Filtron: A Learning-Based Anti-Spam Filter

Eirinaios Michelakis[1], Ion Androutsopoulos[2], Georgios Paliouras[1], George Sakkis[3], and Panagiotis Stamatopoulos[4]

[1] National Centre for Scientific Research “Demokritos”, Greece.
[2] Athens University of Economics and Business, Greece.
[3] Rutgers, the State University of New Brunswick, USA.
[4] University of Athens, Greece.

Abstract. We present Filtron, a prototype anti-spam filter that integrates the main empirical conclusions of our comprehensive analysis on using machine learning to construct effective personalized anti-spam filters. Filtron is based on the experimental results over several design parameters on four publicly available benchmark corpora. After describing Filtron's architecture, we assess its behavior in real use over a period of seven months. The results are deemed satisfactory, though they can be improved with more elaborate preprocessing and regular re-training.

1 Introduction

The proliferation of unsolicited commercial e-mail (UCE), more commonly known as spam, over the last few years has been undermining constantly the usability of e-mail. The availability of bulk mailing software and lists of e-mail addresses harvested from Web pages, newsgroup archives, and service provider directories allows messages to be sent blindly to millions of recipients at essentially no cost. Spam messages are extremely annoying to most users, as they clutter their mail-boxes and prolong dial-up connections. They also waste the bandwidth and CPU time of ISPs, and often expose minors to unsuitable (e.g. pornographic) content.

As of now, anti-spam filters – software tools that attempt to identify incoming spam messages automatically – seem to be the most viable solution to the problem. Most commercially available filters of this type currently appear to rely on simple techniques such as white-lists of trusted senders, black-lists of known spammers, and hand-crafted rules that block messages containing specific words or phrases. On the other hand, the success of machine learning techniques in text categorization [12] has led researchers to explore learning algorithms in anti-spam filtering. Previous research work on anti-spam filtering studied the performance of many popular machine learning algorithms, including Naive Bayes, C4.5, Support Vector Machines and boosting with C4.5 [10, 4, 2].

Following such results, learning-based anti-spam filters, mostly based on Naive Bayes, are becoming operational. This paper presents Filtron, a prototype anti-spam filter implementation that incorporates the key findings of our previous work. Filtron emerged as the result of our thorough investigation of learning approaches to anti-spam filtering [1].

The remainder of this paper is organized as follows: Section 2 reviews the related work on available anti-spam filters, focusing on learning-based and novel alternative approaches. Section 3 presents Filtron and section 4 summarizes the results of using Filtron on several benchmark corpora. Section 5 exhibits a real life assessment of Filtron's performance and finally in section 6 we conclude and suggest further directions.

2 Related Work

The majority of commercial anti-spam filters currently appear to rely on black-lists, white-lists and hand-crafted rules that search for particular keywords, phrases, or suspicious patterns in the headers. Black-lists of known spammers containing e-mail addresses or domain names are of little use, as spammers typically use fake sender addresses; a more effective approach maintains on-line databases of IP numbers that have or could be used by spammers (e.g., open SMTP relays)[1]. On the white-list side, some commercial filters send replies to senders not in the user's white-list, asking them to answer a simple question in order to rule out spamming robots. Since there is no learning component to admit messages whose content looks legitimate, this may lead to a frustrating proliferation of automatic answer-seeking replies.

Following research publications, the developers of anti-spam filters are becoming increasingly aware of the potential of machine learning. Consequently, there is a growing number of filter implementations that incorporate learning-based components, mostly Naive Bayes classifiers[2]. For example, an anti-spam filter based on a flavor of Naive Bayes was recently added to Mozilla's e-mail client. The filter is trained on legitimate and spam messages of its particular user, and can be configured to perform actions such as moving incoming messages suspected to be spam to a special folder. Mozilla's filter supports incremental learning, i.e., the user can correct the category of misclassified messages, and the probabilities of the Bayesian filter are adjusted accordingly, without retraining on the entire message collection. POPFILE is similar in that it also resides on the client computer, employs Naive Bayes, and supports incremental learning. However, it is a more general filter, which apart from detecting spam messages can also classify legitimate messages into different categories. It can also act as an e-mail proxy for POP3 servers and add tags to messages it considers spam. The tags can then trigger rules of the e-mail client. One of the most interesting characteristics of POPFILE is its elaborate preprocessor, which attempts to confront tricks spammers are beginning to use to confuse anti-spam filters; for example, inserting spaces, other characters, HTML comments and formatting tags between letters or substituting letters with visually equivalent digits or other symbols to fool tokenizers (as in "`g e t r*i*c*h f-a-s-t`", "`vi<!--45-->agra`", "`s<b>e</b>x`", "`d@.tef1nder`").

