Highly Scalable Discriminative Spam Filtering

Michael Brückner, Peter Haider, and Tobias Scheffer
Max Planck Institute for Computer Science, Saarbrücken, Germany
Humboldt Universität zu Berlin, Germany
{brum, haider, scheffer}@mpi-inf.mpg.de

Abstract

This paper discusses several lessons learned from the SpamTREC 2006 challenge. We discuss issues related to decoding, preprocessing, and tokenization of email messages. Using the Winnow algorithm with orthogonal sparse bigram features, we construct an efficient, highly scalable incremental classifier, trained to maximize a discriminative optimization criterion. The algorithm easily scales to millions of training messages and millions of features. We address the composition of training corpora and discuss experiments that guide the construction of our SpamTREC entry. We describe our submission for the filtering tasks with periodical re-training and active learning strategies, and report on the evaluation on the publicly available corpora.

1 Introduction

Spam filtering remains a technological challenge; the commercial incentive for spam senders results in an arms race between filtering methods and spam obfuscation techniques. Naive Bayes [5, 7] and rule based learners [2] have been very popular; discriminative approaches like Support Vector Machines, Logistic Regression as well as Maximum Entropy have also been studied for spam filtering. Most discriminatively trained methods are non-incremental and often scale poorly to large training samples. Regular updates of the classifier which are necessary due to topic drift and the adversarial nature of spam, are consequently costly or even infeasible.

P-norm algorithms such as Winnow [9, 10] are incremental and can be implemented to scale to large amounts of training data. These methods have proven to be very robust [4] and highly scalable in practice. A number of alternative approaches such as network-based spam detection, collaborative filtering strategies, or email batch detection using graph theoretical methods have been studied but cannot be applied to the SpamTREC challenge because of the nature of the data that is provided.

In this paper, we address challenging issues for the construction of practical filtering systems. We discuss how a large-scale filter can be trained using a discriminative optimization criterion. We address preprocessing, decoding, and tokenization issues, and questions regarding the assembly of training corpora. We report on experiments that guide the construction of our filtering system. We discuss the evaluation of this system on the public SpamTREC corpora and conclude with a number of lessons learned.

The rest of this paper is structured as follows. We address challenges for practical spam filters in Section 2 and report on our experiments. Section 3 describes the system that we used in the competition; Section 4 concludes.

2 Challenges in Email Classification

In this section, we address design issues for practical spam filtering systems and report on experiments that guide the construction of a system that performs well for the SpamTREC challenge.

2.1 Scalable Discriminative Training

Text classification requires highly scalable methods as it involves large amounts of high dimensional data. Winnow, a perceptron-like algorithm with multiplicative updates [9], can handle a huge amount of data very efficiently, outperforming other well-established filters such as Naive Bayes at the same time [10]. However, practical difficulties and implementation issues still remain.

Beyond preprocessing and tokenization, an efficient implementation has to compute the features – hash values of the parsed tokens or N-grams – and spam scores on the fly. Computation of the score involves huge hash structures with several million buckets. Our implementation is an efficient version of Winnow which scales to large corpora and feature vectors. A related version of our filtering system is used by a commercial webspace and email service provider and filters about 35 million email messages per day.

2.2 Preprocessing and Tokenization

Encoding schemes provide spam senders with a rich set of tools to obfuscate tokens. For example, the word "café" would be HTML-encoded as "café", URL-encoded as "caf=E9", or base64-encoded as "Y2Fm6Q==". In order to avoid an inflation of the attribute space, it is highly desirable to map all these representations to the same token. This can be achieved by transforming tokens into a canonical encoding scheme, such as UTF-8.

Our spam filtering system includes modules for a variety of different preprocessing measures. They incorporate procedures to analyze the structure, and conformity to several standards, of each email and its parts. They include procedures that transform the email contents into the canonical UTF-8 encoding scheme. These transformation steps result in a representation of the message text that is independent of the particular encapsulation method used.

The conformity checks and the attachment dissection provide additional features that can be used for classification. For example, spammers often forge the date of the email, such that the message appears user as the most recent item the user's inbox for a long time.

In detail, prior to tokenization each email is subjected to the following actions:

- parsing structure of MIME-parts;
- decoding MIME-part contents (e.g., base64 or "quoted/printable" decoding);
- transforming the character set into the UTF-8 encoding scheme;
- decoding the subject-string according to RFC2947 and RFC2231;
- transforming HTML- or URL-encoded characters into UTF-8;
- plausibility checks of the "received" time according to RFC2822;
- extraction of language information based on used character sets;
- extraction of information about attachment types;

- checking of standard conformity of MIME structure and attached files.

