Spam Filtering Based on Latent Semantic Indexing

Wilfried N. Gansterer[*] Andreas G. K. Janecek[†] Robert Neumayer[‡]

Abstract

In this paper, a study on the classification performance of a vector space model (VSM) and of latent semantic indexing (LSI) applied to the task of spam filtering is summarized. Based on a feature set used in the extremely widespread, de-facto standard spam filtering system SpamAssassin, a vector space model and latent semantic indexing are applied for classifying e-mail messages as spam or not spam. The test data sets used are partly from the official TREC 2005 data set and partly self collected.

The investigation of LSI for spam filtering summarized here evaluates the relationship between two central aspects: (*i*) the truncation of the SVD in LSI and (*ii*) the resulting classification performance in this specific application context. It is shown that a surprisingly large amount of truncation is often possible without heavy loss in classification performance. This forms the basis for good and extremely fast approximate (pre-) classification strategies, which are very useful in practice.

The approachies investigated in this paper are shown to compare favorably to two important alternatives: (*i*) They achieve better classification results than SpamAssassin, and (*ii*) they are better and more robust than a related LSI-based approach using textual features which has been proposed earlier.

Keywords: information retrieval, vector space model, latent semantic indexing, feature selection, e-mail classification, spam filtering

1 Introduction

Unsolicited bulk or commercial e-mail (UBE, UCE, 'spam') has become a severe problem on the Internet over the last years. Although many strategies for addressing this problem have been proposed, we are still far away from a satisfactory and fundamental solution. In part, this is due to the fact that many of the methods proposed and developed into concepts usable in practice have an 'ad-hoc' nature and lack a lasting effect. Those individuals who make profit based on the spam phenomenon (the 'spammers') can easily find ways to circumvent them.

The investigations described here were motivated by the general assumption that spam e-mail messages tend to have several semantic elements in common, which are usually not present in regular e-mail (commonly called 'ham'). This assumption is plausible due to the economic aspects underlying the spam phenomenon (see, for example, [12] for a discussion). However, these semantic elements are hard to pinpoint comprehensively, because they are not fully known, not always explicit, dynamically changing, etc.. In other words, they are in some sense implicit or *latent*. Many of the approaches developed so far try to identify some of these distinctive elements and then to design methods based thereon. But, due to the difficulties in identifying a comprehensive set of elements which unambiguously characterize spam, these approaches did not yet result in fundamental and persistent anti-spam strategies.

1.1 Approach Investigated. The problem of filtering spam is a binary classification problem in the sense that every incoming e-mail has to be classified as 'spam' or 'not spam'. The main objective in this paper is to investigate and evaluate the applicability of two very powerful classification techniques, *vector space models* (*VSM*) and *latent semantic indexing* (*LSI*, see [4] and the references there), in this context. Since LSI is capable of

[*]University of Vienna, Research Lab Computational Technologies & Applications and Institute of Distributed and Multimedia Systems.

[†]University of Vienna, Institute of Distributed and Multimedia Systems.

[‡]Vienna University of Technology, Institute of Software Technology and Interactive Systems

extracting underlying latent semantic information from e-mail data, it seemed to be an interesting candidate for the task at hand.

1.2 Background and Related Work. Existing anti-spam methods can be categorized according to their point of action in the e-mail transfer process. Consequently, three classes of approaches can be distinguished (for details, see [9]). *Pre-send* methods act before the e-mail is transported over the network, whereas *post-send* methods act after the e-mail has been transferred to the receiver. A third class of approaches comprises new protocols, which are based on modifying the transfer process itself. Pre-send methods (see, for example, [3] or [7]) are very important because of their potential to avoid most of the waste of resources (network traffic, etc.) caused by spam. However, the efficiency of pre-send methods and of new protocols heavily depends on their widespread deployment. It is unrealistic to expect global acceptance and widespread use of these new methods in the near future, and thus the third group of methods, post-send spam filtering methods will continue to be the 'work horse' in spam defense.