An interesting alternative to learning-based anti-spam filters is Vipul's Razor[3], a form of collaborative filtering, which relies on a network of servers that store signatures of spam messages. The users of the network report to its servers spam messages they receive, and at the same time they use screening software that compares their incoming messages to the spam messages other users have reported. To make the comparison fast, the reported spam messages are stored as signatures, in the simplest case a hash value for each message. When an incoming message arrives at the mailbox of a user, the screening software computes its signature and compares it to the signatures in the network's servers. If a match is found, the message is moved to a special folder. If a spam message escapes the matching, the user can report it to the network, and this helps other users avoid it. Unfortunately, spammers can defeat simple signatures by introducing random changes to the copies of the messages they send (e.g., inserting a random string), which causes the signatures to be different and the matching to fail. Hence, much of the work in this area is devoted to devising signatures that are insensitive to such changes.

In the long run, the spam problem is more likely to be solved by a combination of several techniques. The SpamAssassin filter is a good example of such a combination. It incorporates elaborate hand-crafted rules, white-lists, on-line black-lists, Vipul's Razor, and a Naive Bayes classifier. Users can add their own rules, white-lists, and black-lists, and they can retrain the Naive Bayes classifier, or modify a set of weights that specify how much SpamAssasin relies on each filtering technique. A genetic algorithm can also be used to generate the weights automatically.

3 Anti-spam Filtering with Filtron

Filtron is a fully functional learning-based anti-spam filter, whose design and implementation emerged as the result of our extensive exploration of learning approaches to anti-spam filtering [1]. It is based on Weka,

[1] See for example `http://www.mail-abuse.org/`, `http://www.ordb.org`, `http://www.spamhaus.org/`, and `http://www.spews.org/`.

[2] Naive Bayes spam filter implementations include Mozilla Mail (`http://www.mozilla.org/mailnews/spam.html`), POPFILE (`http://popfile.sourceforge.net/`), Mailfilter (`http://mailfilter.sourceforge.net/`), SpamAssassin (`http://www.spamassassin.org/`), K9 (`http://keir.net/k9.html`), Spammunition (`http://www.upserve.com/spammunition/`) and Bogofilter (`http://bogofilter.sourceforge.net/`).

[3] See `http://razor.sourceforge.net/`.

the open source machine learning platform[4]. Hereafter we discuss Filtron's architecture and illustrate how standard machine learning techniques can be embedded in operational filters.

Figure 1 depicts Filtron's architecture. The training subsystem tailors the filter to the particular user it is intended to protect, by treating messages previously received by the user as training examples. Thereafter, when a new message for the user arrives, the classification subsystem is invoked to classify it.

Filtron's training subsystem comes with its own collection of duplicate-free spam messages, which are used when users lack their own collection of spam messages they have received. We update this collection regularly, and it currently contains 2559 spam messages. Spam messages are typically sent blindly, without considering the professions, interests, etc. of the recipients. Hence, training Filtron on spam messages not received by its particular user does not affect substantially the quality of the filter. On the other hand, we always train Filtron on legitimate messages received by the user, because we have found that many of the most useful attributes of anti-spam filtering correspond to terms that are characteristic of the legitimate incoming messages of the particular user. When using Filtron's pool of spam messages, the user can choose the number of spam messages to be included in the training collection, to approximate the legitimate-to-spam ratio of incoming messages s/he is experiencing, which may lead to a more accurate classifier.

