Journal of Machine Learning Research 7 (2006) 2673-2698 Submitted 03/06; Revised 09/06; Published 12/06

Spam Filtering Using Statistical Data Compression Models

Andrej Bratko ANDREJ.BRATKO@IJS.SI
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia SI-1000

Gordon V. Cormack GVCORMACK@UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Bogdan Filipič BOGDAN.FILIPIC@IJS.SI
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia SI-1000

Thomas R. Lynam TRLYNAM@PLG.UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Blaž Zupan BLAZ.ZUPAN@FRI.UNI-LJ.SI
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25, Ljubljana, Slovenia SI-1000

Editor: Philip Chan

Abstract

Spam filtering poses a special problem in text categorization, of which the defining characteristic is that filters face an active adversary, which constantly attempts to evade filtering. Since spam evolves continuously and most practical applications are based on online user feedback, the task calls for fast, incremental and robust learning algorithms. In this paper, we investigate a novel approach to spam filtering based on adaptive statistical data compression models. The nature of these models allows them to be employed as probabilistic text classifiers based on character-level or binary sequences. By modeling messages as sequences, tokenization and other error-prone preprocessing steps are omitted altogether, resulting in a method that is very robust. The models are also fast to construct and incrementally updateable. We evaluate the filtering performance of two different compression algorithms; dynamic Markov compression and prediction by partial matching. The results of our empirical evaluation indicate that compression models outperform currently established spam filters, as well as a number of methods proposed in previous studies.

Keywords: text categorization, spam filtering, Markov models, dynamic Markov compression, prediction by partial matching

1. Introduction

Electronic mail is arguably the "*killer app*" of the internet. It is used daily by millions of people to communicate around the globe and is a mission-critical application for many businesses. Over the last decade, unsolicited bulk email has become a major problem for email users. An overwhelming amount of spam is flowing into users' mailboxes daily. In 2004, an estimated 62% of all email was attributed to spam, according to the anti-spam outfit Brightmail.[1] Not only is spam frustrating for most email users, it strains the IT infrastructure of organizations and costs businesses billions of dollars in lost productivity. In recent years, spam has evolved from an annoyance into a serious security threat, and is now a prime medium for phishing of sensitive information, as well the spread of malicious software.

Many different approaches for fighting spam have been proposed, ranging from various sender authentication protocols to charging senders indiscriminately, in money or computational resources (Goodman et al., 2005). A promising approach is the use of content-based filters, capable of discerning spam and legitimate email messages automatically. Machine learning methods are particularly attractive for this task, since they are capable of adapting to the evolving characteristics of spam, and data is often available for training such models. Nevertheless, spam filtering poses a special problem for automated text categorization, of which the defining characteristic is that filters face an active adversary, which constantly attempts to evade filtering. Unlike most text categorization tasks, the cost of misclassification is heavily skewed: Labeling a legitimate email as spam, usually referred to as a *false positive*, carries a much greater penalty than vice-versa. Since spam evolves continuously and most practical applications are based on online user feedback, the task calls for fast, incremental and robust learning algorithms.

In this paper, we consider the use of adaptive data compression models for spam filtering. Specifically, we employ the dynamic Markov compression (Cormack and Horspool, 1987) and prediction by partial matching (Cleary and Witten, 1984) algorithms. Classification is done by first building two compression models from the training corpus, one from examples of spam and one from legitimate email. The compression rate achieved using these two models on the target message determines the classification outcome. Two variants of the method with different theoretical underpinnings are evaluated. The first variant (Frank et al., 2000; Teahan, 2000) estimates the probability of a document using compression models derived from the training data, and assigns the class label based on the model that deems the target document most probable. The second variant, which is introduced in this paper, selects the class for which the addition of the target document results in a minimal increase in the description length of the entire data set. The difference in practice is that the models are used adaptively in this second approach, that is, they are updated at each position in the target document so that statistics from the initial part of the document are taken into account when examining the remainder. In this way, repeated occurrences of similar, "*already seen*", patterns, have diminishing influence on the classification outcome, thereby putting greater weight on heterogeneous evidence.