Many of the spam filtering concepts currently used in practice are mostly static in nature, such as black- and whitelisting [9], or rule-based filters. A de-facto standard of a rule-based spam filtering system is SpamAssassin [2]. SpamAssassin extracts a multitude of features from incoming e-mail messages, comprising pieces of information from header, body, and the full text of the message. One of the tests integrated is a Bayesian classifier, a statistical approach exclusively based on textual features. An example for its application to the scenario of spam filtering is given in [1]. The underlying idea is to compute a conditional probability for an e-mail being spam based on the words (*tokens*) it contains. Another approach closely related to the methods investigated in this paper has been proposed by Gee [11], where an LSI-based classifier is used for spam filtering. However, in contrast to the method investigated here, Gee's approach is exclusively based on textual features of the e-mail messages.

One of the main motivations for our work was to investigate the influence of the choice of different feature sets on the performance of VSM- and LSI-based spam filtering. In particular, our objective was to compare the classification performance of Gee's approach (LSI on text-based features) to the performance achieved with LSI on a set of 'meta features', such as used in SpamAssassin.

From the point of view of the individual user, these and similar filtering methods may achieve reasonably satisfactory results provided they are trained, tuned and maintained permanently. The user effort required for sustaining satisfactory performance is one of the disadvantages of almost all existing filtering methods.

1.3 Synopsis. In Section 2, the basic principle of LSI is reviewed briefly, and in Section 3 its adaptation to the context of spam filtering is discussed. Section 4 describes our implementation of this concept and summarizes a set of experiments performed with two e-mail test sets, and in Section 5 conclusions are drawn and future work in this context is outlined.

2 Classification Methodology

In this section the standard VSM and LSI methods—as, for example, used in web search engines [14]—are reviewed briefly. Their adaptation and application to the task of spam filtering is discussed in Section 3.

2.1 The Basic Vector Space Model. In information retrieval, a vector space model (VSM) is a widely used model for representing information. For example, documents and queries are represented as points in a potentially very high dimensional, metric vector space. The distance between a query vector and the document vectors is the basis for the information retrieval process, which we try to summarize here very briefly in its standard form (see, for example, [13] or [14]).

Generally speaking, a vector space model for n documents is based on a certain set F of f features, $F = \{F_1, F_2, \ldots, F_f\}$. A feature extraction component has to be available to extract the values v_{ij} for each of these features F_i from each document j. The f-dimensional (column) vector V_j

with the components $v_{ij}, i = 1, 2, \ldots, f$, then represents document j in the VSM. The $f \times n$ feature-document matrix M (in other contexts called the term-document matrix) is composed using the vectors V_j for all documents as columns.

Given a query vector q of length f, the distances to all documents represented in M can then be measured (for example) in terms of the cosines of the angles between q and the columns of M. The closest column represents the closest match between the document collection and the query vector.

2.2 LSI. Latent semantic indexing (LSI) or latent semantic analysis (LSA) is a variant of the basic vector space model. Instead of the original feature-document matrix M it operates on an approximation to M. More specifically, based on the singular value decomposition of M, a low rank approximation M_k is constructed:

$$M = U\Sigma V^\top \approx U_k \Sigma_k V_k^\top =: M_k,$$

where Σ_k contains the k largest singular values of M and $k < f$. Technically, this approximation replaces the original matrix M by another matrix whose column space is a subspace of the column space of M.

LSI Truncation. In this paper, we call this approximation process of M *LSI truncation*. The amount of truncation is denoted here as a percentage in the following sense: If Σ_k contains only those k singular values of M greater than p percent of the maximum singular value of M, then the approximation is called 'LSI p%' truncation. If, for example, the maximum singular value of M is 100, then for LSI 2.5% all k singular values greater than 2.5 are used in the approximation of M. As the truncation parameter p increases, the rank of the approximation matrix decreases.

