A Two-Pass Statistical Approach for Automatic Personalized Spam Filtering

Khurum Nazir Junejo, Mirza Muhammad Yousaf, and Asim Karim

Dept. of Computer Science, Lahore University of Management Sciences
Lahore, Pakistan
{junejo, 05030019, akarim}@lums.edu.pk

Abstract. Typically, spam filters are built on the assumption that the characteristics of e-mails in the training dataset is identical to those in individual users' inboxes on which it will be applied. This assumption is oftentimes incorrect leading to poor performance of the filter. A personalized spam filter is built by taking into account the characteristics of e-mails in individual users' inboxes. We present an automatic approach for personalized spam filtering that does not require users' feedback. The proposed algorithm builds a statistical model of spam and non-spam words from the labeled training dataset and then updates it in two passes over the unlabeled individual user's inbox. The personalization of the model leads to improved filtering performance. We perform extensive experimentation and report results using several performance measures with a discussion on tuning the two parameters of the algorithm.

1 Introduction

E-mail is an indispensable communication method for most computer users and it plays an essential role in the functioning of most businesses. Globalization has resulted in an exponential increase in the volume of e-mails. Unfortunately, a large chunk of it is in the form of spam or unsolicited e-mails. Last year 40% of all e-mails were spam, which totals up to 12.4 billion messages per day and about 176 messages per user per day [1]. In 2002, spam cost non-corporate Internet users 255 million dollars and resulted in a loss of 8.9 billion dollars to U.S. corporations alone [1]. Spam messages not only waste users' time and money but are also harmful for their computer's security. Commtouch, a security service provider, reported 19 new e-mail borne viruses in the month of January 2006 [2]. E-mail users spend an increasing amount of time reading messages and deciding whether they are spam or non-spam and categorizing them into folders. Some e-mail clients require users to label their received messages for training local (or personalized server-based) spam filters. E-mail service providers would like to relieve users from this burden by installing server-based spam filters that can classify e-mails as spam automatically and accurately without user feedback.

Typically, server-based spam filters are trained on general training datasets and then applied to individual users' inboxes. However, the characteristics of individual

users' inboxes are usually not identical to that of the general e-mail corpus used for training the spam filter, resulting in poor filtering performance. Furthermore, the characteristics of spam e-mails evolve with time making non-adaptive filters less robust to change. Thus, there is a need for personalized spam filters that learn from general training datasets and adapt to the characteristics of individual users' inboxes. This adaptation must be done without asking the users to label their e-mails. Earlier works on personalized spam filtering utilize users input. This approach is clearly not convenient for the e-mail user.

In this paper, we present a statistical approach for classifying individual users' e-mails in accordance with the users' e-mails characteristics without requiring their input. The algorithm learns from a general corpus of labeled e-mails in a single pass over them and then updates this learned model in two passes over the individual users' unlabeled e-mails. This approach allows automatic specialization of the general model to the underlying distribution of e-mails in individual users' inboxes. The statistical model is built from the tokens in the e-mail body and their frequencies. Our initial implementation of the algorithm won the performance award at the Discovery Challenge held in conjunction with ECML-PKDD [3]. This paper presents the detailed results of our implementations using several performance metrics and with varying parameters of the algorithm.

The rest of the paper is organized as follows. In section 2, we provide a brief review of content-based spam filters with specific focus on personalized spam filtering. Section 3 describes our algorithm for automatic personalized spam filtering. Section 4 describes the experiments and their results, and a discussion of the results with specific focus on parameter tuning is given in section 5. We conclude in section 6.

