SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:327–335
Published online 25 September 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.639

RESEARCH ARTICLE

Compression-based spam filter

Tiago A. Almeida[1]* and Akebo Yamakami[2]

[1] Department of Computer Science, Federal University of São Carlos – UFSCar, 18052-780, Sorocaba, SP, Brazil
[2] School of Electrical and Computer Engineering, University of Campinas – UNICAMP, 13083-970, Campinas, SP, Brazil

ABSTRACT

Nowadays, e-mail spam is not a novelty, but it is still an important problem with a high impact on the economy. Spam filtering poses a special problem in text categorization, in which the defining characteristic is that filters face an active adversary, which constantly attempts to evade filtering. In this paper, we present a novel approach to spam filtering based on a compression-based model. We have conducted an empirical experiment on eight public and real non-encoded datasets. The results indicate that the proposed filter is fast to construct, is incrementally updateable, and clearly outperforms established spam classifiers. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

compression-based model; spam filter; text categorization; knowledge-based system; machine learning

***Correspondence**

Tiago A. Almeida, Department of Computer Science, Federal University of São Carlos – UFSCar, 18052-780, Sorocaba, SP – Brazil.
E-mail: talmeida@ufscar.br

1. INTRODUCTION

The term *spam* is used to denote an unsolicited commercial e-mail. According to updated reports, the amount of spam is still increasing. The average of spam sent per day increased from 2.4 billion in 2002* to 300 billion in 2010.† The cost in terms of lost productivity in the USA has reached US$ 21.58 billion annually, whereas that in the worldwide productivity is estimated to be US$ 50 billion.‡ On a worldwide basis, the information technology cost of dealing with junk e-mail was estimated to rise from US$ 20.5 billion in 2003 to US$ 198 billion in 2010.

In recent years, many methods have been proposed to automatic spam filtering, such as: white and black-lists, challenge and response systems, rule-based approaches, methods which take into account the sender's domain, clustering, and others. However, machine learning algorithms have been achieved more success [1]. These methods include approaches that are considered top performers in text categorization, such as Rocchio [2,3], Boosting [4,5], Naïve Bayes (NB) classifiers [6–11], and Support Vector Machines (SVMs) [12–16]. The two latter currently appear to be the best spam filters available in the literature [1,11]. For more details about spam filters, refer to the surveys [17,18,1,19,20].

A relatively recent approach for inductive inference, which is rarely employed in text categorization, is the Minimum Description Length principle. It states that the best explanation, given a limited set of observed data, is the one that yields the greatest compression of the data [21–23].

In this paper, we present a spam classifier based on the Minimum Description Length principle and compare its performance with seven different models of the well-known NB classifiers and the linear SVM. We have conducted an empirical experiment using eight well-known, large, and public databases, and the reported results states that our approach outperforms currently established spam filters.

Fragments of this work were previously presented at ACM SAC 2010 [24,10]. Here, we have connected all ideas in a very consistent way and offered a lot more details about each study and significantly extended the performance evaluation. To be clear, first, we present more details about the proposed method, its main features and how it works. Second, we compare the new approach with several established classifiers instead of only two, as presented in the mentioned papers. Third and the most important, we evaluate the new classifier using public and larger e-mail corpora, such as TREC06 and CEAS08.

The remainder of this paper is organized as follows: Section 2 presents the general concepts regarding spam classifiers. In Section 3, we offer details about the proposed compression-based approach. The experiments

*See http://www.spamlaws.com/spam-stats.html.
†See www.ciscosystems.cd/en/US/prod/collateral/vpndevc/cisco_2009_asr.pdf.
‡See http://www.rockresearch.com/news_020305.php.

and results are showed in Section 4. Finally, Section 5 offers conclusions and outlines for future work.

2. GENERAL CONCEPTS

In general, the machine learning algorithms applied to spam filtering can be summarized as follows.

Given a set of messages $\mathcal{M} = \{m_1, m_2, \ldots, m_j, \ldots, m_{|\mathcal{M}|}\}$ and a category set $\mathcal{C} = \{\text{spam } (c_s), \text{ legitimate } (c_l)\}$, where m_j is the jth e-mail in $\mathcal{M}$ and $\mathcal{C}$ is the possible label set, the task of automated spam filtering consists in building a Boolean categorization function $\Psi(m_j, c_i) : \mathcal{M} \times \mathcal{C} \rightarrow \{true, false\}$. When $\Psi(m_j, c_i)$ is true, it indicates message m_j belongs to category c_i; otherwise, m_j does not belong to c_i.