One of the main drawbacks of LSI is the rather large (sometimes even prohibitive) computational cost required for computing the singular value decomposition. On the other hand, it has the positive side effect of reducing storage requirements and computational cost for determining the distances between the query vector and the documents. These and other considerations are discussed in greater detail in [4].

3 VSM and LSI for Spam Filtering

The main focus of this paper is the application of VSM and LSI to the task of spam filtering. In this case, each document is an e-mail message and each e-mail message is represented by a point in the vector space, i.e., by a vector in the vector space model.

In order to construct this vector space model, each e-mail message has to be represented by a set of feature values determined by a feature extraction component. This leads to two central questions arising in the context of spam filtering methodology: *(i)* Which features of e-mail data are (most) relevant for the classification into spam and ham (textual features, header information, etc.)? *(ii)* Based on a certain set of features, what is the best method for categorizing into spam and ham?

In this paper, we consider and compare two types of features which are very important and widely used in the context of spam filtering. On the one hand, we use the features extracted by the state-of-the-art spam filtering system SpamAssassin. These features are denoted by 'F_SA'. On the other hand, we use a comparable number of purely text-based features. These features are denoted by 'F_TB'. In the following, these two types of feature sets and their extraction are discussed.

3.1 SpamAssassin Features. For our first set of features we use the SpamAssassin system [2] to extract their values from each message. A big set of tests and rules is used to extract these features. Different areas of each e-mail message are tested, comprising tests of the message body (56% of all tests), of the message header (36%), URI checks (5%) as well as rawbody tests (MIME checks, 2%) and blacklist tests (1%). Tests of the message body comprise Bayesian classifiers, language specific tests, tests searching for obfuscated words, html tests and many others (for a complete list see `http://spamassassin.apache.org/tests_3_1_x.html`).

Each test is assigned a certain value. This value is a positive real number, if the test points to a spam message, and a negative real number, if the test points to a ham message. The specific values have been derived using a neural network trained with error back propagation. The overall rating of a message is computed by summing up all values of the tests. If this sum exceeds a user-defined threshold (the standard threshold is set to 5) a message is classified as spam. Increasing the threshold will decrease the spam detection rate but will also reduce the false positive rate, decreasing it will increase the spam detection rate, but the false positive rate as well.

Although the number of features extracted from each e-mail message by the SpamAssassin system is very large (795 in the version used), experimental analysis shows that only a relatively small subset of these features provides widely applicable useful information. More specifically, for our data sets (see Section 4.1) only about *half* of all tests for extracting a feature triggered at least once, and only about 4% of the tests triggered for more than 10% of the messages. Beside the integrated Bayesian classifier (which returns a value for every message) the most often triggering SpamAssassin tests for our data sets used are the *"all_trusted"* test that checks if a message has only passed through trusted SMTP relays (checks for a ham), as well as the *"uri_offers"* test that checks if a message contains a link to company offers, and the *"razor2"* real-time blacklist test (both check for a spam). A more detailed list of the best triggering tests for the data sets used in this work can be found in [8].

Moreover, the feature set used by SpamAssassin is *'unbalanced'* in the sense that about 96% of all tests check for a feature which is characteristic for spam and only about 4% of the tests check for a feature which is characteristic for ham. Another potential problem is that some of the SpamAssassin tests tend to trigger incorrectly. For our test data, eleven tests triggered wrongly for more than 2% of the messages, and seven of them triggered wrongly more often than correctly (see [8]). Obviously, this has a negative impact on the classification performance. However, most of these problematic tests are assigned a low default score and therefore do not have a strong impact on the overall classification result.

In the work described here, we selected the 377 and 299 SpamAssassin features, respectively, which triggered at least once for both of the data sets used (see Section 4.1) as a first starting point. For the SpamAssassin feature set we used features in binary form as well as the original values used by SpamAssassin. In some cases, using binary features only turned out to yield slightly better classification results than using the original, weighted values. Due to the shortcomings of the SpamAssassin feature set mentioned above, we are currently working on developing improved feature selection and extraction strategies.