The corpus preprocessor scans the training messages and removes duplicates, attachments, and HTML tags. It can also replace tokens by unique numbers to make a collection of messages publicly available, safeguarding the privacy of their recipients and senders. The messages can be provided in a variety of formats, including folders of several popular email clients. Moreover, the preprocessor builds automatically a white-list with the addresses the user has received legitimate messages from, and a black-list with the addresses of the senders of all spam messages, though, as mentioned in Section 2, black-lists of this form are not very useful. The user can inspect and edit both lists. Also, it is worth noting that the preprocessor retains only the first five messages from each sender. This is justified by the fact that a large percentage of a user's incoming messages are from senders the user has regular correspondence with, i.e., individuals who are unlikely to send spam messages and their addresses are usually stored in the user's address book. The latter can act as an additional personal white-list: messages from senders in the address book can be categorized immediately as legitimate, without invoking the learning-based component of the anti-spam filter. Hence, for the training collection to be representative of the messages that the learning-based component will be faced with, messages from regular correspondents should not overwhelm the legitimate subset of the collection.

The attribute selector identifies the best attributes to include in the vector representation, since it is impractical in terms of computation and storage to consider the whole pool of candidate attributes. Attributes are selected according to the Information Gain measure [14], which has been shown to be very effective in practice. Filtron uses 1-gram attributes only, as our experimental results do not provide evidence on the benefit of employing n-grams for $n > 1$ [1]. The number of attributes to retain can be set at will. The optimal number of attributes, in terms of accuracy, training, and classification speed, will vary per user, and is hard to specify without user-specific experiments. Existing results, however, suggest that using a few hundreds of attributes is a good compromise.

Once the attributes have been selected, the vectorizer converts the training messages to vectors. All attributes are numeric, with the value of attribute X_i being defined as $t_i(d)/l(d)$, where $t_i(d)$ is the number of occurrences in document d of the token represented by X_i, and $l(d)$ is the length of d measured in token occurrences. The training vectors are then passed to the learning component, along with a value for the parameter λ, the relative cost of misclassifying a legitimate message as spam ($L \rightarrow S$) compared to misclassifying a spam message as legitimate ($S \rightarrow L$). In the learning component, the user can choose any of the learning algorithms provided by Weka. The default choice is SMO, a Support Vector Machine (SVM) implementation, which has demonstrated a good trade-off between accuracy and speed. The learning component produces a user-specific classifier that is saved along with the user's white and black list in the user's model.

Filtron is implemented in Java and Tcl/Tk and thus is executable on most platforms. The training component typically resides on the same machine with the user's e-mail client, while the classification component resides on the user's incoming e-mail server, and is currently compatible only with SMTP servers oriented towards the message delivery mechanism of UNIX. The classification subsystem intercepts the user's messages

[4] Weka is available from `http://www.cs.waikato.ac.nz/ml/weka/`.