2 Personalized Content-based Spam Filtering

Many approaches are used in practice to control the menace of spam including global and local blacklists, global and local whitelists, IP blocking, legislation, and content-based filtering. Content-based filters employ machine learning techniques to learn to predict spam e-mails given a corpus of training e-mails. Such filters are typically deployed on the mail server that filters e-mails for all users of the server. Researchers have developed content-based spam filters using Bayesian approaches [4-7], support vector machines (SVM) [8, 9], nearest neighbor classifiers [10], rule-based classifiers [11, 12], and case-based reasoning [13]. Among these techniques, Bayesian approaches and SVMs have shown consistently good performances. Sahami et al. present one of the earliest naïve Bayes classifier for the spam classification problem [4]. Since then, numerous variations of the naïve Bayes classifier have been presented for spam filtering [5-7]. The popular Mozilla's e-mail client implements a naïve Bayes classifier for spam filtering [6]. Support vector machine (SVM) is a powerful supervised learning paradigm based on the structured risk minimization principle from computational learning theory. SVMs exhibit good generalization capabilities and have shown good spam classification performance. One of the first SVM for the

spam classification problem is presented in [8]. Since then, several extensions and variations have been presented such as [9].

The majority of the supervised machine learning techniques presented for spam filtering assume that e-mails are drawn independently from a given distribution. That is, the statistical distribution of e-mails in the training dataset is identical to that of the individual user's e-mails on which the trained filter will be applied. This assumption, however, is usually incorrect in practice; the training dataset is typically derived from multiple Internet sources reflecting different distributions of spam and non-spam e-mails that are different from that of the individual user's e-mails. A personalized spam filter is capable of adapting to the distribution of e-mails of each individual user. Previous works on personalized spam filtering have relied upon user feedback in the form of e-mail labels from each individual user [14, 15]. This strategy burdens the e-mail user with the additional task of aiding the adaptation of the spam filter. Recently, the problem of automatic personalized spam filtering is investigated by Bickel and Scheffer [16]. They present a nonparametric hierarchical Bayesian model that generalizes across several users' e-mails by minimizing a loss function. Their experiments indicate performance improvements over classifiers developed by assuming independent and identically distributed data.

We present a simple approach for automatic personalized spam filtering that does not require users' feedback. The approach is based on a statistical model of spam and non-spam words in e-mails similar to that developed in Bayesian approaches. However, unlike many Bayesian approaches presented in the literature, we specialize the model to reflect the distributions of e-mails in individual users' inboxes.

3 Our Algorithm

Our personalized spam filtering algorithm consists of two phases of processing. In the first phase, called the training phase, the algorithm learns a statistical model of spam and non-spam words from the training dataset in a single pass over the training dataset. The second phase, called the specialization phase, consists of two passes over the user's inbox (evaluation dataset). In the first pass, the statistical model developed in the training phase is used to label the e-mails in the individual user's inbox, and to build an updated statistical model of the e-mails. In the second pass, the updated statistical model is used to score and classify the e-mails in the individual user's inbox. The pseudo-code of our algorithm is given in Figure 1.

The statistical model is developed as follows: For each distinct word in the labeled (i.e. training or after first pass of evaluation) dataset, count its occurrences in spam and non-spam e-mails. Then, find the difference of these two values for each word. Now choose the significant words by selecting only those words for which the absolute difference between their spam and non-spam occurrences is greater than some integer threshold t. This approach also categorizes the significant words as either a spam word or a non-spam word. Each spam and non-spam word is assigned a weight based on the ratio of its occurrences in the spam and non-spam e-mails. This statistical model of words can then be used to classify a given e-mail by computing its spam score and non-spam score values, where the spam score (non-spam score) of an

N = Total number of e-mails
N_S = Number of spam e-mails
N_N = Number of non-spam e-mails
D = number of words in dictionary (indexed from 1 to D)
C_{Si} = count of word i in all spam e-mails
C_{Ni} = count of word i in all non-spam e-mails
Z_S = set of significant spam words
Z_N = set of significant non-spam words
t = threshold (e.g. $C_{Si} - C_{Ni} > t$ for it to be included in significant spam words)
s = scale factor
W_{Si} = weight associated with significant spam word i
W_{Ni} = weight associated with significant non-spam word i
T_i = word i