3.2 Text-based Features. The classical alternative, which is used widely, not only in text mining, but also in the area of spam filtering, is a purely text-based feature set. Consequently, we consider a second feature set in this paper, which consists of words extracted from the e-mail messages. Document frequency thresholding is used for reducing the otherwise potentially prohibitive dimensionality.

Feature Selection and Dimensionality Reduction. When tokenizing text documents one often faces very high dimensional data. Tens of thousands of dimensions are not easy to handle, therefore feature selection plays a significant role. Document frequency thresholding achieves reductions in dimensionality by excluding terms having very high or very low document frequencies. Terms that occur in almost all documents in a collection do not provide any discriminating information. A good overview of different term selection methods explaining their advantages and disadvantages is given in [15].

It is therefore possible to scale down to the requested dimensionality, usually about 300 features. Document frequency thresholding is often used as a first step to reduce the dimensionality. Further, techniques like information gain are used to select the best n features from that 'pre-selection' due to performance issues.

Document Frequency Thresholding. Document frequency thresholding is a feasible feature selection for unsupervised data, too. The basic assumption here is that very frequent terms are less discriminative to distinguish between classes (a term occurring in every single spam and ham message would not contribute to differentiate between them). The largest number of tokens (words), however, occurs only in a very small number of documents. The biggest advantages of document frequency thresholding is that there is no need for class information and it is therefore mainly used for clustering applications, where the data only consists of one class or no class information is available at all. Besides, document frequency thresholding is far less expensive in terms of computational power. In this context that technique is used for dimensionality reduction for clustering and to compare the classification results obtained by the more sophisticated approaches. Document frequency thresholding proceeds as follows:

- At first the upper threshold is fixed to 0.5, hence all terms that occur in more than half of the documents are omitted.
- The lower boundary is dynamically changed as to achieve the desired number of features.

Information Gain. Information gain (IG) is a technique originally used to compute splitting criteria for decision trees. Different feature selection models including information gain are described in [15]. The basic idea behind IG is to find out how well each single feature separates the given data set.

The overall entropy I for a given dataset S is computed in Equation 3.1.

$$I = -\sum_{i=1}^{C} p_i \log_2 p_i \tag{3.1}$$

where C denotes the available classes and p_i the proportion of instances that belongs to one of the i classes. Now the reduction in entropy or gain in information is computed for each attribute or token.

$$IG(S,A) = I(S) - \sum_{v \epsilon A} \frac{|S_v|}{|S|} I(S_v) \tag{3.2}$$

where v is a value of A and S_v the number of instances where A has that value (i.e. the number of instances A occurs in).

This results in an information gain value for each token extracted from a given document collection. Documents are represented by a given number of tokens having the highest information gain values for the content-based experiments.

3.3 Training and Classification. The application of VSM and LSI to spam filtering investigated here involves two phases: a 'training phase' and the actual 'classification phase'.

The training phase comprises the indexing of two known data sets (one consisting of spams and one consisting of hams) and, in the case of LSI, the computation of the singular value decomposition. The classification phase comprises the query indexing and the retrieval of the closest message from the training sets. A newly arriving message can be classified by indexing it based on the feature set used and comparing the resulting query vector to the vectors in the training matrices. If the closest message is contained in the spam training set, then the query message is classified as spam, otherwise it is classified as ham. The distance measurement is based on the angles between the query vector and the vectors in the training matrices and is computed analogously to standard VSM (see [13]).

3.4 Performance Metrics. In spam filtering, the goal is to maximize the rate of 'true positives' and to minimize rate of 'false positives' simultaneously (in order not to lose important messages). In this context, a 'positive' denotes an e-mail message classified as spam. A 'true positive' consequently is a spam message which was (correctly) classified as spam, and a 'false positive' is thus a ham message which was (wrongly) classified as spam. These two metrics are sometimes aggregated into a single one, for example, by computing the percentage of correctly classified messages (correctly classified spams plus correctly classified hams) in all messages classified (see, for example, Figure 1).