In the setting of spam filtering, there are only two category labels: spam and legitimate (also called ham). Each message $m_j \in \mathcal{M}$ can only be assigned to one of them, but not to both. Therefore, we can use a simplified categorization function $\Psi_{\text{spam}}(m_j) : \mathcal{M} \rightarrow \{\text{true}, \text{false}\}$. Hence, a message is classified as spam when $\Psi_{\text{spam}}(m_j)$ is true and legitimate otherwise.

The application of supervised machine learning algorithms for spam filtering consists of two stages:

(1) *Training*. A set of labeled messages ($\mathcal{M}$) must be provided as training data, which are first transformed into a representation that can be understood by the learning algorithms. The most commonly used representation for spam filtering is the vector space model, in which each document $m_j \in \mathcal{M}$ is transformed into a real vector $\rightarrow x_j \in \Re^{|\Phi|}$, where Φ is the vocabulary (feature set), and the coordinates of $\rightarrow x_j$ represent the weight of each feature in Φ. Then, we can run a learning algorithm over the training data to create a classifier $\Psi_{\text{spam}}(\rightarrow x_j) \rightarrow \{\text{true}, \text{false}\}$.
(2) *Classification*. The classifier $\Psi_{\text{spam}}(\rightarrow x_j)$ is applied to the vector representation of a message $\rightarrow x$ to produce a prediction whether $\rightarrow x$ is spam or not.

2.1. Message representation

Each message m is composed by a set of tokens $m = \{t_1, \ldots, t_{|m|}\}$, where each token t_k may be a term (or word; e.g., "viagra"), a set of terms (e.g., "to be removed"), or a single character (e.g., "$"). Thus, we can represent each e-mail by a vector $\rightarrow x = \langle x_1, \ldots, x_{|m|} \rangle$, where $x_1, \ldots, x_{|m|}$ are values of the attributes $X_1, \ldots, X_{|m|}$ associated with the tokens $t_1, \ldots, t_{|m|}$. In the simplest case, each token represents a single term, and all attributes are Boolean: $X_i = 1$ if the message contains t_i, or $X_i = 0$ otherwise.

Alternatively, attributes may be an integer computed by token frequencies (TFs) representing how many times each token appears in the message. A third alternative is to associate each attribute X_i to a normalized TF, $x_i = \frac{n(t_i)}{|m|}$, where $n(t_i)$ is the number of occurrences of the token represented by X_i in m and $|m|$ is the length of m measured in token occurrences. Normalized TF takes into account the token repetition versus the size of message [25].

3. COMPRESSION-BASED SPAM FILTER

Compression-based techniques seem to be a promising alternative to categorization tasks. Teahan and Harper [26] and Frank *et al*. [27] performed extensive experiments to evaluate the performance of different approaches for text categorization on the standard Reuters-21578 collection and compared compression-based algorithms, such as prediction by partial matching with NB classifiers and SVM. According to the found results, compression-based models perform better than word-based NB techniques and approach the performance of linear SVM.

A relatively recent method for inductive inference, which is still rarely used in text categorization, is the Minimum Description Length (MDL) principle. It states that the best explanation, given a limited set of observed data, is the one that yields the greatest compression of the data [21–23].

The purpose of statistical modeling is to discover regularities in observed data. The success in finding such regularities can be measured by the length with which the data can be described. This is the rationale behind the MDL principle [21]. The fundamental idea is that any regularity in a given set of data can be used to compress the data.

According to the traditional MDL principle, the favorite model results in the shortest description of the model and the data, given this model. In other words, the model that best compresses the data is selected. This model selection criterion naturally balances the complexity of the model and the degree to which this model fits the data. This principle was first introduced by Rissanen [21], and it has become an important concept in information theory.

Let $\mathcal{Z}$ be a finite or countable set and let P be a probability distribution on $\mathcal{Z}$. Then there exists a prefix code C for $\mathcal{Z}$ such that for all $z \in \mathcal{Z}$, $L_C(z) = \lceil -\log_2 P(z) \rceil$. C is called the code corresponding to P. Similarly, let C be a prefix code for $\mathcal{Z}$. Then there exists a (possibly defective) probability distribution P such that for all $z \in \mathcal{Z}$, $-\log_2 P'(z) = L_{C'}(z)$. P' is called the probability distribution corresponding to C'. Thus, large probability according to P means small code length according to the code corresponding to P and vice versa [21–23].

The goal of statistical inference may be cast as trying to find regularity in the data. Regularity may be identified with ability to compress. MDL combines these two insights by viewing learning as data compression: it tells us that, for a given set of hypotheses $\mathcal{H}$ and data set $\mathcal{D}$, we should try to find the hypothesis or combination of hypotheses in $\mathcal{H}$ that compresses $\mathcal{D}$ most [21–23].