We conduct a set of experiments in order to study the benefit of these pre-processing and feature extraction steps. First, we study the effectiveness of pre-processing and its contribution to the classification results. Therefore we train two classifiers using identical training sets containing 100,000 randomly selected emails from our English corpus (Table 1). For the first classifier, the itemized preprocessing steps are carried out, whereas they are disabled for the second classifier. We use an evaluation set of again 100,000 emails from the same corpus. We observe an AUC performance of 0.999477 when preprocessing is employed and 0.999380 when it is disabled. That is, the preprocessing step decreases the risk $(1 - AUC)$ by 16% from 0.00062 to 0.000523. In this experiment, clearly the accuracy is high for both classifiers. Nevertheless, a 16% reduction of the risk is a significant finding that emphasizes the importance of the preprocessing step.

Preprocessing has a noticeable effect on the subsequent feature generation step. Without preprocessing, trained on 100,000 English documents, the filter has 6 million features with nonzero weights. When preprocessing is employed, only 3 million features are used.

The stream of preprocessed characters has to be tokenized in a way that facilitates the extraction of discriminative attributes. Whereas tokenization can be based on whitespace characters and punctuation symbols for European languages, such whitespace characters are absent for many Asian languages such as Chinese, Japanese and Korean languages. For Asian languages, we treat each character as an individual token, resulting in syllabic tokens. The resulting loss of inter-syllabic context is compensated for by the use of N-gram features in the following feature extraction step.

Table 1: Data used for experiments

Source	*English docs*	*Chinese docs*
CCERT Data Set (www.ccert.edu.cn/spam)	10125	64015
Disclosed Enron emails [6]	20556	0
Guenter spam trap (untroubled.org/spam)	142887	0
IMC mailing lists (www.imc.org)	20154	15075
Various newsletters	9861	0
SpamArchive.org	65541	0
SpamAssassin collection	8270	0
TREC 2005 corpus	91146	0
Various moderated Usenet groups	39655	32021
Wouters archive (www.xtdnet.nl/paul/spam)	0	24278
Own collection	91805	64611

2.3 Text Feature Extraction

From the stream of preprocessed tokens, features have to be extracted that provide the classification method with sufficiently discriminative information to allow for a highly accurate decision. In many cases, contextual information that spans across multiple tokens hints at the semantics of sentences. This is particularly true for Asian languages for which words span across multiple tokens. In addition, it has become popular among spam senders to blur the bag-of-word-view of the messages by appending random sets of *good words*, individually drawn for each message.

Orthogonal sparse bigrams provide a mechanism for obtaining discriminative features [10]. Empirically, they have shown to be an effective and scalable mechanism to represent information

Table 2: Dimensionality of the filter, based on training corpus

Corpus	*number of documents*	*features*
English & Chinese	200,000	3,737,556
English	400,000	6,264,128
English	200,000	4,477,527
English	50,000	2,246,516
English	100,000	3,207,268
English, no preprocessing	100,000	6,064,259
Chinese	100,000	656,150
TREC fully trained classifier	530,000	8,626,863
TREC weakly trained classifier	530,000	4,092,195

contained in N adjacent tokens. A window of fixed width, in our case $N = 5$, slides over the sequence of tokens. For each window position, the set of all two-elementary combinations of the N tokens is generated. Each combination is constrained to always include the left-most token and zero or one other token; the remaining tokens in the window are replaced by a special *skip* symbol "$\circ$". The resulting N *orthogonal sparse bigram* combinations uniquely represent the content of the current window position.

For example, the window containing the sequence "All medications at low price", is represented by the *orthogonal sparse bigram features* ⟨All⟩, ⟨All medications⟩, ⟨All $\circ$ at⟩, ⟨All $\circ\circ$ low⟩, and ⟨All $\circ\circ\circ$ price⟩. The combination of these five features represents the entire content of the window. Information is lost, however, when the *orthogonal sparse bigram* features of the entire message are pooled into a single feature vector. Thus, *orthogonal sparse bigrams* implement an appealing and empirically proven trade-off between scalability and representational richness. Table 2 displays the number of features that the filter employs, based on the training corpus. Between 600,000 and 8.6 million orthogonal sparse bigram features have nonzero weights.