4 Experimental Evaluation

4.1 Data Sets. For the experiments we used two different data sets, one consisting of a part (25%) of the TREC 2005 spam and ham corpus. This data set is denoted as 'S1' in the following and is publicly available [5]. The other data set consists of self collected e-mail messages and is denoted as 'S2' in the following. The messages in S2 have been collected recently over a period of several months from several sources to cover as much diversity as possible. Spam was mostly collected from various spam traps, and ham from volunteers and private e-mail. S1 contains a ham sample consisting of 9751 ham messages and a spam sample consisting of 13179 spam messages. S2 contains a ham sample consisting of 5502 ham messages and a spam sample consisting of 5502 spam messages.

4.2 Experimental Setup. We performed two different cross validations for our data sets, a three-fold cross validation for the larger sample S1 and a tenfold cross validation for the smaller sample S2. For an n-fold cross validation we split each of our samples randomly into n parts of roughly equal size and used alternately *n-1* of these parts for training and one of them for testing. The true/false positive rates and true/false negative rates as well as the aggregated classification results were measured. The cross validations were performed for six different LSI truncations using features F_SA and F_TB (as defined in Section 3).

The average rank k of the truncated SVD matrices for both data sets and both types of features are listed in Tables 1 and 2.

Table 1: Rank of the truncated SVD matrices for different cut-off values in the singular values for S1

cut:	0.0%	2.5%	5.0%	10.0%	25.0%	50.0%
Features F_SA						
kHam:	377	24	12	5	1	1
kSpam:	377	94	47	23	6	1
Features F_TB						
kHam:	377	83	37	14	4	3
kSpam:	377	86	50	19	7	2

4.3 Hardware. Due to the large amount of data and the variety of different parameter-settings we used six different machines for our test runs ranging from a sun sparc 64-bit multiprocessor running SunOS v5.10 to standard desktop PCs running Windows XP and Linux Ubuntu 4.0.3.

Table 2: Rank of the truncated SVD matrices for different cut-off values in the singular values for S2

cut:	0.0%	2.5%	5.0%	10.0%	25.0%	50.0%
Features F_SA						
kHam:	299	31	15	7	3	1
kSpam:	299	79	42	17	5	2
Features F_TB						
kHam:	299	53	26	14	4	2
kSpam:	299	55	27	11	4	2

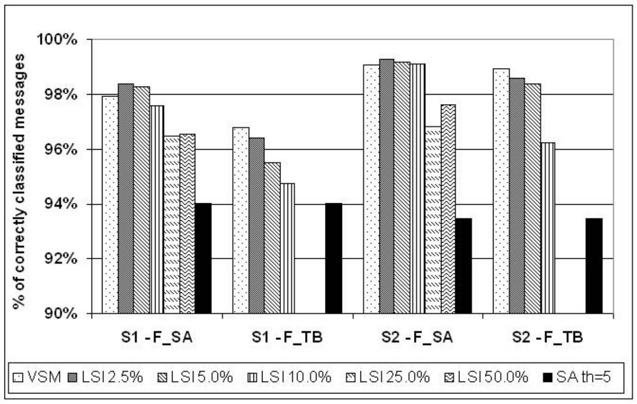

Figure 1: Aggregated classification results for S1 and S2

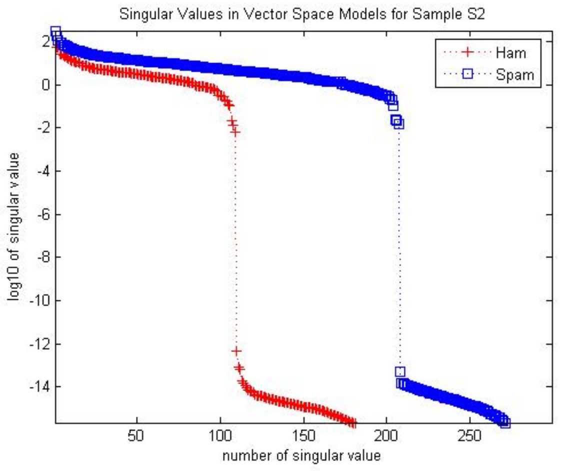

Figure 2: Singular values in vector space models for sample S2 using features F_SA