This idea can be applied to all sorts of inductive inference problems, but it turns out to be most fruitful in problems of model selection and, more generally, dealing with overfitting [23]. An important property of MDL methods is that they provide automatically and inherently protection against overfitting and can be used to estimate both the parameters and the structure of a model. In contrast, to avoid overfitting when estimating the structure of a model, traditional methods such as maximum likelihood must be modified and extended with additional, typically ad hoc principles [23].

Consider the following example. Suppose we flip a coin 1000 times and we observe the numbers of heads and tails. We consider two model classes: the first consists of a code that represents each outcome with a 0 for heads or a 1 for tails. This code represents the hypothesis that the coin is fair. The code length according to this code is always exactly 1000 bits. The second model class consists of all codes that are efficient for a coin with some specific bias, representing the hypothesis that the coin is not fair. Say that we observe 510 heads and 490 tails. Then the code length according to the best code in the second model class is shorter than 1000 bits. For this reason a naive statistical method might put forward this second hypothesis as a better explanation for the data. However, in an MDL approach, we would have to construct a single code based on the hypothesis; we cannot just use the best one. A simple way to do it would be to use a two-part code, in which we first specify which element of the model class has the best performance, and then, we specify the data using that code. We will need quite a lot of bits to specify which code to use; thus, the total codelength based on the second model class would be larger than 1000 bits. Thus, if we follow an MDL approach, the conclusion has to be that there is not enough evidence in support of the hypothesis that the coin is biased, even though the best element of the second model class provides better fit to the data. Consult Grunwald [23] for further details.

In essence, compression algorithms can be applied to text categorization by building one compression model from the training documents of each class and using these models to evaluate the target document.

3.1. The proposed spam filter

Given a set of classified training messages $\mathcal{M}$, the task is to assign a target message m with an unknown label to one of the classes $c \in \{\text{spam}, \text{ham}\}$. First, the method measures the increase of the description length of the data set as a result of the addition of the target document. Finally, it chooses the class for which the description length increase is minimal.

We consider in this work, each class (model) c as a sequence of tokens extracted from the messages and inserted into the training set. Each token t from m has a code length L_t based on the sequence of tokens presented in the messages of the training set of c. The length of m when assigned to the class c corresponds to the sum of all code lengths associated with each token of m, $Lm = \sum_{i=1}^{|m|} L_{t_i}$. We calculate $L_{t_i} = \lceil -\log_2 P_{t_i} \rceil$, where P is a probability distribution related with the tokens of class. Let $n_c(t_i)$ be the number of times that t_i appears in messages of class c, then the probability that any token belongs to c is given by the maximum likelihood estimation:

$$P_{t_i} = \frac{n_c(t_i) + \frac{1}{|\Delta|}}{n_c + 1}$$

where n_c corresponds to the sum of $n_c(t_i)$ for all tokens that appear in messages that belongs to c and $|\Delta|$ is the vocabulary size. In this work, we set $|\Delta| = 2^{32}$, that is, each token in an uncompress mode is a symbol with 32 bits. This estimation reserves a "portion" of probability to terms that the classifier has never seen before.

The proposed spam filter classifies a message using the following steps:

(1) Tokenization: the classifier extracts all tokens of the message $m = \{t_1, \ldots, t_{|m|}\}$.
(2) The method calculates the increase of the description length when m is assigned to each class $c \in \{\text{spam}, \text{ham}\}$ by the following equations:

$$L_m(\text{spam}) = \sum_{i=1}^{|m|} \left\lceil -\log_2 \left(\frac{n_{\text{spam}}(t_i) + \frac{1}{|\Delta|}}{n_{\text{spam}} + 1} \right) \right\rceil$$

$$L_m(\text{ham}) = \sum_{i=1}^{|m|} \left\lceil -\log_2 \left(\frac{n_{\text{ham}}(t_i) + \frac{1}{|\Delta|}}{n_{\text{ham}} + 1} \right) \right\rceil$$

(3) If $L_m(\text{spam}) < L_m(\text{ham})$, then m is classified as spam; otherwise, m is labeled as ham.
(4) If necessary, the training method is called.

In the following, we offer more information regarding steps 1 and 4.

3.2. Preprocessing

Tokenization involves breaking the text stream into tokens, usually by means of a regular expression. We consider in this work that tokens start with a printable character, followed by any number of alphanumeric characters, excluding dots, commas, and colons from the middle of the pattern. With this pattern, domain names and mail addresses will be split at dots, so the classifier can recognize a domain even if subdomains vary [28]. As proposed by Drucker *et al.* [12] and Metsis *et al.* [29], we do not consider the number of times a token appears in each message. So, each token is computed only once per message it appears.