2.4 Large Training Corpora

The high dimensionality of the feature representation and the required high accuracy of the resulting classifier call for an extremely large training corpus. In this section, we want to clarify to which extent the winnow algorithm can benefit from additional training data and which amount of data is tractable. Therefore we trained four classifiers on between 50,000 and 400,000 emails of the English corpus (Table 1). Another 100,000 emails were kept for testing. Table 3 and Figure 1 show the results of the experiments. They confirm the linear runtime behavior of Winnow as well as a significant improvement of accuracy when using larger training sets. It can be seen that the number of training examples has a much greater impact on classification performance than preprocessing.

Table 3: Impact of training set size on classifier performance and training time

Training set size	*AUC*	*Max f-measure*	*Spam precision at 0.1% non-spam error*	*Training time in s*
50,000 emails	0.981703	0.925522	73.26 %	2,269
100,000 emails	0.987880	0.927897	82.17 %	4,658
200,000 emails	0.991310	0.932474	78.53 %	8,233
400,000 emails	0.994391	0.937107	84.25 %	19,284

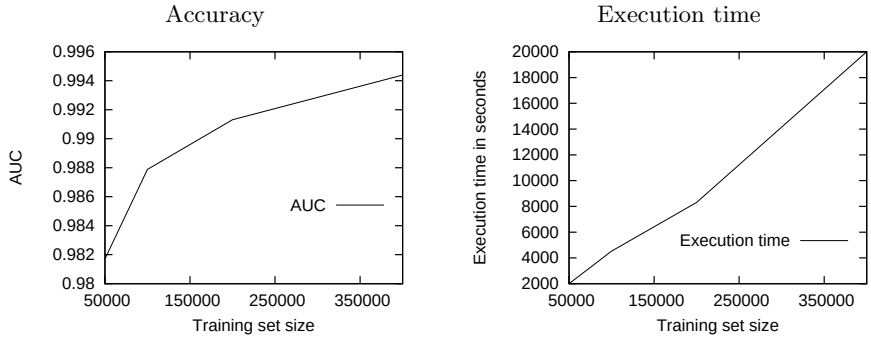

Figure 1: Plot of classifier quality (left) and training time (right) depending on the size of the training set

2.5 Differences in Distributions Underlying Training and Test Data

Most machine learning methods assume that training data be governed by the exact same distribution that the classifier is exposed to at application time. In the spam filtering application, control over the data generation process is less perfect. A number of public sources of spam messages and a much more limited number of sources of non-spam emails are available. On the other hand, American, Chinese, professional, recreational and many other groups of email users receive emails from a range of diverging distributions. Questions arise about the effect of such divergence between training and testing data on the classifier, and about the optimal way of dealing with this divergence. For a discussion of these generally under-studied questions, see [1].

It is relatively easy to identify the language used in a message. We study the impact that training a classifier on an English, Chinese, or mixed corpus will have on its performance on English, Chinese, or mixed testing data. Our goal is to obtain guidance on the optimal training corpus for a classifier that may be exposed to additional, unforeseen languages. We would also like to know whether it is advisable to use a joint classifier, or separate classifiers that perform well for only one language.

We perform several experiments using a Chinese and an English email corpus containing 200,000 emails each. After splitting the data in 50% training and 50% test data we train three classifiers using both data sets and a mixed corpus. The experimental results, given in Table 4, indicate that a classifier trained on emails of both languages performs very similar to the monolingually trained classifiers on each individual language.

Recently, the compensation of *sample selection bias* is being studied (e.g., [1]). In order to explicitly account for a divergence between training and testing data, the (unlabeled) testing data has to be available at training time. Unfortunately, this is not the case in the SpamTREC challenge.

From our experiments, we conclude that in the absence of unlabeled testing data, a classifier trained on a heterogeneously mixed corpus is generally preferable, because it does not lose accuracy on the individual languages while gaining additional robustness. We therefore compile a training corpus from maximally heterogeneous sources. It includes mailing lists, newsletters, several distinct spam traps, personal emails, moderated usenet groups, public email corpora such as the Enron dataset, and many more. For details, see Section 3.1.