While the idea of using data compression algorithms for text categorization is not new, we are aware of no existing research that considers such methods for spam filtering. In the present paper, we demonstrate that compression models are extremely well suited to the spam filtering problem. We propose a simple, yet effective modification of the original method, which substantially improves

[1] `http://brightmail.com/`, 2004-03-12

filtering performance in our experiments. We generalize the results to compression algorithms not considered in other studies, showing that they exhibit similar, strong performance.

In online learning experiments on three large email collections, we find that compression models generally outperform established spam filters according to several measures. On the TREC public corpus, currently the largest publicly available spam data set, spam misclassification at a false positive rate of 1 in 1000 is 1.2% – 1.8% (depending on the compression algorithm), compared to 2.6% – 6.9% achieved by six reference systems. We also conducted cross validation experiments on the Ling-Spam, PU1 and PU3 data sets, in which compression models compare favorably to a variety of methods considered in previous studies on the same data. Finally, we show that compression models are robust to the type of noise introduced in text by obfuscation tactics which are commonly used by spammers against tokenization-based filters.

2. Related Work

The basic idea of using data compression in classification and other machine learning tasks has been reinvented many times. The intuition arises from the principal observation that compact representations of objects are only possible after some recurring patterns or statistical regularities are detected. This has motivated many applications of general-purpose data compression algorithms in machine learning and data mining problems, in which data compression algorithms are most often used to produce a distance or (dis)similarity measure between pairs of data points (Li et al., 2003; Keogh et al., 2004; Sculley and Brodley, 2006).

Most relevant to our study is the work of Frank et al. (2000), who first proposed the use of compression models for automated text categorization. They investigate using the prediction by partial matching (PPM) algorithm as a Bayesian text classifier, that is, they build a PPM model of each class and use these models to estimate the class-conditional probabilities of the target document. They find that the compression-based method is inferior to support vector machines (SVM) and roughly on par with naive Bayes on the classical Reuters-21578 data set. The same method was investigated by Teahan (2000), and later applied to a number of text categorization problems, such as authorship attribution, dialect identification and genre classification (Teahan and Harper, 2003). They find the compression-based method particularly suitable for dialect identification and authorship attribution, and report fair performance for genre classification and topic detection.

Peng et al. (2004) propose augmenting a word-based naive Bayes classifier with statistical language models to account for word dependencies. They also consider training the language models on characters instead of words, resulting in a method that is effectively very similar to the compression-based method of Frank et al. (2000). In experiments on a number of categorization tasks, they find that the character-based method often outperforms the word-based approach. They also find that character-level language models reach or improve previously published results on four of the six classification tasks considered in the study.

In most spam filtering work, text is modeled with the bag-of-words (BOW) representation, even though it is widely accepted that tokenization is a vulnerability of keyword-based spam filters. Some filters use character n-grams instead of word tokens and apply standard machine learning algorithms on the resulting feature vector (Goodman et al., 2005).

Two particular approaches (that we are aware of) go a step further towards operating directly on character sequences. IBM's Chung-Kwei system (Rigoutsos and Huynh, 2004) uses pattern matching techniques originally developed for DNA sequences. Messages are filtered based on the number of occurrences of patterns associated with spam and the extent to which they cover the target document. Recently, Pampapathi et al. (2006) proposed a filtering technique based on the suffix tree data structure. They investigate a number of ad hoc scoring functions on matches found in the target document against suffix trees constructed from spam and legitimate email. However, the techniques proposed in these studies are different to the statistical data compression models that are evaluated here. In particular, while these systems use character-based features in combination with some ad hoc scoring functions, compression models were designed from the ground up for the specific purpose of *probabilistic* modeling of sequential data. This property of data compression models allows them to be employed in an intuitively appealing and principled way.

The methods presented in this paper were first evaluated on a number of real-world email collections in the framework of the 2005 Text REtrieval Conference (TREC). The results of this evaluation showed promise in the use of statistical data compression models for spam filtering (Bratko and Filipič, 2005). This article describes the methods used at TREC in greater detail and extends the TREC paper in a number of ways: We compare the use of adaptive and static models for classification, extend our analysis to compression algorithms not considered in previous work, compare the performance of compression models with results published in other studies, and evaluate the effect of noise in text on the filtering performance of compression models and tokenization-based filters.