It is important to observe that we did not perform language-specific preprocessing techniques, such as word stemming, stop word removal, or case folding. However, we use an e-mail-specific preprocessing before the

Table I. Enron 1 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	91.33	85.09	93.48	96.36	92.86	0.831
MN TF NB	82.00	73.21	87.77	92.29	86.10	0.676
MN Bool NB	82.67	60.19	77.72	91.67	79.15	0.560
MV Bern NB	72.67	60.56	80.71	87.87	78.38	0.508
Bool NB	96.00	52.55	64.67	97.54	73.75	0.551
Gauss NB	85.33	89.51	95.92	94.13	92.86	0.824
Flex Bayes	86.67	88.44	95.38	94.61	92.86	0.825
SVM	83.33	87.41	95.11	93.33	91.70	0.796
MDL	92.00	92.62	97.01	96.75	**95.56**	**0.892**

classification stage. We employ the Jaakko Hyvattis normalizemime.[§] This algorithm converts the character set to UTF-8, decoding Base64, Quoted-Printable, and URL encoding, and adding warn tokens in case of encoding errors. It also appends a copy of HTML/XML message bodies with most tags removed, decodes HTML entities, and limits the size of attached binary files.

3.3. Training stage

The training stage is basically responsible to update and store the number of times each token appears in the messages of each class. Therefore, for each message $m = \{t_1, \ldots, t_{|m|}\}$ to be trained, the MDL spam filter performs the following simple steps:

For each token t_i of m do the following:

(1) Search for t_i in the training database.
(2) If t_i is found, then update the number of messages on the class of m that t_i has appeared; otherwise insert t_i in the database.

An advantage of the MDL classifier is that we can start with an empty training set, and according to the user feedback, the classifier builds the models for each class. Moreover, it is not necessary to keep the messages used for training because the models are incrementally being built by the TFs. As the tokens presented in the training set are kept in a lexicographical order, the computational complexity to train each message is in the order of $O(|m| \cdot \log n)$, where $|m|$ is the number of tokens presented in the message and n is the amount of tokens in the training set. Therefore, besides that the proposed approach is incrementally updateable, it is also very fast to construct, especially when compared with other established methods. Note that, for training, the NB classifier has a computational complexity equivalent to $O(|m| \cdot n)$ [29,11] and the linear SVM $O(|m| \cdot n^2)$ [30].

A basic training approach is to start with an empty model, classify each new sample, and train it in the right class if the classification is wrong. This is known as train on error (TOE). An improvement to this method is to train also when the classification is right, but the score is near the boundary – that is, train on near error (TONE) [28].

The advantage of TONE over TOE is that it accelerates the learning process by exposing the filter to additional hard-to-classify samples in the same training period. Therefore, we employ the TONE as training method used by the proposed spam filter.

4. EXPERIMENTS AND RESULTS

First, we evaluated the proposed approach using the six well-known, large, real and public Enron datasets.[¶] The corpora are composed by legitimate messages extracted from the mailboxes of six former employees of the Enron Corporation. For further details about the dataset statistics, refer to Metsis *et al*. [29].

Tables I–VI present the performance achieved by each classifier for each Enron dataset. We highlight the highest score in bold.

According to Cormack [1], the filters should be judged along four dimensions: autonomy, immediacy, spam identification, and non-spam identification. However, it is not obvious how to measure any of these dimensions separately, nor how to combine these measurements into a single one for the purpose of comparing filters.

As pointed out by Almeida *et al*. [11], to provide a fair evaluation, we consider as the most important measures the Matthews correlation coefficient (*MCC*) [31] and the weighted accuracy rate (Acc_w %) [25] achieved by each filter.

The *MCC* provides a balanced evaluation of the prediction, especially if the two classes are of different sizes [11,32]. It is given by the following equation.

$$MCC = \frac{(A \cdot B) - (C \cdot D)}{\sqrt{(A + C) \cdot (A + D) \cdot (B + C) \cdot (B + D)}}$$

[§]Available at http://hyvatti.iki.fi/jaakko/spam.

[¶]The Enron datasets are available at http://www.iit.demokritos.gr/skel/i-config/.