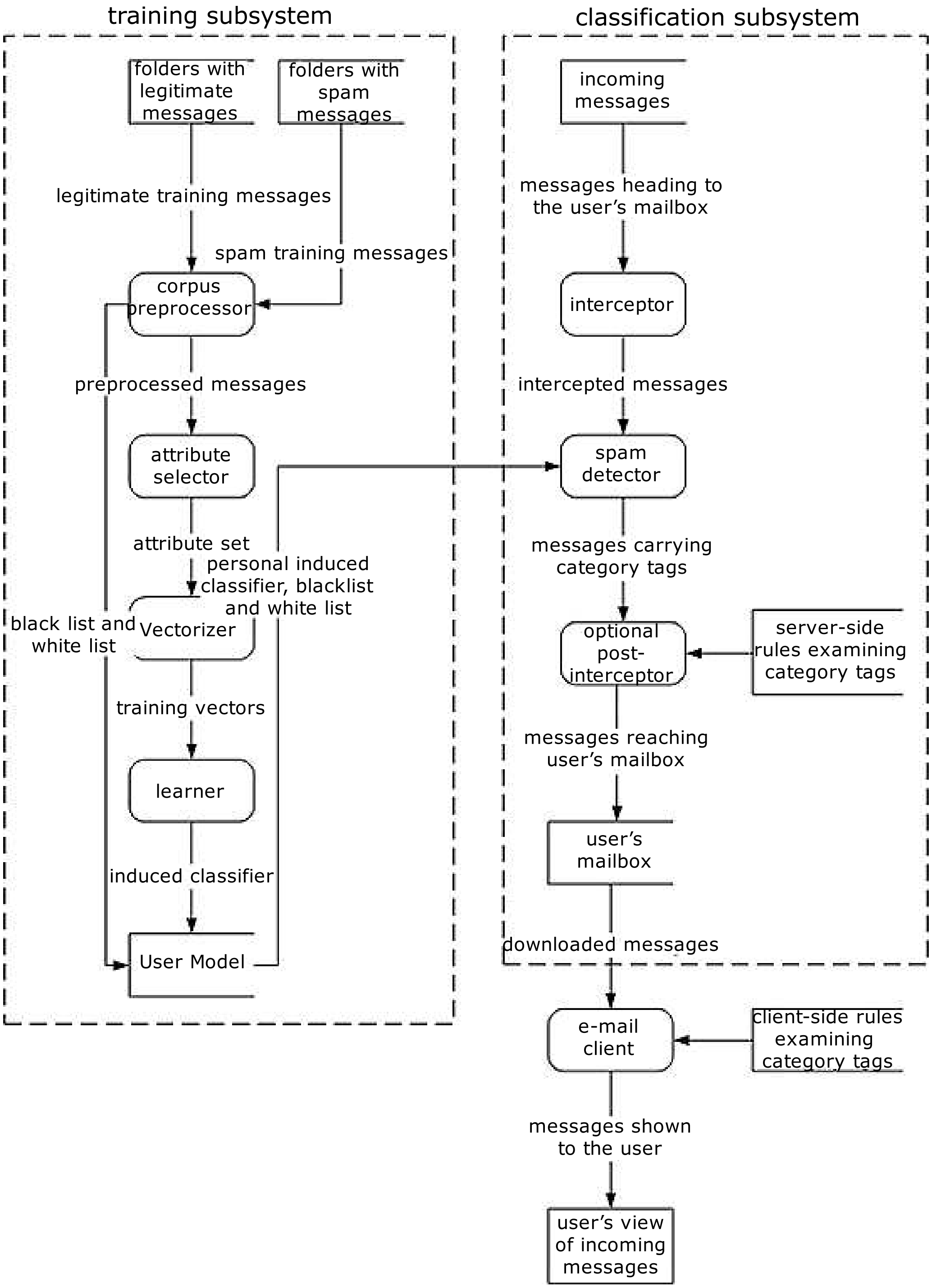
training subsystem
classification subsystem
folders with legitimate messages
folders with spam messages
legitimate training messages
spam training messages
corpus preprocessor
preprocessed messages
attribute selector
attribute set
personal induced classifier, blacklist and white list
black list and white list
Vectorizer
training vectors
learner
induced classifier
User Model
incoming messages
messages heading to the user's mailbox
interceptor
intercepted messages
spam detector
messages carrying category tags
optional post-interceptor
server-side rules examining category tags
messages reaching user's mailbox
user's mailbox
downloaded messages
e-mail client
client-side rules examining category tags
messages shown to the user
user's view of incoming messages

Fig. 1. Filtron's architecture.

before they reach her mailbox using Procmail[5], an e-mail processor that accompanies most UNIX distributions. The interceptor invokes Filtron's spam detector, passing it as inputs the incoming message and the user's model. If the sender is in the user's white-list or black-list, the spam detector classifies the message as legitimate or spam, respectively, without further processing. Otherwise, it relies on the decision of the user-specific classifier. In any case, the only effect of the spam detector is the addition of the prefix `[Spam?]` to the subject of the messages it considers spam.