3. Statistical Data Compression

Probability plays a central role in data compression: Knowing the exact probability distribution governing an information source allows us to construct optimal or near-optimal codes for messages produced by the source. A statistical data compression algorithm exploits this relationship by building a statistical model of the information source, which can be used to estimate the probability of each possible message. This model is coupled with an encoder that uses these probability estimates to construct the final binary representation. For our purposes, the encoding problem is irrelevant. We therefore focus on the source modeling task.

3.1 Preliminaries

We denote by X the random variable associated with the source, which may take the value of any message the source is capable of producing, and by P the probability distribution over the values of X with the corresponding probability mass function p. We are particularly interested in modeling of text generating sources. Each message $\mathbf{x}$ produced by such a source is naturally represented as a sequence $\mathbf{x} = x_1^n = x_1 \ldots x_n \in \Sigma^*$ of symbols over the source alphabet Σ. The length $|\mathbf{x}|$ of a sequence can be arbitrary. For text generating sources, it is common to interpret a symbol as a single character, but other schemes are possible, such as binary (bitwise) or word-level models.

The entropy $H(X)$ of a source X gives a lower bound on the average per-symbol code length required to encode a message without loss of information:

$$H(X) = E_{\mathbf{x} \sim P}\left(-\frac{1}{|\mathbf{x}|} \log p(\mathbf{x}) \right).$$

This bound is achievable *only* when the true probability distribution P governing the source is known. In this case, an average message could be encoded using no less than $H(X)$ bits per symbol. However, the true distribution over all possible messages is typically unknown. The goal of any statistical data compression algorithm is then to infer a probability mass function over sequences

$f : \Sigma^* \rightarrow [0,1]$, which matches the true distribution of the source as accurately as possible. Ideally[2], a sequence $\mathbf{x}$ is then encoded with $L(\mathbf{x})$ bits, where

$$L(\mathbf{x}) = -\log f(\mathbf{x}).$$

The compression algorithm must therefore *learn* an approximation of P in order to encode messages efficiently. A better approximation will, on average, lead to shorter code lengths. This simple observation alone gives compelling motivation for the use of compression algorithms in text categorization, but we defer this discussion until Section 5, and first describe the compression models used in the study.

3.2 Finite Memory Markov Sources

To make the inference problem over all (possibly infinite) sequences tractable, sources are usually modeled as stationary and ergodic Markov sources[3] with limited memory k. Each symbol in a message $\mathbf{x}$ is therefore assumed to be independent of all but the preceding k symbols (the *context*):

$$p(\mathbf{x}) = \prod_{i=1}^{|\mathbf{x}|} p(x_i|x_1^{i-1}) \approx \prod_{i=1}^{|\mathbf{x}|} p(x_i|x_{i-k}^{i-1}).$$

We assume that a string of k leading symbols that otherwise cannot occur in any sequence is prepended to $\mathbf{x}$ to overcome the technical difficulty in estimating the first k symbols. In practice, a compression algorithm would normally use a shorter context for prediction here.

The number of context symbols k is referred to as the *order* of the Markov model. Higher order models have the potential to better approximate the characteristics of a complex source. However, since the number of possible contexts increases exponentially with context length, accurate parameter estimates are hard to obtain. For example, an order-k model requires $\Sigma^k(\Sigma - 1)$ parameters. To tackle this problem, different strategies are employed by different algorithms. The common ground to all such algorithms is that the complexity of the model is increased only after the amount of training data is sufficient to support a more complex model. We describe two particular algorithms that were also used in our experiments later in this section.

3.3 Two-part vs. Adaptive Coding

Two-part data compression methods first transmit the model which is used for encoding, followed by the encoded data. The decoder reads the model first and then uses this information to decode the remaining part of the message. Such methods require two passes over the data. The first pass is required to train the model and the second pass is required for the encoding.