DOI: 10.1002/sec

Table II. Enron 2 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	80.00	97.56	99.31	93.53	94.38	0.850
MN TF NB	75.33	96.58	99.08	92.13	93.02	0.812
MN Bool NB	76.00	98.28	99.54	92.36	93.53	0.827
MV Bern NB	66.00	81.82	94.97	89.06	87.56	0.657
Bool NB	95.33	81.25	92.45	98.30	93.19	0.836
Gauss NB	75.33	98.26	99.54	92.16	93.36	0.823
Flex Bayes	64.00	97.96	99.54	88.96	90.46	0.743
SVM	90.67	90.67	96.80	96.80	95.23	0.875
MDL	91.33	99.28	99.77	97.10	**97.31**	**0.937**

Table III. Enron 3 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	58.00	100.00	100.00	86.45	88.59	0.708
MN TF NB	62.00	100.00	100.00	87.58	89.67	0.737
MN Bool NB	60.00	100.00	100.00	87.01	89.13	0.723
MV Bern NB	100.00	85.23	93.53	100.00	95.29	0.893
Bool NB	95.33	87.73	95.02	98.20	95.11	0.881
Gauss NB	55.33	97.65	99.50	85.65	87.50	0.676
Flex Bayes	52.67	97.53	99.50	86.78	72.83	0.656
SVM	91.33	96.48	98.76	96.83	96.74	0.917
MDL	90.00	100.00	100.00	96.40	**97.28**	**0.931**

Table IV. Enron 4 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	95.33	100.00	100.00	87.72	96.50	0.914
MN TF NB	94.00	100.00	100.00	84.75	95.50	0.893
MN Bool NB	97.11	100.00	100.00	92.02	97.83	0.945
MV Bern NB	98.22	100.00	100.00	94.94	98.67	0.966
Bool NB	98.00	100.00	100.00	94.34	98.50	0.962
Gauss NB	94.22	100.00	100.00	85.23	95.67	0.896
Flex Bayes	95.78	100.00	100.00	88.76	96.83	0.922
SVM	98.89	100.00	100.00	96.77	**99.17**	**0.978**
MDL	97.11	100.00	100.00	92.02	97.83	0.945

Table V. Enron 5 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	90.22	98.81	97.33	80.22	92.28	0.832
MN TF NB	88.59	100.00	100.00	78.12	91.89	0.832
MN Bool NB	95.11	100.00	100.00	89.29	96.53	0.922
MV Bern NB	98.10	91.86	78.67	94.40	92.47	0.814
Bool NB	85.87	100.00	100.00	74.26	89.96	0.799
Gauss NB	88.59	99.39	98.67	77.89	91.51	0.821
Flex Bayes	91.58	98.54	96.67	82.39	93.05	0.845
SVM	89.40	99.70	99.33	79.26	92.28	0.837
MDL	99.73	98.39	96.00	99.31	**98.65**	**0.967**

Here, A, B, C, and D correspond to the amount of true positive, true negative, false positive, and false negative samples, respectively.

The *MCC* returns a value inside a predefined range, which provides more information about the classifiers performance. It returns a real value between -1 and $+1$.

Table VI. Enron 6 – Results achieved by each filter.

Classifiers	*Sre*(%)	*Spr*(%)	*Lre*(%)	*Lpr*(%)	Acc_w(%)	*MCC*
Basic NB	87.78	99.00	97.33	72.64	90.17	0.781
MN TF NB	75.78	99.42	98.67	57.59	81.50	0.651
MN Bool NB	93.33	97.45	92.67	82.25	93.17	0.828
MV Bern NB	96.00	92.31	76.00	86.36	91.00	0.753
Bool NB	66.67	99.67	99.33	49.83	74.83	0.572
Gauss NB	89.56	98.05	94.67	75.13	90.83	0.785
Flex Bayes	94.22	97.03	91.33	84.05	93.50	0.833
SVM	89.78	95.28	86.67	73.86	89.00	0.727
MDL	98.67	95.48	86.00	95.56	**95.50**	**0.878**

A coefficient equals to + 1 indicates a perfect prediction; 0, an average random prediction; and −1, an inverse prediction.

Additionally, we present other well-known measures, such as: spam recall (*Sre*%), legitimate recall (*Lre*%), spam precision (*Spr*%), and legitimate precision (*Lpr*%).

The results achieved by the MDL spam filter are compared with the ones attained by methods considered the actual top performers in spam filtering: seven different models of NB classifiers (Basic NB [6], Multinomial term frequency NB [MN TF NB] [33], Multinomial Boolean NB [MN Bool NB] [34], Multivariate Bernoulli NB [MV Bern NB] [35], Boolean NB [Bool NB] [29], Multivariate Gauss NB [Gauss NB] [29], and Flexible Bayes [Flex Bayes] [36]); and linear SVM with Boolean attributes [12–15].

Table VII summarizes all compared NB spam filters. For further information, consult Almeida *et al*. [9–11].

A comprehensive set of results, including all tables and figures, is available at http://www.dt.fee.unicamp.br/tiago/research/spam/spam.htm.