Training Phase (Phase 1 on Training Dataset)
Build_Statistical_Model Procedure
-For each distinct word i in dataset find C_{Si} and C_{Ni}
-Find the significant spam words Z_S such that for each word T_i in Z_S, $C_{Si} - C_{Ni} > t$
-Find the significant non-spam words Z_N such that for each word T_i in Z_N, $C_{Ni} - C_{Si} > t$
-For each significant spam and non-spam word find their weight as follows:
-$W_{Si} = [C_{Si} / C_{Ni}] * [N_N / N_S]$, for all words in Z_S
-$W_{Ni} = [C_{Ni} / C_{Si}] * [N_N / N_S]$, for all in Z_N

Specialization Phase (Phase 2 on Evaluation Dataset)
First Pass
Score_Emails Procedure
-For each e-mail in the evaluation dataset
-spam_score = $\sum W_{Si}$ (sum is over all significant spam words in e-mail)
- nonspam_score = $\sum W_{Ni}$ (sum is over all significant non-spam words in e-mail)
-If (s * spam_score > nonspam_score) then classify as spam and output spam_score; other wise classify as non-spam and output -nonspam_score.

-Build statistical model concurrently with scoring e-mails (procedure is identical to Build_Statistical_Model given above)

Second Pass
-Score and classify e-mails using the updated statistical model (procedure is identical to the Score_Emails procedure given above)

Fig. 1. Our automatic personalized spam filtering algorithm

e-mail is the weighted sum of the words of that e-mail that belong to the significant spam (non-spam) words set. If the spam score multiplied by a scaling factor (*s*) is greater than the non-spam score then the e-mail is labeled as spam; otherwise, it is labeled as non-spam. This statistical model is developed in the training phase as well

as in the first pass of the specialization phase. In the second pass of the specialization phase, the final scores and classifications of e-mails are output.

The motivation for using significant words that have differences of their counts in spam and non-spam e-mails greater than a specified threshold is: (1) a word that occurs much more frequently in spam e-mails (or non-spam e-mails) will be a better feature in distinguishing spam and non-spam e-mails than a word that occurs frequently in the dataset but its occurrence within spam and non-spam e-mails is almost similar, and (2) this approach greatly reduces the number of words that are of interest, simplifying the model and its computation. The scale factor is used to cater for the fact that the number of non-spam words, and their weighted sum in a given e-mail, is usually greater than the number of spam words and their weighted sum.

The purpose of the weighting scheme for the significant words is to give an advantage to words for which either the spam count or the non-spam count is proportionally greater than the other. For example, if the word with ID '10' has spam and non-spam counts of 0 and 50, respectively, and the word with ID '11' has spam and non-spam counts of 950 and 1000, respectively, then even though their difference is the same (50) the word with ID '10' gives more information regarding the classification of the e-mail than word with ID '11'.

The specialization phase adapts the general statistical model to the characteristics of the individual user's inbox. The model developed from the training phase is used for the initial classification of the user's e-mails. Furthermore, the statistical model is updated to incorporate the characteristics of the user's inbox. This updated model is then used to finally score and classify the e-mails in the user's inbox.

4 Experiments

In this section, we present the results of our experimental evaluation of the automatic personalized spam filtering algorithm. The algorithm is implemented in Java. The code uses special built-in data structures of Java such as Hash Maps that provide an efficient way of retrieving word objects by avoiding the cost of searching through an array list of word IDs.

4.1 Datasets

We use the datasets available from the ECML-PKDD Discovery Challenge website [3]. The training dataset is a general corpus containing 4000 e-mails collected from several users' inboxes. The evaluation datasets consist of three different users' inboxes each containing 2500 e-mails. These evaluation datasets are identified as Eval-00, Eval-01, and Eval-02. In addition to these datasets, a training and a test (labeled evaluation) dataset containing 4000 and 2500 e-mails, respectively, is provided for parameter tuning. The ratio of spam and non-spam e-mails in all the datasets is 50-50. The distribution of e-mails in the training corpus which is a combined source of training data is different from the distributions of the e-mails received by individual users. An additional evaluation dataset is created from the

Table 1. Characteristics of datasets

	Distinct Words	**Total Words**
Training Corpus	41675	2761246
Eval-00	26580	842998
Eval-01	27523	843634
Eval-02	20227	494536
Eval-Combined	39962	2181168
Tune-Training	39967	2915199
Tune-Test	22991	1197283

above datasets. Dataset Eval-Combined is the collection of e-mails in datasets Eval-00, Eval-01, and Eval-02.