Adaptive methods do not explicitly include the model in the encoded message. Rather, they start encoding the message using an empty model, for example, a uniform distribution over all symbols. The model is incrementally updated after each symbol, gradually adapting to an ever

[2] The “ideal” code length ignores the practical requirement that codes have an integer length.

[3] The stationarity and ergodicity properties are necessary preconditions for learning to stabilize arbitrarily close to (the best approximation of) the data generating process asymptotically, that is, as the length of the training sequence increases without bound. They are stated here for completeness and are usually taken for granted in a typical machine learning setting, where we expect to learn from past observations.

closer approximation of the data generating process. The probability assigned to a sequence **x** by an order-*k* adaptive algorithm is then

$$f(\mathbf{x}) = \prod_{i=1}^{|\mathbf{x}|} f(x_i | x_{i-k}^{i-1}, M(x_1^{i-1})) \tag{1}$$

where $M(x_1^{i-1})$ denotes the current model at time i, constructed from the input sequence x_1^{i-1}. The decoder repeats the learning process, building its own version of the model as the message is decoded. It is important to note that adaptive methods require only a single pass over the data, a property that we will turn to our advantage in subsequent developments.

3.4 Algorithms

In this section, we describe the dynamic Markov compression and prediction by partial matching compression algorithms with which we obtain our main results. Both of the algorithms are adaptive, that is, the model used for prediction may be updated efficiently after each symbol in a sequence.

3.4.1 DYNAMIC MARKOV COMPRESSION

The dynamic Markov compression (DMC) algorithm (Cormack and Horspool, 1987) models an information source with a finite state machine (FSM). A probability distribution over symbols is associated with each state and is used to predict the next binary digit emitted by the source. The algorithm begins in a predefined initial state and moves to the next state after each digit in the sequence. An example FSM structure, corresponding to an order-1 binary Markov model, is shown in the left side of Figure 1. Each state S in the FSM has two outbound transitions, one for each binary symbol. These transitions are equipped with frequency counts, from which next-symbol probabilities for the state S are calculated (as relative frequencies).

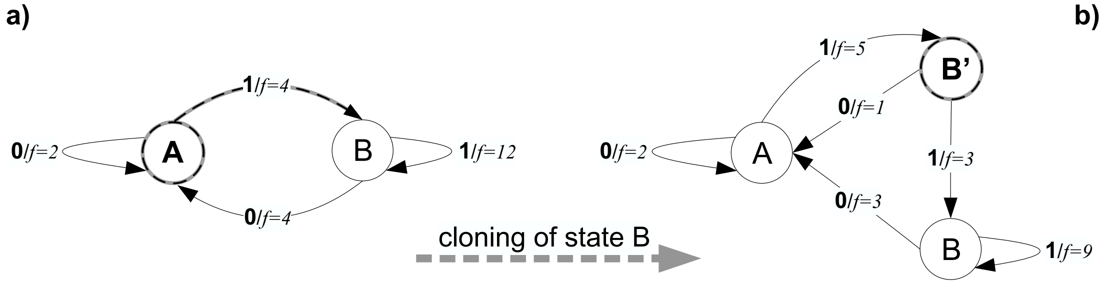


Figure 1: An example of DMC's state cloning operation. The active state and transition at time a) and b) are highlighted. The left hand side shows the model when state A is active and the observed symbol is '1'. This triggers the cloning of state B and a state transition to the new state B', as shown on the right hand side of the figure. The transition frequencies (visit counts) before and after the cloning operation are also shown.

In an adaptive DMC algorithm, the frequency counts of transitions are incremented whenever a transition fires (once after each symbol in the sequence). The structure of the state machine may also be built incrementally, by using a special state *cloning* operation. Specifically, as soon as the algorithm finds that a transition from some state A to some other state B in the FSM is used often, the target state of the transition is considered for cloning. Figure 1 depicts this cloning operation, in which a new state B' is spawned. The new state B' has a single inbound transition, which replaces the former transition from A to B. All outbound transitions of B are copied to B'.