The MDL spam filter outperformed the compared methods for the majority e-mail datasets used in our empirical evaluation. Note that, in some situations, the MDL performs much better than SVM and NB classifiers. For instance, for Enron 1 (Table I), MDL achieved spam recall rate equal to 92%, whereas SVM attained 83.33%. It means that for Enron 1 MDL was able to recognize over 8% more of spams than SVM, representing an improvement of 10.40%. The same result can be found for Enron2 (Table II), Enron 5 (Table V), and Enron 6 (Table VI). Both methods, MDL and SVM, achieved similar performance with no significant statistical difference only for Enron 3 (Table III) and Enron 4 (Table IV).

The results show that the data compression model is more efficient to distinguish messages as spams or hams than other compared spam filters. It achieved an impressive average accuracy rate higher than 97%, and high precision recall rates for all datasets indicating that the MDL spam filter makes few mistakes. We also verify that the MDL classifier achieved an average *MCC* score higher than 0.925 for all tested e-mail collections. It clearly indicates that the proposed filter almost accomplished a perfect prediction ($MCC = 1.000$), and it is much better than not using a filter ($MCC = 0.000$).

Among the evaluated NB classifiers, the results indicate that all of them achieved similar performance with no significant statistical difference. However, they achieved lower results than MDL and SVM, which attained an accuracy rate higher than 90% for the most of Enron spam datasets.

To reinforce the validation, we performed another experiment to evaluate the MDL spam filter on more

Table VII. Naïve Bayes spam filters.

		Complexity on	
NB classifier	$P(\rightarrow x\|c_i)$	Training	Classification
Basic NB	$\prod_{k=1}^{\|m\|} P(t_k\|c_i)$	$O(\|m\| \cdot n)$	$O(\|m\|)$
MN TF NB	$\prod_{k=1}^{\|m\|} P(t_k\|c_i)^{x_k}$	$O(\|m\| \cdot n)$	$O(\|m\|)$
MN Boolean NB	$\prod_{k=1}^{\|m\|} P(t_k\|c_i)^{x_k}$	$O(\|m\| \cdot n)$	$O(\|m\|)$
MV Bernoulli NB	$\prod_{k=1}^{\|m\|} P(t_k\|c_i)^{x_k} \cdot (1 - P(t_k\|c_i))^{(1-x_k)}$	$O(\|m\| \cdot n)$	$O(\|m\|)$
Boolean NB	$\prod_{k=1}^{\|m\|} P(t_k\|c_i)$	$O(\|m\| \cdot n)$	$O(\|m\|)$
MV Gauss NB	$\prod_{k=1}^{\|m\|} g(x_k; \mu_{k,c_i}, \sigma_{k,c_i})$	$O(\|m\| \cdot n)$	$O(\|m\|)$
Flexible Bayes	$\prod_{k=1}^{\|m\|} \frac{1}{L_{k,c_i}} \sum_{l=1}^{L_{k,c_i}} g(x_k; \mu_{k,c_i,l}, \sigma_{c_i})$	$O(\|m\| \cdot n)$	$O(\|m\| \cdot n)$

realistic and larger datasets: TREC06 and CEAS08. The TREC Spam Track and CEAS Spam Challenge are the largest and most realistic laboratory evaluations to date. The main task at TREC 2006 – the *immediate feedback* task – simulates the online deployment of a spam filter with idealized user feedback [1]. On the other hand, the 2008 CEAS Spam Challenge evaluated the filters using a *live simulation* task that offers delayed partial feedback in which feedback is delayed and given only for messages addressed to a subset of the recipients. The main goal is to simulate a real mailbox in order to explore the impact of imperfect or limited user feedback.

To offer fair evaluation and reproducible results, we followed exactly the same guidelines used in the challenges including the same measures: spam caught (%), blocked ham (%), *LAM*% and $(1-AUC)$% [1].

The Logistic Average Misclassification ($LAM\,\%$) is a single quality measure, based only on the filter's binary classifications. Let the false positive rate (*fpr*) and false negative rate (*fnr*), $LAM\,\%$ is given by

$$LAM\,\% = \text{logit}^{-1}\left(\frac{\text{logit}(fpr) + \text{logit}(fnr)}{2}\right)$$

where

$$\text{logit}(p) = \log\left(\frac{p}{100-p}\right)$$

This measure imposes no *a priori* relative importance on ham or spam misclassification, and rewards equally a fixed-factor improvement in the odds of either [1]. The lower the $LAM\,\%$, the better the performance.