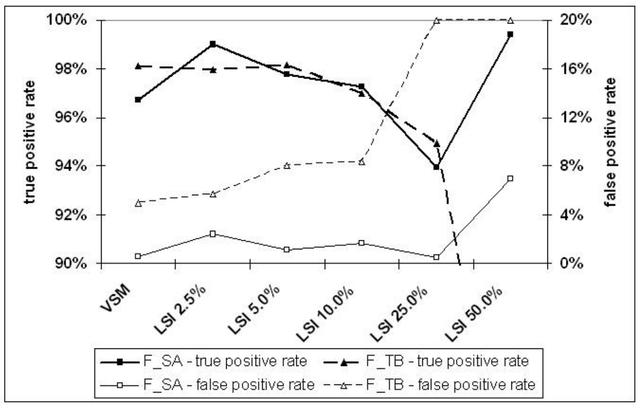

Figure 3: True and false positive rates for sample S1

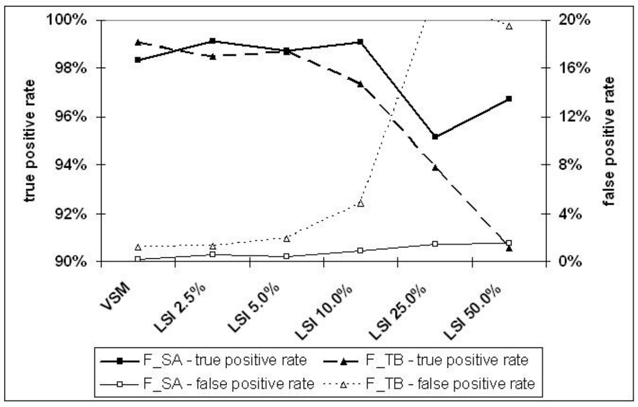

Figure 4: True and false positive rates for sample S2

4.4 Aggregated Classification Results. Figure 1 depicts the aggregated classification results for both data sets used showing six different LSI truncations for the two feature sets F_SA and F_TB. SpamAssassin assigns positive values to 'spam' features and negative values to 'ham' features and performs classification based on a threshold value, the standard setting and rather conservative decision border being a sum of 5. To compare our results to a standard approach the classification results of SpamAssassin using the standard threshold are plotted as well ('SA th=5'). The bars for LSI 25.0% and LSI 50.0% for features F_TB are not visible as they could not achieve a aggregated classification result over 90%. It is clearly visible that for low LSI truncations all results show an improvement in the aggregated classification result over SpamAssassin.

The structure of the singular values in the vector space models for sample S2 is illustrated in Figure 2. It is interesting to note the sharp decline in the magnitude of the singular values for both groups (ham and spam). Based on these observations, it is easier to understand that when using features F_SA the truncation of the SVD matrix to $k = 3$ rows (sample S1) or even to $k = 1$ row (sample S2) still achieves a classification quality better than SpamAssassin (see Figure 1 and Tables 1 and 2 - features F_SA for LSI 50%).

Comparing the feature sets F_SA and F_TB it can be seen that for all LSI truncations the results for F_SA are better than the corresponding results for F_TB. For the LSI classification the aggregated results for the sample S2 are better than for S1. The results for the SA standard threshold of 5 are opposite—here the result for S1 exceeds the result for S2. Using features F_SA the aggregated results for LSI 2.5% and LSI 5.0% are slightly better than the results for LSI 0.0% (standard VSM) where all features are used for the classification. In contrast, for F_TB every increase of the LSI truncation (decrease in the rank of the truncated SVD) causes a decrease in the classification result.