Table 4: Performance of classifier depending on language

	Chinese test data		English test data	
Training data	*AUC*	*Max f-measure*	*AUC*	*Max f-measure*
Chinese	0.999880	0.999314	0.852904	0.831978
English	0.986990	0.957835	0.999477	0.996775
Both	0.999884	0.998543	0.999464	0.996166

3 TREC 2006 Spam Track Tasks

The TREC 2006 Spam Track consists of two different tasks. In the *online filtering task*, the spam filter classifies each message from the test corpus, and subsequently receives the true label of each message for training. There are two sub-tasks, the first with immediate feedback, and the second with delayed feedback; for the second sub-task, training is carried out after a randomly sized chunk of messages has been processed. In the *active filtering task*, the spam filter gets to choose a number of emails for which the label is then disclosed.

All tasks are carried out on four distinct evaluation corpora. There are two private corpora, tagged *B2* and *X2*, one public corpus with English emails and one public corpus with Chinese messages. A summary of the corpus sizes is shown in Table 5.

Table 5: Evaluation corpora

Dataset	*#Non-spams*	*#Spams*
B2 (private)	9274	2751
X2 (private)	9039	40135
English (public)	12910	24912
Chinese (public)	21766	42854

The rules of the Spam Track allow each contestant to submit four different filter configurations for each of the two main tasks. In the following, we describe the configurations of our entries, and the results on the different evaluation corpora.

3.1 Pre-Training

To take advantage of the capability of the winnow algorithm to handle large sets of known features and large training corpora, we assemble a corpus of emails to pre-train our filter prior to submission. Table 6 gives an overview of the sources of our training emails and their numbers of spam and non-spam emails.

Extensive pre-training imposes a risk if the chosen training data only poorly reflects the distribution at application time. Therefore we submit one weakly trained filter configuration for each task. For this, only a single iteration of the Winnow algorithm over the data is exercised, instead of re-iteration until convergence.

3.2 Online Classification with Periodical Re-Training

The winnow algorithm performs best when the model is trained by iterating several times over all training emails. However, all TREC tasks are designed to expose the filter to each training email only once. To overcome this discrepancy, we experiment with several strategies to cache all seen training emails and re-train the classifier on them periodically. Our experiments on different corpora show a notable influence of the re-training strategy on overall accuracy, but

Table 6: Training corpus composition

Source	*#Non-spams*	*#Spams*
CCERT Data Set (www.ccert.edu.cn/spam)	9272	30390
Disclosed Enron mails [6]	14000	0
Guenter spam trap (untroubled.org/spam)	187	49983
IMC mailing lists (www.imc.org)	14000	0
Various newsletters	18000	0
Nazario corpus (monkey.org/∼jose/phishing)	0	1000
SpamArchive.org	671	29938
SpamAssassin collection	1994	6
TREC 2005 corpus	30000	0
Various moderated Usenet groups	80000	0
Wouters archive (www.xtdnet.nl/paul/spam)	23	5000
Own collection	35682	209854

no single strategy can outperform the others consistently. Therefore, we choose to vary the strategies over the four allowed submissions. We include two submissions where a re-training is executed after each misclassified email, one configuration with re-training after every 500th email, and one submission with no re-training at all.

The results on the four evaluation datasets in Table 7 show that the periodical re-training is indeed successful in improving the performance in the online setting. On all but one dataset the configuration without re-training performs worst.

The private B2 corpus seems to be the most difficult dataset with the most mistakes made. Presumably this is due to the corpus differing strongly from our training data. This assumption is also consistent with the second configuration performing best on this dataset. The weak pre-training allows it to faster adapt to the different evaluation set.

Table 7: AUC results for the online filter tasks, on the two private corpora (B2, X2), the public English (E), and the public Chinese (C) corpus

Number	1	2	3	4
Pre-Training	Full	Weak	Full	Full
Re-train after	Mistakes	Mistakes	Never	Every 500th
B2 immediate	0.994705	**0.995724**	0.993775	0.994223
B2 delayed	0.994221	**0.994694**	0.991896	0.993216
X2 immediate	**0.998820**	0.998550	0.997898	0.998615
X2 delayed	**0.998641**	0.998173	0.997706	0.998237
E immediate	**0.998690**	0.998306	0.998436	0.998671
E delayed	**0.998582**	0.997048	0.998042	0.997994
C immediate	0.999762	0.999727	0.999646	**0.999767**
C delayed	**0.999681**	0.999631	0.999505	0.999670

Figures 2 (for immediate feedback) and 3 (for delayed feedback) show learning curves for all participants of SpamTREC, aggregated over all (public and private) evaluation corpora. The data has kindly been provided Godon Cormak. For all participants, the best of all submitted filters was used. When few or no training examples have been provided, the AUC mesaure of our filter exceeds the AUC of other submitted systems in both cases. For larger training samples, the filter falls back behind other submissions. We believe that this behavior is at