After cloning state B, the FSM maintains separate statistics for situations when state B is reached from state A, and when it is reached from other states. Without loss of generality, suppose the former transition from A to B is associated with the symbol ‘1’, as in the example. The new state B' then corresponds to the situation when state A is followed by ‘1’. As cloning continues, new states begin to express more and more specific situations, allowing the algorithm to incorporate richer context information when predicting the next symbol. The context used for prediction is implicitly determined by the longest string of symbols that matches all suffixes of paths leading to the current state of the FSM.

After cloning, the statistics associated with the cloned state B are distributed among B' and B in proportion to the number of times state B was reached from state A, relative to the number of times state B was reached from other states (again, refer to Figure 1). Two parameters control the state cloning operation in DMC. These are the minimal frequencies of B' and B after cloning. Both of the new states must exceed this minimal frequency that is required for stable probability estimates in order to trigger the cloning operation. At each position in the sequence only one state needs to be considered for cloning: The target state of the transition in the FSM that is triggered by the current symbol.

In the most basic version, the initial model contains a single state, corresponding to a memoryless source. When dealing with byte-aligned data, it is customary to start with a slightly more complex initial state machine which is capable of expressing within-byte dependencies. This initial FSM structure corresponds to an order-7 binary Markov model. All transitions in the initial FSM are primed with a small initial visit count to avoid singular probabilities. We note that although DMC restricts the source alphabet to binary symbols, it nevertheless achieves state-of-the-art performance on typical ASCII encoded text sequences (Cormack and Horspool, 1987).

3.4.2 PREDICTION BY PARTIAL MATCHING

The prediction by partial matching (PPM) algorithm (Cleary and Witten, 1984) has set the standard for lossless text compression since its introduction over two decades ago (Cleary and Teahan, 1997). Essentially, the PPM algorithm is a back-off smoothing technique for finite-order Markov models, similar to back-off models used in natural language processing.

It is convenient to assume that an order-k PPM model stores a table of all contexts (up to length k) that occur anywhere in the training text. For each such context, the frequency counts of symbols that immediately follow it is maintained. When predicting the next symbol x_i in a sequence x_1^n, its context x_{i-k}^{i-1} is matched against the stored statistics. The longest matching context x_{i-l}^{i-1} found in the table is examined first (note that $l \leq k$). If the target symbol has appeared in this context in the training text, its relative frequency within the context is used for prediction. However, this probability is *discounted* by a small amount to reserve some probability mass, which is called the *escape probability*. The escape probability that is accumulated in this way estimates the probability of observing a zero-frequency symbol in the context x_{i-l}^{i-1}. The escape probability is distributed among symbols *not* seen in the current context, according to a lower-order model, that is, according to statistics for the context x_{i-l+1}^{i-1}. The procedure is applied recursively until all symbols receive a non-zero probability. If necessary, a default model of order -1 is used, which always predicts a uniform distribution among all possible symbols.

An adaptive compression algorithm based on the PPM model starts with an empty model which always defaults to the uniform distribution among all symbols. After each symbol is encoded, the algorithm updates the statistics of all contexts (up to order k) of the current symbol.

Many versions of the PPM algorithm exist, differing mainly in the way the escape probability is estimated. In our implementation, we used escape method D (Howard, 1993), which simply discounts the frequency of each observed character by $1/2$ occurrence and uses the gained probability mass for the escape probability.

4. Minimum Description Length Principle

The minimum description length (MDL) principle (Rissanen, 1978; Barron et al., 1998; Grünwald, 2005) favors models that yield compact representations of the data. The traditional two-part MDL principle states that the preferred 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.

A problem of the two-part MDL principle is that it gives no guidelines as to how the model should be encoded. The refined MDL principle which is described later in this section aims to remedy this problem.

4.1 Universal Codes

A universal code relative to a class of source models has the property that it compresses data “almost” as well as the best model in the model class. More precisely, the difference in code length between a universal code and the best model in the model class increases sublinearly with the length of the sequence. Rissanen gives a precise non-asymptotic lower bound on this difference in the worst case (Rissanen, 1986), which turns out to be linearly related to the complexity of the data generating process (in terms of the number of parameters). He also shows that codes exist that achieve this bound.