In a typical configuration, the user writes rules in her e-mail client to identify messages that carry the `[Spam?]` prefix and delete them, move them to a special folder, change their priority, ask their sender to repost them, etc. Writing such rules is easy in most modern e-mail clients. Alternatively, a Procmail-based post-interceptor can be set up to intercept messages that carry the `[Spam?]` prefix to prevent them from being stored into the user's mailbox. The post-interceptor can be configured to perform similar actions as the rules of the e-mail client. Users that get their e-mail through low bandwidth lines may prefer this option, as it avoids downloading messages suspected to be spam. Periodically, the training subsystem can be re-invoked to adjust the classifier to changes in the content and wording of spam messages, and to exploit larger training collections the user may have accumulated.

4 In vitro evaluation

In [1], we looked thoroughly into several parameters of anti-spam filtering and studied experimentally their effect on four benchmark corpora, derived by different users. The parameters studied included attribute vector representation (n-grams for $n = 1, 2, 3$), attribute selection (frequency thresholding, Information Gain), size of the attribute and training set and four learning algorithms, namely Naive Bayes [8, 7], Flexible Bayes [6], Support Vector Machines [13, 9] and LogitBoost [3], a boosting algorithm on top of regression stumps as weak learners. All experiments were formulated and evaluated under a cost-sensitive framework, also described in the report. Table 1 summarizes some of the results that gave rise to the design of Filtron; the reader may consult the report for further details.

	$\lambda = 1$			$\lambda = 9$		
	Pr	Re	*WAcc*	Pr	Re	*WAcc*
1–grams						
Naive Bayes	90.56	94.73	94.65	91.57	92.17	94.87
Flexible Bayes	95.55	89.89	95.15	98.88	74.63	97.76
LogitBoost	92.43	90.08	93.64	97.71	74.89	97.24
SVM	94.95	91.43	95.42	98.12	78.33	97.60
1/2/3–grams						
Flexible Bayes	92.98	91.89	93.89	97.43	81.36	96.91
SVM	94.73	91.70	95.05	98.70	76.40	97.67

Table 1. Precision, Recall and Weighted Accuracy (%) of the learning algorithms considered, for two cost scenarios ($\lambda = 1, 9$ – $L \rightarrow S$ errors are considered λ times more costly than $S \rightarrow L$ ones) and attribute representations (1–grams and 1/2/3–grams). The results were averaged over all corpora produced for the study, retaining per corpus and algorithm the number of attributes that gave the best results.

5 In vivo evaluation

To explore how well a learning-based filter performs in real life, the third author used Filtron for seven months. The system was trained using the PU3 collection[6], a collection of 2313 legitimate messages of the third author

[5] Procmail is freely available from: `http://www.procmail.org/`.

[6] The PU1, PU2, PU3 and PUA corpora produced for the experiments of this study are publicly available at: `http://www.iit.demokritos.gr/skel/i-config/`.

and 1826 spam messages which were included in Filtron at that time. The filter was configured for the $\lambda = 1$ scenario, where messages suspected to be spam are simply flagged to help the user prioritize the reading of incoming messages. An e-mail client rule was used to move messages tagged as spam to a special folder, which was occasionally inspected for misclassified legitimate messages. The number of retained attributes was set to 520, based on the in vitro results. No black-list was used, since a preliminary trial indicated that its contribution was insignificant.

In order to train the filter, a Support Vector Machines implementation was chosen, as it had exhibited good performance on PU3. SVMs [5, 13] are a popular instance of the class of kernel learning methods. The main idea is to map the input data into some higher dimensional feature space in which the learning problem becomes linearly separable. More formally, SVMs learn generalized linear discriminant functions of the following form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{m'} w_i \cdot h_i(\boldsymbol{x}) + w_0 \tag{1}$$

where m' is the dimensionality of the new vector space, and $h_i(\boldsymbol{x})$ are the non-linear functions that map the original attributes to the new ones. The above equation can be rewritten in its dual form, in which $h_i(\boldsymbol{x})$ appear only in dot products. These dot products can then be replaced (under some conditions) by a kernel function, which solves the computational problem of working on very high (or even infinite) dimensional feature spaces. Several specific kernel classes are known, such as polynomial, sigmoid and gaussian. The selection of a kernel and its associated parameters is application-dependent and can in some cases be important for the performance of SVMs. However, research in text classification has shown that simple linear SVMs usually perform as well as non-linear ones [5]. Since our experimental results did not provide any evidence of significant performance improvement when using polynomial kernels of various degrees, a linear SVM model was induced.