The area under the receiver operating characteristic curve (AUC) provides an estimate of the effectiveness of a soft classifier over all threshold filtering settings. AUC also has a probabilistic interpretation: it is the probability that the classifier will award a random spam message a higher score than a random ham message. The $(1-AUC)\,\%$ is often used instead of AUC because, in spam filtering, the AUC value is very close to 1 [1].

Again, we compared the performance achieved by the MDL spam filter against the following:

- NB classifier: the open-source and well-known Bogofilter classifier (version 0.94.0, default parameters – available at http://www.bogofilter.org).
- SVM classifier: relaxed linear SVM [37].

Table VIII present the results achieved by MDL spam filter and established spam filters on immediate feedback task for the TREC06 data set. Table IX present the results achieved by the filters for CEAS08 data set on live simulation. The results by each e-mail collection are ordered by decreasing performance in the $(1-AUC)\,\%$ statistic.

Table VIII. TREC 2006 – Results achieved by each filter.

Filters	$(1-AUC)\,\%$	*SC*%	*BH*%	*LAM*%
MDL	0.053	96.92	0.19	0.52
SVM	0.060	99.51	0.71	0.62
NB	0.084	91.77	0.00	0.00

Table IX. CEAS 2008 – Results achieved by each filter.

Filters	$(1-AUC)\,\%$	*SC*%	*BH*%	*LAM*%
MDL	0.0129	95.79	0.08	0.21
NB	0.0157	89.04	0.00	0.00
SVM	0.0221	98.96	0.51	0.49

The MDL spam filter achieved very good performance for all tested e-mail data sets independent on the task. If we take into account that the $(1-AUC)\,\%$ is the standard measure used in the spam challenges, we can conclude that the proposed approach outperforms the top-performance spam filters. Note that, although the MDL has not achieved the best $SC\,\%$ rates, it attained very low $BH\,\%$ and $LAM\,\%$ rates for all tested data sets. It indicates that the proposed filter rarely misclassify a legitimate message. It is a remarkable feature because in spam filtering false positives are considered much worse than false negatives. Despite the fact that Bogofilter (NB classifier) attained the best $BH\,\%$ rates for all used e-mail data sets, it achieved the worst $SC\,\%$ rates.

5. CONCLUSIONS AND FURTHER WORK

In this paper, we have presented a spam filtering approach based on data compression model that have proved to be fast to construct and incrementally updateable. We have also compared its performance with the well-known linear SVM and different models of NB classifiers, something the spam literature does not always acknowledge.

To evaluate the proposed method, we have conducted empirical experiments using real, large, and public datasets, and the results indicate that the proposed classifier outperforms currently established spam filters.

Actually, we are conducting more experiments to compare our approach with other established commercial and open-source spam filters, such as SpamAssassin, ProcMail, OSBF-Lua, and others.

ACKNOWLEDGEMENTS

This work is partially supported by the Brazilian funding agencies CNPq, CAPES, and FAPESP.