4.5 True/False Positives. In contrast to Figure 1 where the results are aggregated, Figures 3 and 4 show the true and false positive rates separately. It should be pointed out, that for sample S1 SpamAssassin (using the standard threshold of 5) achieves true/false positive rates of only 89.47% and 0.34%, respectively, and for the sample S2 rates of only 87.12% and 0.18%, respectively (partly out of range in Figures 3 and 4). Using a more aggressive threshold for SpamAssassin increases the false positive rate as well as the true positive rate. For example, for S1 a classification using a threshold of 3 achieves a true positive rate of 93.89% and a false positive rate of 1.94%.

Both figures show a similar behavior for the true/false positive rates for F_SA and F_TB and indicate a high false positive rate using features F_TB. In particular, for S1 the false positive rate is very high (4.91% using VSM), although the

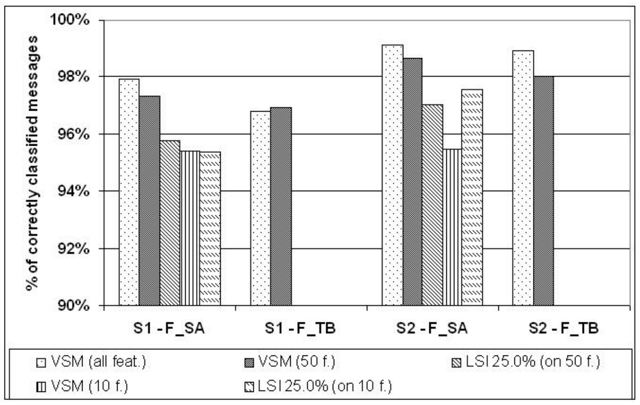

Figure 5: VSM and LSI on reduced feature sets

aggregated classification result reaches respectable 96.80% (see Figure 1, F_FB using VSM). The false positive rate for S2 using features F_TB is lower but still clearly higher than using features F_SA. The false positive rates do not differ much for low LSI truncations for both features.

4.6 Feature Reduction. Finally, we also evaluated the aggregated classification results using only a reduced number of features. Reducing the number of features saves computation in the classification process for each message and therefore computing time. Moreover, using LSI on reduced features further decreases the computational effort. This can be used as a fast pre-processing classifier where a huge amount of messages can be assigned to a given category very fast. Figure 5 shows the classification results using only the 50 (10) most triggering features from F_SA and the results using only the 50 (10) top discriminating features from F_TB. All results for a reduced amount of features are slightly lower than for the original amount of features (except for the top 50 F_TB features for sample S1 where the results are nearly equal). As LSI is applied to the reduced feature set only the results for features F_SA remain acceptable and thus features F_SA prove to be more robust in terms of feature reduction than features F_TB.

5 Conclusions

The application of latent semantic indexing (LSI) to the task of spam filtering has been investigated. The underlying vector space models are based on two different feature sets: purely text-based features were used for comparison with related work on LSI for spam filtering, and, as a competing alternative, a feature set used in the widespread SpamAssassin system for spam filtering.

Experimental evaluations on two large data sets showed several interesting results:

1. The results achieved with both, VSM and LSI, based on the feature set of SpamAssassin (F_SA) are better than the results achieved on pure textual features (F_FB).

2. For both feature sets (F_SA and F_TB), VSM and LSI achieve significantly better classification results than the extremely widespread defacto standard for spam filtering, SpamAssassin.

3. For both feature sets, VSM achieves better classification results than LSI at many of the truncation levels. When using LSI based on the SpamAssassin feature set (F_SA), the classification results are surprisingly robust to very low rank approximations. This indicates methods for computing good preclassifications very fast, which can be extremely useful in practice.