Each e-mail in the datasets is represented by a word (term) frequency vector. Each word in an e-mail is identified by an ID and its frequency count in the e-mail. An additional attribute identifies the label of the e-mail as either spam or non-spam.

4.2 Evaluation Criteria

We evaluate our algorithm using the following performance measures:

1. True positive rate (TP): fraction of spam e-mails correctly classified as spam
2. False positive rate (FP): fraction of non-spam e-mails incorrectly classified as spam
3. Accuracy: fraction of all e-mails that are correctly classified
4. Precision: TP / (TP + FP)
5. Recall: TP / (TP + FN), where false negative rate (FN) is the fraction of spam e-mails that are incorrectly classified as non-spam
6. AUC: area under the receiver operating characteristics (ROC) curve

The first five measures are calculated by taking zero as the discriminating threshold in the scores output by a given filter. The AUC is computed from the ROC generated by sweeping through all possible discriminating thresholds in the scores output by a given filter. The AUC is considered to be a better measure of the overall performance of a filter [17].

4.3 Results

We ran our algorithm by iterating over several values of the threshold (t) and scale factor (s) parameters to find the parameter combination that produces the best performance. The parameters that yield the highest AUC value on the tuning datasets are $t = 8$ and $s = 13$. With these parameters, the performance of our algorithm on the evaluation datasets (users' inboxes) is given in Table 2. The average AUC value for these 3 inboxes is 0.9875. This value is significantly better than 0.9539 obtained by the filter submitted for the Discovery Challenge. The submitted filter used the

Table 2. Performance results of our algorithm with parameters $t = 8$ and $s = 13$ tuned on the tuning datasets

	AUC	Precision (%)	Recall (%)	Accuracy (%)	FP (%)	TP (%)
Eval-00	0.9832	85.53	98.88	91.08	16.72	98.88
Eval-01	0.9896	91.87	98.56	94.92	7.76	98.08
Eval-02	0.9898	98.68	90.24	94.52	1.2	90.24
Average	0.9875	92.02	95.89	93.50	8.56	95.73

parameters $t = 400$ and $s = 8$. These parameters were selected after a few iterations over the tuning datasets. Furthermore, at that time we preferred a large value for t to keep the size of the significant words set small.

We have now experimented with hundreds of parameter combinations by tuning them on the individual evaluation datasets. The performance results of some selected parameter combinations are given in Table 3. The highest AUC value (the 'optimal' filter) for each evaluation dataset is highlighted. It is seen that whenever the AUC value is the highest the corresponding accuracy is also the highest. The filter with threshold value of 400 is the one that was submitted to the Discovery Challenge. In general, the AUC value increases slightly as the threshold increases from zero and then starts decreasing with further increase in the threshold. This is because as the threshold increases the number of significant words decreases to an 'optimal' feature set and then, with further increase in the threshold, the significant words become less discriminatory. For the three evaluation datasets we suggest a threshold of around 10 as this produces a reduction of the total words by more than half at the expense of about 0.3% loss in accuracy. The tradeoff between the accuracy and threshold is discussed in more detail in Section 5.

The key part of our algorithm is the specialization phase in which the model learned from the general corpus is updated in accordance to the characteristics of each individual user's inbox. The second pass of this phase is the step that significantly differentiates our algorithm from the conventional techniques that assume that e-mails are drawn independently from a given distribution. To highlight this personalization characteristic of our algorithm, we ran our algorithm with and without the second pass of the specialization phase. The results of this experiment are shown in Table 4 for the 'optimal' filters highlighted in Table 3. It is seen that the second pass produces a significant improvement in the AUC value because now the model is specialized for each user's inbox. Similarly, the second pass results in an increase of at least 10% in the accuracy on all evaluation datasets, which is a significant amount.