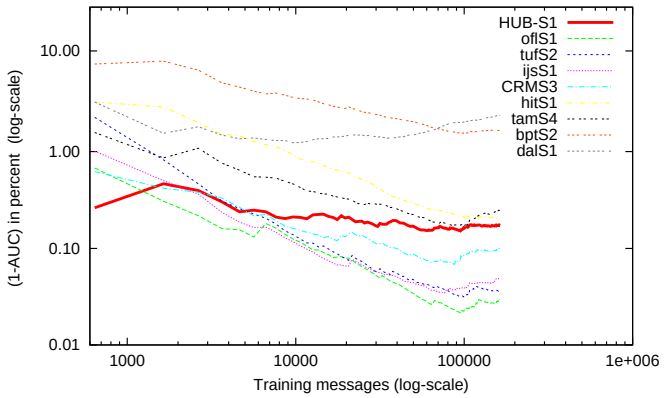

Figure 2: Results of the evaluation aggregated over all corpora for immediate feedback.

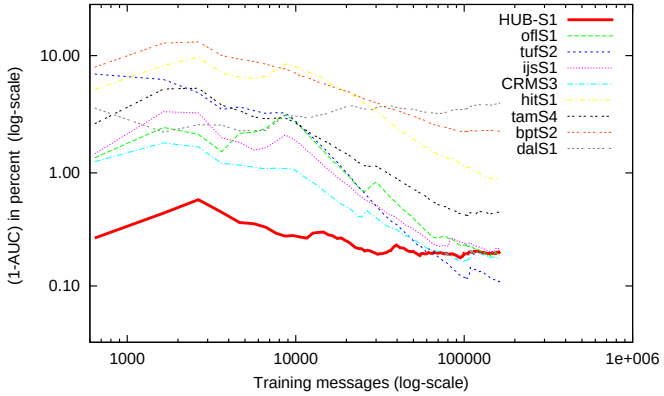

Figure 3: Results of the evaluation aggregated over all corpora for delayed feedback.

least partly explained by the large-scale pre-training that we used for all of our submissions—even the weakly trained filter was pre-trained with many emails. Pre-training leads to a better performance from the start but slows down the adaptation to the evaluation corpora. A careful comparative evaluation of all submitted filter is provided by [3].

3.3 Active Learning Task

In the Active Learning task, the spam filtering system has to decide which emails from a given set are to be labeled for training. There are various possible methods to select the next email. The idea behind most of them is to select those training emails which are expected to provide the most knowledge about the class labels of the test emails. One standard approach is uncertainty sampling [8], where one selects the training item which lies closest to the current decision boundary. This captures the intuition that an update of the decision boundary with this item has the largest effect on the shape of the decision regions, and therefore the greatest knowledge gain.

Besides uncertainty sampling, we evaluate several ad-hoc strategies, such as selecting the email with the highest number of unknown features, or selecting the email with the best feature

coverage over all available training mails. But our experiments show that none of them can outperform uncertainty sampling. Therefore, three of our four submissions for Task 2 use this standard method. The remaining, weakly trained configuration uses random sampling, because uncertainty sampling is unlikely to yield good results if the distance to the decision boundary depends only on very few update steps.

In contrast to our preliminary experiments, uncertainty sampling does not consistently outperform random sampling on the evaluation data of the competition. In combination with the periodical re-training strategy, the spam classifier tends to degrade to a highly imbalanced state, yielding almost useless spam scores. Apparently the selection of some disadvantageous training emails shifted the score of a large portion of spam emails after re-training far below the decision threshold.