days used	212	
messages received	6732	(avg. 31.75 per day)
spam messages received	1623	(avg. 7.66 per day)
legitimate messages received	5109	(avg. 24.10 per day)
legitimate-to-spam ratio	3.15	
correctly classified legitimate messages ($L \rightarrow L$)	5057	
incorrectly classified legitimate messages ($L \rightarrow S$)	52	(avg. 1.72 per week)
correctly classified spam messages ($S \rightarrow S$)	1450	
incorrectly classified spam messages ($S \rightarrow L$)	173	(avg. 5.71 per week)
precision	96.54%	(PU3: 96.43%)
recall	89.34%	(PU3: 95.05%)
WAcc	96.66%	(PU3: 96.22%)

Table 2. Real-life evaluation results of Filtron, using the SVM with 520 1-gram attributes, for $\lambda = 1$. The SVM was trained on PU3. Bracketed precision, recall, and *WAcc* scores are the corresponding scores we had obtained with 10-fold cross validation on PU3.

Table 2 presents the overall results of the seven-month evaluation. Since no black-list was used, all the messages that were classified as spam were caught by the learnt classifier. Precision (96.54%) was very similar to the corresponding score obtained with 10-fold cross-validation on PU3 (96.43% for 520 attributes). Recall, however, was lower (89.34% as opposed to 95.05%). The analysis of the misclassified messages below sheds more light on this issue.

Overall, Filtron's performance proved to be quite satisfactory, though there is scope for improvement. The e-mail client rule that moved messages tagged as spam to the special folder left on average fewer spam messages per week (5.71) in the third author's inbox than he received in a single day (7.66). The downside was that approximately two (1.72) legitimate messages were incorrectly moved to the special folder each week. The third author developed the habit of checking that folder at the end of each week, which made the misclassifications easier to accept. He also felt that in many cases the misclassified legitimate messages were rather indifferent to him (e.g., subscription verifications, commercial newsletters, etc.), an observation confirmed by the analysis of the misclassified messages that follows. It should be noted that Filtron was never retrained during the evaluation; retraining would have allowed the system to keep both the learnt model and its white-list updated, presumably leading to better results.

We now turn to the analysis of the misclassified messages, starting from the 173 misclassified spam messages. A 30% of those were "hard spam", i.e., messages with very little or no text, containing mostly hyperlinks, messages whose text was sent as images or was hidden in attachments, messages containing tricks to confuse simple tokenizers (e.g., HTML tags within words), and messages carefully written to avoid words, phrases, and punctuation patterns that are common in spam messages. Many of these messages could have been caught by using a more elaborate preprocessor. Overall, however, they were indicative of an arms race between spammers and developers of anti-spam filters. Future anti-spam filters may well have to incorporate optical character recognition to cope with messages sent as images, and they may have to follow hyperlinks to Web pages to cope with spam messages that contain only hyperlinks and no text. Additional support for the latter point comes from the observation that a further 8% of the misclassified spam messages were advertisements of pornographic sites, carefully written as friendly letters with no pornographic vocabulary, but containing hyperlinks to the sites.

Another 23% of the misclassified spam messages were written in languages other than English, mostly German and Spanish. Non-English spam messages were rare at the time the PU corpora were assembled, but appear to have become more frequent thereafter. Hence, they were under-represented in PU3, which was the training collection for the model of the real-use evaluation. We believe that a large number of these misclassified messages would have been caught if Filtron had been frequently retrained during the seven months. This would have allowed Filtron to select non-English words as attributes, as non-English spam messages were becoming more common. Filtron's collection of spam messages now includes non-English messages.

A further 30% of the misclassified spam messages were written in BASE64 or quoted printable format, which Filtron could not handle at that time; appropriate support has now been added. Messages in these formats were very rare in the third author's legitimate messages, and, hence, the problem affected only spam messages. Filtron classified all the spam messages that were written in the two formats as legitimate, because the encoding did not allow it to detect any attribute tokens. In PU3, messages of this kind had been excluded, which partially explains the higher recall score. Another factor that contributed to the higher recall of the cross-validation is that spam messages that contained only attachments, hyperlinks, and no text were removed when PU3 was constructed, because the preprocessor converted them to empty messages. During its real-use, Filtron's preprocessor again turned spam messages of this kind to empty messages, but this time they were classified as legitimate, which lowered recall.