4.4 Results of Some Variations of the Algorithm

We also experiment with some variations of the algorithm given in Figure 1. In the first variation, instead of maintaining the frequencies of words in e-mails we use only their occurrences (i.e. count 1 if a word occurs in an e-mail and 0 otherwise).

Table 3. Performance results of our algorithm with various parameter combinations. Average AUC of the highlighted rows is 0.9902. Average AUC of the submitted filter (with $t = 400$) is 0.9539

	Threshold (*t*)	**Scale Factor (*s*)**	**AUC**	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**FP (%)**	**TP (%)**
Eval-00	0	9	0.9844	97.42	94	95.76	2.48	94
Eval-00	10	9.5	0.9845	96.80	94.64	95.76	3.12	94.64
Eval-00	**11**	**9.5**	**0.9848**	**96.81**	**94.88**	**95.88**	**3.12**	**94.88**
Eval-00	100	7.5	0.9816	95.55	91.2	93.44	4.24	91.12
Eval-00	400	8	0.955	86.94	93.2	89.6	14	93.2
Eval-01	0	11	0.993	94.96	96.56	95.72	5.12	96.56
Eval-01	**4**	**11.5**	**0.993**	**94.76**	**97.04**	**95.84**	**5.36**	**2.96**
Eval-01	10	11.5	0.9929	95.51	95.44	95.48	3.92	94.72
Eval-01	100	9	0.9821	95.60	90.48	93.16	4.16	90.48
Eval-01	400	8	0.9717	91.36	88.88	90.24	8.4	88.88
Eval-02	0	14.5	0.9924	97.02	96.56	96.8	2.96	96.56
Eval-02	**2**	**16**	**0.9928**	**96.29**	**97.68**	**96.96**	**3.76**	**97.68**
Eval-02	10	16.5	0.9918	96.19	97.12	96.64	3.84	97.12
Eval-02	100	14	0.9751	89.73	94.4	91.8	10.18	94.4
Eval-02	400	8	0.935	85.86	83.12	84.72	13.68	83.12

Surprisingly, this variation resulted in a significant increase in the AUC value for the datasets as shown in Table 5. It is seen from Table 5 that the 'optimal' average AUC value for this variation is 0.9932 as compared to 0.9902 for the original algorithm. In particular, the increase in the AUC value of dataset Eval-00 is quite substantial.

To compare the performance of a single classifier/filter for all users to that of personalized filters for each individual user, we experiment with a second variation of the algorithm. In this variation, we combine the 3 evaluation datasets into a single dataset Eval-Combined for which a single filter is built. This results in an increase in the AUC value. The average AUC value of the three evaluation datasets using the frequency counts comes out to be 0.9902 while the AUC value with the combined dataset comes out to be 0.9937. This improvement can be attributable to: (1) the larger size of the combined evaluation dataset, and (2) the fact that the general corpus is a collection of individual users' inboxes and thus resembles the dataset Eval-Combined. Table 5 also shows the performance of the filter based on word occurrences when applied to dataset Eval-Combined. This variation outperformed the rest with an AUC value of 0.9942. Thus, the best AUC value is obtained if we merge the evaluation datasets and use word occurrences rather frequencies for building the statistical model.

5 Parameter Tuning

Our algorithm uses two parameters: threshold (t) and scale factor (s). We ran hundreds of experiments with varying values for these parameters. The performance

Table 4. Performance results of the algorithm after the first and second pass over the evaluation dataset

	After First Pass				After Second Pass			
	AUC	Precision (%)	Recall (%)	Accuracy (%)	AUC	Precision (%)	Recall (%)	Accuracy (%)
Eval-00	0.9090	94.45	70.8	83.32	0.9848	96.81	94.88	95.88
Eval-01	0.9220	93.61	77.36	86.04	0.993	94.76	97.04	95.84
Eval-02	0.9346	89.38	82.88	86.52	0.9928	96.29	97.68	96.96
Average	0.9218	92.48	77.01	85.29	0.9902	95.95	96.53	96.22
Eval-Combined	0.9283	93.10	77.01	56.96	0.9937	96.79	96.66	96.73

Table 5. Performance comparison of algorithm using word frequecies and occurrences