4. Looking at true/false positive rates it can be seen that for all truncation levels of LSI the false positive rate based on features F_SA is much lower than the one based on using features F_TB, which is a very important aspect in spam filtering. SpamAssassin is also able to achieve a quite good false positive rate, however combined with a very poor true positive rate compared to the LSI results.

5. The classification performance of LSI based on SpamAssassin features is also quite robust when the number of features used is reduced significantly and still achieves remarkably good aggregated classification results, which is not the case for LSI based on purely textual features.

6. Consequently, both VSM and LSI based on a reduced feature set are suitable as a coarse

but highly efficient pre-classification strategy, for example, in the context of a recently developed component based architecture for spam filtering [10].

Overall, the experiments indicate that VSM and LSI perform very well for spam filtering if the feature set is properly chosen. In particular, we showed in this paper that the SpamAssassin feature set achieves better results than purely textual feature sets.

Current work focuses on further analysis of various feature selection and feature extraction strategies in order to further improve upon the currently used SpamAssassin tests and on comparisons with other classification methods (see, for example, [6]). Moreover, the utilization of the sparsity structure of M for classification is an imporant topic in current investigations. Applying sparse matrix techniques will make VSM highly competitive, in particular in terms of computational cost.

Acknowledgements. We would like to express our gratitude to Internet Privatstiftung Austria, mobilkom austria and UPC Telekabel for supporting this research. We also thank the anonymous referees for their valuable comments which helped us improve this paper.

References

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. D. Spyropoulos, *An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages.*, in SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval, 2000, pp. 160–167.

[2] Apache Software Foundation, *SpamAssassin open-source spam filter*, 2006. `http://spamassassin.apache.org/`.

[3] A. Back, *Hashcash - a denial of service counter-measure*, 2002. `http://www.hashcash.org/papers/hashcash.pdf`.

[4] M. W. Berry, Z. Drmac, and E. R. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335–362.

[5] G. V. Cormack and T. R. Lynam, *Trec 2005 spam public corpora*, 2005. `http://plg.uwaterloo.ca/cgi-bin/cgiwrap/gvcormac/foo`.

[6] N. Cristiani and B. Scholkopf, *Support vector machines and kernel methods: The new generation of learning machines*, AI Magazine, 23 (2002), pp. 31–41.

[7] W. Gansterer, H. Hlavacs, M. Ilger, P. Lechner, and J. Strauss, *Token buckets for outgoing spam prevention*, in Proceedings of the IASTED International Conference on Communication, Network and Information Security, 2005.

[8] W. Gansterer, M. Ilger, A. Janecek, P. Lechner, and J. Strauss, *Final report project 'Spamabwehr II'*, Technical Report FA384018-5, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, 05/2006.

[9] W. Gansterer, M. Ilger, P. Lechner, R. Neumayer, and J. Strauss, *Anti-spam methods—state of the art*, Technical Report FA384018-1, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, 2005.

[10] W. N. Gansterer, A. Janecek, and P.Lechner, *A reliable component-based architecture for e-mail filtering*, in Proceedings of ARES07 - 2nd International Conference on Availability, Reliability and Security, 2007. (*to appear*).

[11] K. Gee, *Using latent semantic indexing to filter spam*, in ACM Symposium on Applied Computing, Data Mining Track, 2003, pp. 460–464.

[12] M. Ilger, J. Strauss, W. Gansterer, and C. Proschinger, *The economy of spam*, Technical Report FA384018-6, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, 2006.

[13] A. N. Langville, *The linear algebra behind search engines*, in Journal of Online Mathematics and its Applications (JOMA), 2005, Online Module, 2005.

[14] A. N. Langville and C. Meyer, *The use of linear algebra by web search engines*, in IMAGE Newsletter, 33:2-6, 2004.

[15] Y. Yang and J. O. Pedersen, *A comparative study on feature selection in text categorization*, in Proceedings of ICML-97, 14th International Conference on Machine Learning, D. H. Fisher, ed., Nashville, US, 1997, Morgan Kaufmann Publishers, San Francisco, US, pp. 412–420.