Two-part codes are universal, since only a finite code length is required to specify the model. It turns out that adaptive codes are also universal codes (Rissanen, 1984). In fact, adaptive compression algorithms exist that are proven to achieve Rissanen’s lower bound relative to the class of all finite-memory tree sources (e.g., Willems et al., 1995). The redundancy incurred due to the fact that adaptive methods start with an empty, uninformed model, can be compared to the cost of separately encoding the model in two-part codes.

4.2 Predictive MDL

The limitations of the original two-part MDL principle were largely overcome with the modern version of the principle (Rissanen, 1996), which advocates the use of one-part universal codes for measuring description length relative to a chosen model class. The use of adaptive codes for this task is sometimes denoted predictive MDL and is encouraged when the data is sequential in nature (Grünwald, 2005).

We aim to measure the description length of a set of documents relative to the class of Markov models of a certain order by using adaptive universal data compression algorithms, and to employ this measure as a criterion for classification. It is necessary to mention here that while PPM is universal in this sense, the same cannot be said for DMC. This is due to its “greedy” strategy of adapting its model without bound, that is, increasing the order of the model as soon as possible. On the other hand, this strategy might well lead to a better approximation of the source and thus more accurate prediction. In terms of data compression performance, DMC is competitive to PPM on the types of sequences that are of practical interest to us, particularly for natural language text and binary computer files (Cormack and Horspool, 1987).[4]

5. Text Classification Using Compression Models

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.

In the following subsections, we describe two approaches to classification. Both approaches model a class as an information source, and consider the training data for each class a sample of the type of data generated by the source. They differ in the way classification is performed. We first describe the minimum cross-entropy (MCE) approach (Frank et al., 2000; Teahan, 2000). This method chooses the class for which the associated compression model assigns the highest probability to the target document. We then propose a simple modification to this method, in which the model is *adapted* while evaluating the target document in the sense of Equation 1. Unlike the former approach, this method measures the increase of the description length of the data set as a result of the addition of the target document. It chooses the class for which the description length increase is minimal, which is why we consider this a minimum description length (MDL) approach. In subsequent sections, we also refer to this approach as using *adaptive* models and the MCE approach as using *static* models for obvious reasons.

We denote by C the set of classes and by $c : \Sigma^* \rightarrow C$ the (partially specified) function mapping documents to class labels. Given a set of pre-classified training documents D, the task is to assign a target document $\mathbf{d}$ with an unknown label to one of the classes $c \in C$.

5.1 Classification by Minimum Cross-entropy

The *cross-entropy* $H(X,M)$ determines the average number of bits per symbol required to encode messages produced by a source X when using a model M for compression:

$$H(X,M) = E_{\mathbf{x}\sim P}\left(\frac{1}{|\mathbf{x}|}L(\mathbf{x}|M)\right).$$

In the above equation, $L(\mathbf{x}|M)$ denotes the ideal code length for $\mathbf{x}$ under model M. Note that $H(X,M) \geq H(X)$ always holds, that is, the best possible model achieves a compression rate equal to the entropy.

[4] As a side-note, we mention here that the techniques presented in this paper were also evaluated in combination with the Context Tree Weighting (CTW) compression algorithm (Willems et al., 1995) in the framework of the TREC 2005 spam track (Bratko and Filipič, 2005). Although the CTW algorithm is universal and provably achieves Rissanen’s optimal minimax regret for the class of sources it considers, its performance for spam filtering in the TREC evaluation was comparable, although slightly inferior, to PPM. Since the CTW algorithm is also computationally less efficient, we omit the CTW algorithm from the present paper.