	Based on Frequency Count	Based on Occurrence Count
Eval-00	0.9848	0.9920
Eval-01	0.9930	0.9941
Eval-02	0.9928	0.9936
Average	0.9902	0.9932
Eval-Combined	0.9937	0.9942

spread is seen in the ROC curve for dataset Eval-00 shown in Figure 2. The purpose of the threshold parameter is to find significant spam and non-spam words by filtering out the words that are not discriminatory enough. We experiment with several thresholds (t) to find the t that produces the highest accuracy and AUC value. For the three datasets, the best values for t are 11, 4, and 2, respectively. Figure 3 shows the variation of the threshold t with the number of significant words and the accuracy of the filter. It is observed that for small values of t there is little change in the accuracy of the filter, but the number of significant words in the model is greatly reduced. For example, by increasing t from 10 to 20 does not increase the accuracy significantly, but the number of significant words in the model is reduced by more than a factor of 2. This trade off between the number of significant words and the accuracy suggests a value for t that lies between 10 and 20.

The scale factor (s) is used to counter the effect of greater weight of non-spam words as compared to spam words in e-mails. For example, for dataset Eval-01, the average weight of a spam word is 4.0726, the average weight of a non-spam word is 13.1045, the average number of spam words in spam e-mails is 5.7192, and the average number of non-spam words in spam e-mails is 2.3296. Thus, for this dataset the average spam score of spam e-mails is 4.0726*5.7192 = 23.32 and the average non-spam score of spam e-mails is 13.1045*2.32 = 30.40, resulting in a large number of miss classifications. The scale factor is used to counter this by increasing the spam score by a factor s.

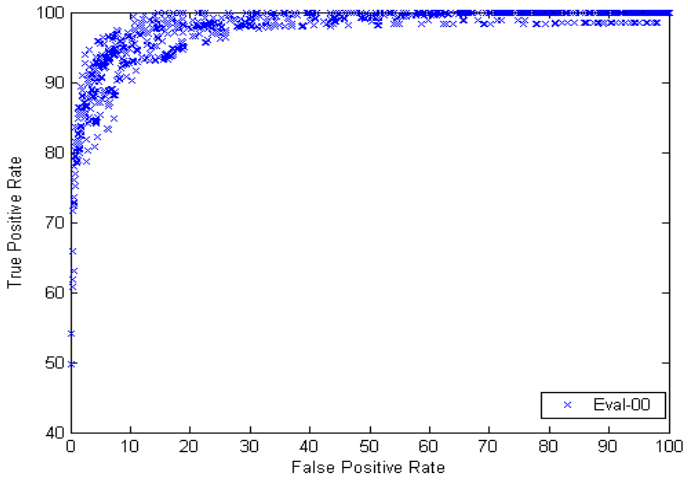

Fig. 2. ROC curve for dataset Eval-00

Based on the above observation, a reasonable way to estimate the scale factor for a dataset for which the ratio of spam-to-non-spam e-mails is known is as follows: adjust the scale factor until the ratio of spam-to-non-spam classification of the filter is equal to the known ratio. This approach can be applied to each individual user's inbox if the ratio of spam-to-non-spam e-mails is known.

6 Conclusion

We present a simple statistical algorithm for automatic personalized spam filtering that does not require users to provide feedback regarding the classifications of e-mails in their inboxes. The algorithm builds a statistical model of words from a training corpus and then adapts it to the distribution of words and e-mails in each individual user's inbox. Overall, the algorithm requires one pass over e-mails in the training corpus and two-passes over e-mails in the individual user's inbox. Our experiments confirm the benefit of personalization with significant performance gains over a filter that assumes that training and evaluation (users' inboxes) datasets follow the same distribution. We present extensive results of our algorithm including a discussion on the estimation of its two parameters.

The problem of automatic personalized spam filtering has generated much interest recently. It is a technically challenging problem that promises significant benefit to e-mail users and e-mail service providers.

