The prediction function $f(\mathbf{x}_i)$ returns a value between 0 and 1 indicating predicted "spamminess" of $\mathbf{x}_i$. Regularization is controlled by the parameter λ, where larger values of λ enforce more regularization and reduce overfitting.

Note that this modified update rule increases computational cost. Where before each update could be performed in time $O(|\mathbf{x}_i|)$, where $|\mathbf{x}_i|$ is the number of nonzero elements of the sparse vector $\mathbf{x}_i$, now each update requires $O(|\mathbf{w}|)$, the number of non-zero features in $\mathbf{w}$, which may be considerably larger.

After a grid search for parameter values of λ and η using `spamassassin` tuning data, we were surprised to find that L2 regularization with values of λ ranging from $\lambda = 10^{-7}$ to $\lambda = 10^2$ did not improve results at any noise level. Values above $\lambda = 10^{-4}$ monotonically degraded results, and smaller values gave results effectively equivalent to $\lambda = 0$. To investigate this further, we chose a value of $\lambda = 0.0001$ (with $\eta = 0.02$) for our final experiments on test data, as this was the largest value that did not significantly decrease performance on tuning data compared to $\lambda = 0$.

4.3 Label Cleaning

Another machine learning approach for coping with noisy labels is automated *label cleaning*, in which examples that are suspected to be incorrectly labeled are discarded from the data set (Brodley & Friedl, 1999). To use this approach in the online filtering scenario, we suggest an obvious online algorithm, given in Figure 3. This method uses confidence thresholds t_{+1} and t_{-1}, to define criteria for cleaning.

In our experiments, we apply this algorithm using Logistic Regression as the base learner, so that t_{+1} and t_{-1} may be interpreted as probability thresh-

olds. After tuning on noisy `spamassassin` data, we set $t_{+1} = 0.7$ and $t_{-1} = 0.3$, as these values gave best results for noise level $p = 0.25$ with $\eta = 0.02$.

for each NEW EXAMPLE $\mathbf{x}_i$:
USE FILTER TO MAKE PREDICTION, USING $f(\mathbf{x}_i)$
GET (POSSIBLY NOISY) LABEL y_i FROM ORACLE
if ($f(\mathbf{x}_i) < t_{+1}$ AND $y_i == -1$) **or**
($f(\mathbf{x}_i) > t_{-1}$ AND $y_i == +1$)
then UPDATE MODEL USING ($\mathbf{x}_i$, y_i)
else DISCARD $\mathbf{x}_i$ AND SKIP MODEL UPDATE

Figure 3: Pseudo-code for Online Label Cleaning.

4.4 Label Correcting

In the label correcting method, the filter proactively changes the labels of examples for which the filter strongly disagrees with the given label (Zeng & Martinez, 2001), at the risk of introducing additional noise into the data (Rebbapragada & Brodley, 2007). We propose a simple online method for label correcting, given in Figure 4, similar to the online label cleaning method. After tuning, we set $t_{+1} = 0.95$ and $t_{-1} = 0.05$ with $\eta = 0.02$, using Logistic Regression as the base learner.

for each NEW EXAMPLE $\mathbf{x}_i$:
USE FILTER TO MAKE PREDICTION, USING $f(\mathbf{x}_i)$
GET (POSSIBLY NOISY) LABEL y_i FROM ORACLE
if ($f(\mathbf{x}_i) > t_{+1}$) **then** SET $y_i := +1$
if ($f(\mathbf{x}_i) < t_{-1}$) **then** SET $y_i := -1$
UPDATE MODEL USING ($\mathbf{x}_i$, y_i)

Figure 4: Pseudo-code for Online Label Correcting.

5 Experiments

In this section, we test the modified filters, with parameters tuned as described in Section 4, on both synthetic label noise and on natural noise from real users.

5.1 Synthetic Label Noise

We first tested the modified methods on synthetic label noise, using the same data and evaluation methods described in Section 3. Results for these experiments are shown in Tables 3 and 4.

First, we note that simply reducing the learning rate η makes Logistic Regression and Perceptron with Margins much more resistant to label noise for both data sets, at a slight cost in performance without label noise. Perceptron with Margins, in particular, demonstrates its effectiveness as a "noise-tolerant" algorithm (Khardon & Wachman, 2007).