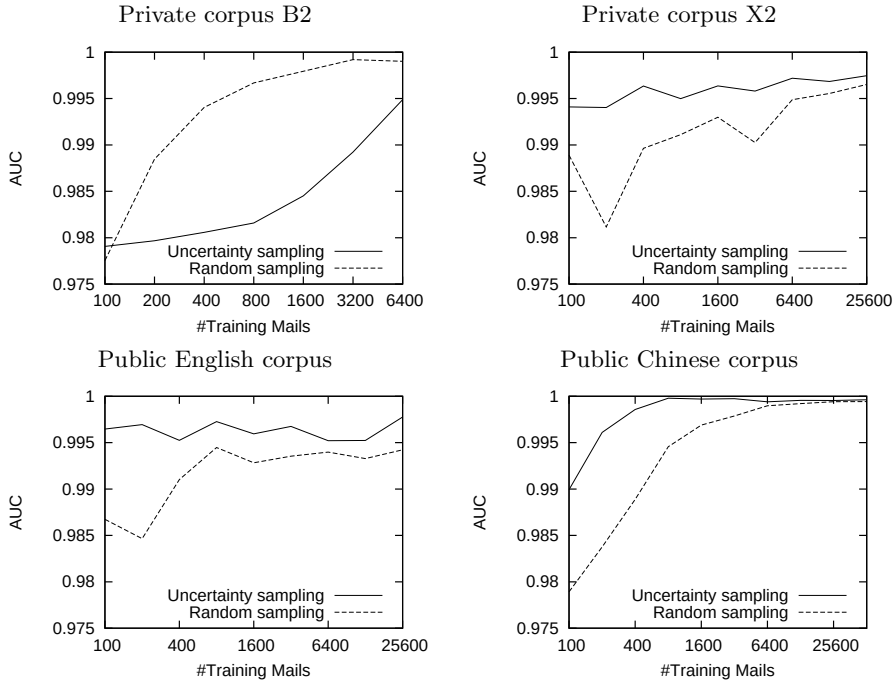

Figure 4: Results on the active learning task

This problem does not occur with our two submitted configurations without pre-training, one with uncertainty sampling and the other with random sampling. In Figure 4 one can again see that the *B2* corpus has different characteristics than the others. As one would expect on an evaluation corpus which is highly different from the training corpus, the uncertainty sampling strategy performs worse than random sampling due to strong initial fluctuations of the decision boundary.

4 Conclusion

The winnow classifier in conjunction with orthogonal sparse bigram features proves to be highly scalable and capable of efficiently handling, and benefiting from, hundreds of thousands of training messages and millions of features. Our experiments emphasize the importance of large,

diverse training sets; accuracy and robustness still improve when the training data is already very large. The accuracy of the classifier is further improved, and the resulting dimensionality of the classifier reduced, by character set decoding and normalization.

For the incremental SpamTREC tasks, using the cache of all previously seen messages in each model update step outperforms an update based on just the newly seen mail. In both tasks, the weakly trained classifier managed to adapt faster to the apparently most difficult evaluation corpus, as we expected. We make an interesting observation with respect to the benefit of uncertainty sampling versus random sampling. Uncertainty sampling is beneficial for the public, but detrimental for the private corpora that apparently deviate from the training data more strongly. Negative results for uncertainty sampling are rare in the literature, possibly because usually the case of identically distributed training and testing data is studied.

Recently, image spam is challenging spam filters. Even state-of-the-art filters are nearly helpless as the number of image spam messages explodes. Visual data requires different feature extraction procedures; the efficiency of the processing steps is crucial.

It would be interesting for future SpamTREC competitions to include evaluation corpora with emails from many different users with complete, unmodified headers. This would permit non-text based approaches to spam filtering, such as classification based on social networks or information about the sending servers. To explore the benefits of collective classification approaches, an additional task could expose the spam filters to more than one email at once instead of one at a time, and thus allow the incorporation of relationships between each other.

Acknowledgment

This work was supported by a Grant from STRATO AG. T.S. is supported by Grant SCHE 540/10-2 of the German Science Foundation DFG. We wish to thank Gordon Cormack for his great support!

References

[1] Steffen Bickel and Tobias Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems*. MIT Press, 2007.

[2] William W. Cohen. Learning to classify English text with ILP methods. In *Advances in Inductive Logic Programming*, pages 124–143. IOS Press, 1996.

[3] Gorgon Cormack. Trec 2006 spam treck overview. In *Proceedings of the TREC Text Retrieval Conference*, 2007.

[4] Claudio Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

[5] Paul Graham. Better bayesian filtering. In *Proceedings of the MIT Spam Conference*, 2003.

[6] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Proccedings of the 15th European Conference on Machine Learning*, volume 3201/2004, pages 217–226. Springer Verlag, Berlin/Heidelberg, 2004.

[7] Kim Larsen. Generalized naive bayes classifiers. *SIGKDD Explorations Newsletter*, 7(1):76–81, 2005.

[8] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann Publishers, San Francisco, 1994.

[9] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[10] Christian Siefkes, Fidelis Assis, Shalendra Chhabra, and William S. Yerazunis. Combining winnow and orthogonal sparse bigrams for incremental spam filtering. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 410–421. Springer Verlag, New York, 2004.