REFERENCES

1. Cormack G. Email spam filtering: a systematic review. *Foundations and Trends in Information Retrieval* 2008; **1**(4): 335–455.
2. Joachims T. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. *Proceedings of 14th International Conference on Machine Learning*, Nashville, TN, USA, 1997; 143–151.
3. Schapire R, Singer Y, Singhal A. Boosting and Rocchio applied to text filtering. *Proceedings of the 21st Annual International Conference on Information Retrieval*, Melbourne, Australia, 1998; 215–223.
4. Carreras X, Marquez L. Boosting trees for anti-spam email filtering. *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, Bulgaria, 2001; 58–64.
5. Reddy C, Park JH. Multi-Resolution Boosting for Classification and Regression Problems. *Knowledge and Information Systems* 2011; **29**(2): 435–456.
6. Sahami M, Dumais S, Hecherman D, Horvitz E. A bayesian approach to filtering junk e-mail. *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, USA, 1998; 55–62.
7. Androutsopoulos I, Koutsias J, Chandrinos K, Paliouras G, Spyropoulos C. An evalutation of Naive Bayesian anti-spam filtering. *Proceedings of the 11st European Conference on Machine Learning*, Barcelona, Spain, 2000; 9–17.
8. Song Y, Kolcz A, Gilez C. Better Naive Bayes classification for high-precision spam detection. *Software – Practice and Experience* 2009; **39**(11): 1003–1024.
9. Almeida T, Yamakami A, Almeida J. Evaluation of approaches for dimensionality reduction applied with Naive Bayes anti-spam filters. *Proceedings of the 8th IEEE International Conference on Machine Learning and Applications*, Miami, FL, USA, 2009; 517–522.
10. Almeida T, Yamakami A, Almeida J. Probabilistic anti-spam filtering with dimensionality reduction. *Proceedings of the 25th ACM Symposium On Applied Computing*, Sierre, Switzerland, 2010; 1804–1808.
11. Almeida T, Almeida J, Yamakami A. Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers. *Journal of Internet Services and Applications* 2011; **1**(3): 183–200.
12. Drucker H, Wu D, Vapnik V. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks* September 1999; **10**(5): 1048–1054.
13. Kolcz A, Alspector J. SVM-based filtering of e-mail spam with content-specific misclassification costs. *Proceedings of the 1st International Conference on Data Mining*, San Jose, CA, USA, 2001; 1–14.
14. Hidalgo J. Evaluating cost-sensitive unsolicited bulk email categorization. *Proceedings of the 17th ACM Symposium on Applied Computing*, Madrid, Spain, 2002; 615–620.
15. Forman G, Scholz M, Rajaram S. Feature shaping for linear SVM classifiers. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009; 299–308.
16. Almeida T, Yamakami A. Content-based spam filtering. *Proceedings of the 23rd IEEE International Joint Conference on Neural Networks*, Barcelona, Spain, 2010; 1–7.
17. Carpinter J, Hunt R. Tightening the net: a review of current and next generation spam filtering tools. *Computers and Security* 2006; **25**(8): 566–578.
18. Blanzieri E, Bryl A. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review* 2008; **29**(1): 335–455.
19. Guzella T, Caminhas W. A review of machine learning approaches to spam filtering. *Expert Systems with Applications* September 2009; **36**(7): 10 206–10 222.
20. Caruana G, Li M. A survey of emerging approaches to spam filtering. *ACM Computing Surveys* 2012; **44**(2): 1–27.
21. Rissanen J. Modeling by shortest data description. *Automatica* 1978; **14**: 465–471.
22. Barron A, Rissanen J, Yu B. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory* 1998; **44**(6): 2743–2760.
23. Grünwald P. A tutorial introduction to the minimum description length principle. *Advances in Minimum Description Length: Theory and Applications*, Grünwald P, Myung I, Pitt M (eds). MIT Press: Cambridge, MA, USA, 2005; 3–81.
24. Almeida T, Yamakami A, Almeida J. Filtering spams using the minimum description length principle. *Proceedings of the 25th ACM Symposium On Applied Computing*, Sierre, Switzerland, 2010; 1856–1860.
25. Androutsopoulos I, Paliouras G, Michelakis E. Learning to filter unsolicited commercial e-mail. *Technical Report 2004/2*, National Centre for Scientific Research "Demokritos", Athens, Greece March 2004.
26. Teahan W, Harper D. Using compression-based language models for text categorization. *Proceedings of the 2001 Workshop on Language Modeling and Information Retrieval*, Pittsburgh, PA, USA, 2001; 1–5.
27. Frank E, Chui C, Witten I. Text categorization using compression models. *Proceedings of the 10th Data Compression Conference*, Snowbird, UT, USA, 2000; 555–565.
28. Siefkes C, Assis F, Chhabra S, Yerazunis W. Combining winnow and orthogonal sparse bigrams for incremental

spam filtering. *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Pisa, Italy, 2004; 410–421.
29. Metsis V, Androutsopoulos I, Paliouras G. Spam filtering with Naive Bayes – which Naive Bayes? *Proceedings of the 3rd International Conference on Email and Anti-Spam*, Mountain View, CA, USA, 2006; 1–5.
30. Bordes A, Ertekin S, Weston J, Bottou L. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research* 2005; **6**: 1579–1619.
31. Matthews B. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta* October 1975; **405**(2): 442–451.
32. Baldi P, Brunak S, Chauvin Y, Andersen C, Nielsen H. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* May 2000; **16**(5): 412–424.
33. McCallum A, Nigam K. A comparison of event models for Naive Bayes text classification. *Proceedings of the 15th AAAI Workshop on Learning for Text Categorization*, Menlo Park, CA, USA, 1998; 41–48.
34. Schneider K. On word frequency information and negative evidence in Naive Bayes text classification. *Proceedings of the 4th International Conference on Advances in Natural Language Processing*, Alicante, Spain, 2004; 474–485.
35. Losada D, Azzopardi L. Assessing multivariate Bernoulli models for information retrieval. *ACM Transactions on Information Systems* June 2008; **26**(3): 1–46.
36. John G, Langley P. Estimating continuous distributions in Bayesian classifiers. *Proceedings of the 11st International Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 1995; 338–345.
37. Sculley D, Wachman G, Brodley C. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. *Proceedings of the 15th Text Retrieval Conference*, Gaithersburg, MD, USA, 2006; 1–10.