Second, additional regularization gives strong results with ROSVM, but does not give added benefit for Logistic Regression with $\lambda = 0.0001$. To check if this was because of a particular λ value, we ran additional tests and found that higher values of λ monotonically degraded classification performance at all noise levels, while lower values of λ converged to the results where $\lambda = 0$. Thus, it appears that L2 regularization for logistic regression is simply not helpful for email spam filtering. We believe this is due to the fact that in the online gradient descent with L2 penalty used for Logistic Regression, rare-but-informative features are penalized over time. In contrast, the SVM variant is better suited to maintaining values for many relevant features (Joachims, 1999).

Third, the approach of label cleaning gave excellent results, clearly improving on base Logistic Regression results on both data sets at moderate to high levels of noise. Label correcting, on the other hand, did not give added benefit. When we explored other parameter settings, we found that more aggressive label correcting only degraded results on these data sets.

5.2 Natural Label Noise

The previous experiments show that there are several methods available for dealing with uniform label noise, an interesting result given the failure of the best TREC filters on the same task. But is a uniform model of label noise always realistic? It seems reasonable that messages such as gray mail may have higher rates of labeling inconsistency than average. Similarly, if spammers are inside the labeling system, then certain spam messages may have a disproportionately high noise rate. In this section, we experiment with our best available approximation of label noise caused by actual users, using human labels collected by the `spamorham.org` project (Graham-Cumming, 2006) for the `trec05p-1` data set.

To prepare test data with natural label noise, for each message in the `trec05p-1` data set we sampled one human labeling from the set of all human labeling for that message, uniformly at random. Thus, the final data set contained the same messages as `trec05p-1` in the same order, but with labels that reflected the distribution of label noise produced by human users. In comparison with the `trec05p-1` gold standard labels, this test set contained 6.75% incorrect labels.

For comparison, we then created a synthetic data set from `trec05p-1`, with a uniform $p = 0.0675$ noise rate identical to that of the natural label noise data set. Finally, we also tested all methods on the original `trec05p-1` data with gold-standard labels.

The results, given in Table 5, show that natural label

Table 3: Results for modified methods on `trec06p` data set with uniform synthetic noise. Results are reported as (1-ROCA)%, with 0.95 confidence intervals. Bold numbers indicate best result, or confidence interval overlapping with confidence interval of best result.

`trec06p`	NOISE 0	NOISE 0.01	NOISE 0.05	NOISE 0.10	NOISE 0.15	NOISE 0.20	NOISE 0.25
LogReg $\eta = 0.02$	0.070 (0.057 - 0.085)	0.060 (0.049 - 0.073)	**0.047** (0.038 - 0.058)	**0.064** (0.051 - 0.080)	**0.074** (0.062 - 0.089)	0.142 (0.120 - 0.169)	0.401 (0.363 - 0.444)
LogReg L2-rglz. $\eta = 0.02, \lambda = 0.0001$	0.068 (0.057 - 0.082)	0.059 (0.049 - 0.071)	**0.046** (0.037 - 0.059)	**0.064** (0.049 - 0.082)	**0.074** (0.060 - 0.090)	0.141 (0.118 - 0.168)	0.398 (0.361 - 0.439)
LogReg Labl-Corr.	0.107 (0.087 - 0.132)	0.112 (0.094 - 0.134)	0.106 (0.086 - 0.132)	0.136 (0.111 - 0.167)	0.093 (0.077 - 0.113)	0.147 (0.122 - 0.176)	0.403 (0.366 - 0.443)
LogReg Labl-Clean	**0.049** (0.037 - 0.065)	**0.049** (0.038 - 0.062)	**0.045** (0.035 - 0.058)	**0.060** (0.047 - 0.076)	**0.055** (0.043 - 0.069)	**0.086** (0.068 - 0.110)	**0.107** (0.090 - 0.128)
PwM $\eta = 0.02, m = 2$	**0.036** (0.027 - 0.047)	**0.035** (0.026 - 0.048)	**0.043** (0.032 - 0.058)	**0.049** (0.035 - 0.068)	**0.053** (0.040 - 0.070)	**0.066** (0.048 - 0.089)	**0.082** (0.066 - 0.103)
ROSVM $C = 0.5$	**0.033** (0.025 - 0.044)	**0.032** (0.023 - 0.044)	**0.036** (0.026 - 0.052)	**0.040** (0.029 - 0.054)	**0.046** (0.034 - 0.062)	**0.062** (0.047 - 0.081)	**0.095** (0.076 - 0.119)

Table 4: Results for modified methods on `trec07p` data set with uniform synthetic noise. Results are reported as (1-ROCA)%, with 0.95 confidence intervals. Bold numbers indicate best result, or confidence interval overlapping with confidence interval of best result.