Another 6% of the misclassified spam messages were formal long letters advertising tremendous business opportunities or asking for financial aid. Since spam messages of this kind are very common and contain many cliche phrases, it came as a surprise that Filtron did not manage to detect so many of them. It appears that the main reason these messages were misclassified is that they contained several occurrences of the third author's name, apparently to make their messages sound more personal. This was uncommon in PU3, where the user's name was among the best legitimate-denoting attributes. Again, retraining would have allowed Filtron to diminish its confidence to this attribute as the user's name was becoming common in both classes of messages.

The remaining 3% of misclassified spam messages contained very unusual content, and, hence, they were very different from the spam messages Filtron had been trained on. Interestingly enough, some of these messages were quite relevant to the third author's scientific interests (e.g., search engines for research articles, natural language processing platforms, etc.), which may be an indication of an attempt made by spammers to target particular user groups (e.g., by collecting user names from particular scientific newsgroups, or customer lists of particular vendors). Personalized messages of this kind are more difficult to detect, as their vocabulary and content is very similar to those of the user's legitimate messages, but at least appear to be more interesting to read.

Turning to the 52 misclassified legitimate messages, more than half of them (52%) proved to be automatically generated messages (e.g., subscription verifications, virus warnings), or commercial newsletters with content very similar to spam messages. A further 22% were very short messages (3-5 words) containing attachments and hyperlinks. The above reemphasizes the need to process these elements in future anti-spam filters. Retraining would also have helped, because the senders of some of these messages were recent colleagues not in the user's white-list. The remaining 26%, were short messages (1-2 lines), with no attachments or hyperlinks, written in a very casual style that is often exploited by spammers. Non-textual attributes indicating that the incoming messages contained no attachments and hyperlinks might also have helped, since it is uncommon for spammers to send such sort messages with no attachments and hyperlinks.

6 Conclusions

We presented Filtron, a personalized anti-spam filter based on the machine learning text categorization paradigm. Its real-life evaluation confirmed that machine learning can play a prominent role in anti-spam filtering.

There were signs, however, of an arms race between spammers and filter developers. Regular re-training seems necessary, but more elaborate preprocessing is also needed to address the techniques that spammers are beginning to use to fool content-based filters. A substantial amount of Filtron's misclassifications of both kinds mentioned above can be attributed to the immaturity of the preprocessing module during its seven-month evaluation period. Having added the missing functionality, we are planning a larger-scale "in vivo" evaluation, representative of the current state implementation of the system.

In the long run, mixtures of different filtering approaches, including collaborative filtering, are more likely to be successful, and machine learning has a central role to play in such mixed filters. Combining different learning algorithms also seems to be promising, as different classifiers often make different errors [11].

References

1. I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. Technical report, National Centre for Scientific Research "Demokritos", 2004.
2. H. D. Drucker, D. Wu, and V. Vapnik. Support Vector Machines for spam categorization. *IEEE Transactions On Neural Networks*, 10(5):1048–1054, 1999.
3. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.
4. J.M. Gomez Hidalgo and M. Mana Lopez. Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of the 4th Computational Natural Language Learning Workshop*, pages 99–102, Lisbon, Portugal, 2000.
5. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Heidelberg, Germany, 1998.
6. G.H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, Montreal, Quebec, 1995.
7. D.D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, Germany, 1998.
8. T.M. Mitchell. *Machine learning.* McGraw-Hill, 1997.
9. J. Platt. Fast training of Support Vector Machines using Sequential Minimal Optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, 1999.
10. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization – Papers from the AAAI Workshop*, pages 55–62, Madison, Wisconsin, 1998.
11. 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*, 6(1):49–73, 2003.
12. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
13. V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New York, 1995.
14. Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, Tennessee, 1997.