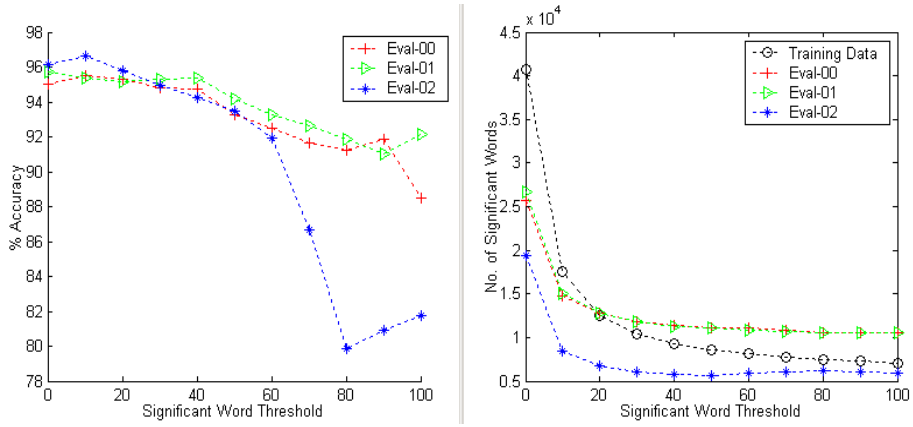

Fig. 3. Impact of threshold on accuracy and number of significant words

References

1. D. Evett: Spam statistics 2006. TopTenREVIEWS http://spam-filter-review.toptenreviews.com/spam-statistics.html (2006)
2. Commtouch: January virus and spam statistics: 2006 starts with a bang. Commtouch Press Release http://www.commtouch.com/Site/News_Events/pr_content.asp?news_id=602&cat_id=1 (2006)
3. ECML-PKDD: Discovery challenge. http://www.ecmlpkdd2006.org/challenge.html (2006)
4. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz: A bayesian approach to filtering junk E-mail. In Proc. of AAAI Workshop on Learning for Text Categorization. AAAI Technical Report WS-98-05 (1998)
5. P. Graham: Better Bayesian filtering. In Proc. of 2003 Spam Conference http://www.paulgraham.com/better.html (2003)
6. E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos: Filtron: a learning-based anti-spam filter. In Proc. 1st Conf. on Email and Anti-Spam (CEAS 2004) (2004)
7. A.K. Seewald: An evaluation of naive Bayes variants in content-based learning for spam filtering. Kluwer Academic Publsihing (2005)
8. H. Drucker, D. Wu, and V.N. Vapnik: Support vector machine for spam categorization. IEEE Transactions on Neural Networks 10(5) (1999) 1048–1054
9. A. Kolcz and J. Alspector: SVM-based filtering of e-mail spam with content-specific misclassification costs. In Proc. of the TextDM'01 Workshop on Text Mining (2001)

10. G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos: A memory-based approach to anti-spam filtering for mailing lists. Information Retrieval, Springer 6(1) (2003) 49-73
11. W.W. Cohen: Learning rules that classify e-mail. In Proc. of 1996 AAAI Spring Symposium in Information Access (1996)
12. E.C.J. Kay and E. McCreath: Automatic induction of rules for e-mail classification. In Proc. of the Sixth Australasian Document Computing Symposium (2001) 13–20
13. S.J. Delany, P. Cunningham, and L. Coyle: An assessment of case-based reasoning for spam filtering. Artificial Intelligence Review, Springer 24(3-4) (2005) 359-378
14. R. Segal. J. Crawford, J. Kephart, and B. Leiba: SpamGuru: an enterprise anti-spam filtering system. In Proc. of Conference on Email and Anti-Spam (CEAS ’04) (2004)
15. A. Gray and M. Haahr: Personalised collaborative spam filtering. In Proc. of Conference on Email and Anti-Spam (2004)
16. S. Bickel and T. Scheffer: Dirichlet-enhanced spam filtering based on biased samples. In Proc. of the Workshop on Learning with Nonparametric Bayesian Methods (with ICML’06) (2006)
17. C. Cortes and M. Mohri: AUC optimization vs. error rate minimization. Advances in Neural Information Processing Systems, NIPS (2004)