`trec07p`	NOISE 0	NOISE 0.01	NOISE 0.05	NOISE 0.10	NOISE 0.15	NOISE 0.20	NOISE 0.25
LogReg $\eta = 0.02$	**0.009** (0.004 - 0.018)	**0.007** (0.004 - 0.011)	**0.007** (0.005 - 0.010)	**0.014** (0.008 - 0.023)	0.036 (0.028 - 0.045)	0.091 (0.078 - 0.107)	0.363 (0.332 - 0.396)
LogReg L2-rglz. $\eta = 0.02, \lambda = 0.0001$	**0.009** (0.004 - 0.020)	**0.007** (0.004 - 0.011)	**0.007** (0.005 - 0.009)	**0.014** (0.008 - 0.025)	0.035 (0.027 - 0.047)	0.091 (0.077 - 0.107)	0.359 (0.326 - 0.395)
LogReg Labl-Corr.	**0.009** (0.005 - 0.018)	**0.010** (0.005 - 0.018)	**0.011** (0.006 - 0.020)	**0.016** (0.011 - 0.024)	0.037 (0.029 - 0.047)	0.092 (0.079 - 0.107)	0.364 (0.338 - 0.393)
LogReg Labl-Clean	**0.010** (0.005 - 0.018)	**0.011** (0.005 - 0.021)	**0.013** (0.008 - 0.023)	**0.010** (0.005 - 0.019)	**0.010** (0.006 - 0.018)	**0.017** (0.011 - 0.025)	**0.018** (0.013 - 0.025)
PwM $\eta = 0.02, m = 2$	**0.008** (0.004 - 0.017)	**0.011** (0.006 - 0.020)	0.021 (0.013 - 0.033)	**0.021** (0.013 - 0.035)	0.043 (0.030 - 0.062)	0.056 (0.040 - 0.079)	0.087 (0.069 - 0.110)
ROSVM $C = 0.5$	**0.006** (0.002 - 0.018)	**0.007** (0.003 - 0.014)	**0.010** (0.006 - 0.017)	**0.008** (0.005 - 0.013)	**0.027** (0.017 - 0.044)	**0.033** (0.020 - 0.054)	0.048 (0.034 - 0.068)

Table 5: Results for natural and synthetic noise at identical noise levels. Natural label noise for `trec05p-1` was uniformly sampled from human labelings collected by the `spamorham.org` project. Results are reported as (1-ROCA)%, with 0.95 confidence intervals.

`trec05p-1` FILTERS	NO NOISE	SYNTH. NOISE	NATURAL NOISE
MN NB	0.871 (0.831-0.913)	1.270 (1.210-1.333)	1.425 (1.364-1.489)
LogReg $\eta = 0.1$	0.013 (0.011-0.015)	0.249 (0.232-0.267)	0.563 (0.531-0.596)
PwM $\eta = 0.5, m = 8$	0.022 (0.018-0.027)	0.324 (0.297-0.352)	1.056 (0.982-1.137)
ROSVM $C = 100$	0.012 (0.010-0.016)	2.063 (1.987-2.142)	2.172 (2.088-2.259)
DMC	0.013 (0.009-0.017)	0.241 (0.217-0.268)	0.574 (0.535-0.616)
BogoF	0.042 (0.031-0.056)	7.215 (7.021-7.413)	0.551 (0.512-0.593)
WAT1	0.012 (0.010-0.015)	1.073 (1.017-1.131)	1.294 (1.233-1.358)
OSBF-Lua	0.011 (0.008-0.014)	37.025 (34.895-39.207)	32.770 (32.436-33.105)
LogReg $\eta = 0.02$	0.031 (0.027-0.034)	0.037 (0.324-0.043)	0.156 (0.145-0.168)
LogReg LABL. CORR.	0.030 (0.028-0.036)	0.039 (0.034-0.043)	0.146 (0.135-0.159)
LogReg LABL. CLEAN	0.022 (0.018-0.027)	0.025 (0.020-0.030)	0.463 (0.423-0.506)
PwM $\eta = 0.02, m = 2$	0.022 (0.018-0.027)	0.047 (0.038-0.060)	0.304 (0.0276-0.336)
ROSVM $C = 0.5$	0.019 (0.015-0.023)	0.030 (0.023-0.039)	0.294 (0.264-0.327)

noise appears to be more challenging to filters than uniform noise. The unmodified filters perform badly on both the synthetic noise and natural noise. In contrast, the modified filters perform relatively well on the synthetic noise, but give results roughly an order of magnitude worse on the natural noise (although still better than the unmodified filters). These results agree with previous observations that uniform class label noise is easier to filter than label noise that skews the label distribution in certain regions of the feature space (Brodley & Friedl, 1999; Rebbapragada & Brodley, 2007). Additional work is needed for this natural label noise.

6 Discussion

At the outset of this investigation, our goal was to find out how top-performing filters from TREC competitions fared in the presence of label noise. To our dismay, we found that even uniform label noise dramatically reduces the effectiveness of these state of the art filters when run "out of the box." We then found inexpensive modifications enabling TREC filters to become significantly more tolerant of label noise. Uniform label noise, which models random user errors in feedback, is well handled by several modified methods. Natural noise, reflecting inconsistent or malicious judgments, remains more difficult.

We observe that these "noise tolerant" filters would not necessarily have achieved best performance on the tasks as given in TREC-style evaluations. The noiseless evaluation setting rewards aggressive online updates, and promotes filters that may be prone to overfitting in real-world applications. However, we feel that a slight decrease in classification performance in the noiseless setting is more than compensated for by improved performance in the noisy setting.

It is critical that we are able to distinguish those filters that are robust to label noise (or may be made robust with appropriate parameter settings) from those that fail in noisy settings. Thus, we would like to propose that future spam filtering evaluations include filtering tasks with various levels of label noise. Ideally, this label noise would be natural noise from real human users rather than synthetic, wherever possible, as this is the more challenging case.

References

F. Assis. OSBF-Lua – a text classification module for lua: the importance of the training method. In *TREC 2006: Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

A. Bratko, B. Filipic, G. V. Cormack, T. R. Lynam, and B. Zupan Spam filtering using compression models. Journal of Machine Learning Research, (7), 2006.

C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. Journal of Artificial Intelligence Research, (11), 1999.

G. V. Cormack. TREC 2006 spam track overview. In *TREC 2006: Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

G. V. Cormack. TREC 2007 spam track overview. In *TREC 2007: Proceedings of the The Sixteenth Text REtrieval Conference*, 2007. (a)

G. V. Cormack. University of waterloo participlation in the TREC 2007 spam track. In *TREC 2007: Proceedings of the The Sixteenth Text REtrieval Conference*, 2007. (b)

G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In *The Fourteenth Text REtrieval*

Conference (TREC 2005) Proceedings, 2005. (a)

G. V. Cormack and T. R. Lynam. Spam corpus creation for TREC. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005. (b)

G. V. Cormack and T. R. Lynam. Online supervised spam filter evaluation. *ACM Transactions on Information Systems*, (25):3, 2007.

J. Graham-Cumming. There's one born every minute: spam and phishing. http://www.jgc.org/blog/2006_05_01_archive.html. 2006.

H. Drucker, V. Vapnik, and D. Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

J. Goodman and W. Yin. Online discriminative spam filter training. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

P. Graham. A plan for spam. 2002.

P. Graham. Better bayesian filtering. 2003.

T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

R. Khardon and G. Wachman. Noise tolerant variants of the perceptron algorithm. *J. Mach. Learn. Res.*, 8, 2007.

W. Krauth and M. Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20(11):745–752, 1987.

D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes – which naive bayes? *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

T. M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression. In *Machine Learning*. http://www.cs.cmu.edu/∼tom/ mlbook/NBayesLogReg.pdf, 2005.

T. M. Mitchell. *Machine Learning*. McGraw-Hill. 1997.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, (14), 2002.

U. Rebbapragada and C. E. Brodley Class noise mitigation through instance weighting. *ECML 2007, 18th European Conference on Machine Learning*, 2007

B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

D. Sculley and G. Wachman. Relaxed online SVMs for spam filtering. In *The Thirtieth Annual ACM SIGIR Conference Proceedings*, 2007.

D. Sculley, G. Wachman, and C. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *The Fifteenth Text REtrieval Conference (TREC 2006) Proceedings*, 2006.

G. L. Wittel and S. F. Wu. On attacking statistical spam filters. *CEAS: First Conference on Email and Anti-Spam*, 2004.

W. Yin, J. Goodman and G. Hulten Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

W. Yin, R. McCann and A. Kolcz Improving spam filtering by detecting gray mail. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS)*, 2007.

X. Zeng and T. Martinez An algorithm for correcting mislabeled data. *Intelligent Data Analysis*, 5(1), 2001.