The exact cross-entropy is hard to compute, since it would require knowing the source distribution P. It can, however, be approximated by applying the model M to sufficiently long sequences of symbols, with the expectation that these sequences are representative samples of all possible sequences generated by the source (Brown et al., 1992; Teahan, 2000):

$$H(X,M) \approx \frac{1}{|\mathbf{x}|} L(\mathbf{x}|M). \tag{2}$$

As $|\mathbf{x}|$ becomes large, this estimate will approach the actual cross-entropy in the limit almost surely if the source is ergodic (Algoet and Cover, 1988). Recall that if M is a Markov model with limited memory k, then

$$L(\mathbf{x}|M) = -\log \prod_{i=1}^{|\mathbf{x}|} f(x_i|x_{i-k}^{i-1}, M)$$

where $f(x_i|x_{i-k}^{i-1}, M)$ is the probability assigned to x_i given x_{i-k}^{i-1} by M.

Following Teahan (2000), we refer to the cross-entropy estimated on the target document $\mathbf{d}$ as the *document* cross-entropy $H(X,M,\mathbf{d})$. This is simply a substitution of $\mathbf{x}$ with $\mathbf{d}$ in the right hand of Equation 2. We expect that a model that achieves a low cross-entropy on the target document approximates the information source that actually generated the document well. This is therefore our measure for classification:

$$\begin{aligned} c(\mathbf{d}) &= \arg\min_{c \in C} H(X, M_c, \mathbf{d}) \\ &= \arg\min_{c \in C} -\frac{1}{|\mathbf{d}|} \log \prod_{i=1}^{|\mathbf{d}|} f(d_i|d_{i-k}^{i-1}, M_c). \end{aligned} \tag{3}$$

In the above equation, M_c denotes the compression model built from all examples of class c in the training data.

5.2 Classification by Minimum Description Length

The MCE criterion assumes that the test document $\mathbf{d}$ was generated by some unknown information source. The document is considered a sample of the type of data generated by the unknown source. Classification is based on the distance between each class and the source that generated the document. This distance is measured with the document cross-entropy, which serves as an estimate of the cross-entropy between the unknown source and each of the class information sources.

However, we know that the document did not originate from some *unknown* source and that it ultimately must be attributed to one of the classes. The MDL classification criterion tests, for each class $c \in C$, the hypothesis that $c(d) = c$, by adding the document to the training data of the class and estimating how much this addition increases the description length of the data set:

$$\Delta L(D,c,\mathbf{d}) = L(\{\mathbf{x}\,; \mathbf{x} \in D, c(\mathbf{x}) = c)\} \cup \{\mathbf{d}\}) - L(\{\mathbf{x}\,; \mathbf{x} \in D, c(\mathbf{x}) = c)\}).$$

We are searching for the classification hypothesis that yields the most compact description of the observed data. The resulting description length is measured with adaptive compression algorithms which allow efficient estimation of this quantity, although other universal codes could also be used to measure the description length increase. This is in line with the approach suggested by Kontkanen et al. (2005) in their MDL framework for clustering, in which the cluster assignment should be such that it results in a minimal description length of the data, measured by a suitable universal model.

Adaptive models are particularly suitable for this type of classification, since they can be used to estimate the increase in description length without re-evaluating the entire data set:

$$\Delta L(D,c,\mathbf{d}) = -\log \prod_{i=1}^{|\mathbf{d}|} f(d_i | d_{i-k}^{i-1}, M_c(d_1^{i-1})).$$

In the above equation, $M_c(d_1^{i-1})$ denotes the current model at position i, constructed from the training examples for class c and the input sequence d_1^{i-1}.

Typically, the description length increase $\Delta L(D,c,\mathbf{d})$ will be larger for longer documents. We therefore use the per-symbol description length increase in the final class selection rule:

$$\begin{aligned} c(\mathbf{d}) &= \arg\min_{c \in C} \frac{1}{|\mathbf{d}|} \Delta L(D,c,\mathbf{d}) \\ &= \arg\min_{c \in C} -\frac{1}{|\mathbf{d}|} \log \prod_{i=1}^{|\mathbf{d}|} f(d_i | d_{i-k}^{i-1}, M_c(d_1^{i-1})). \end{aligned} \tag{4}$$

The additional $1/|\mathbf{d}|$ factor does not affect the classification outcome for any target document, but it does help to produce scores that are comparable across documents of different length. This is crucial when thresholding is used to reach a desirable tradeoff in misclassification rates. Note that the only difference in implementation in comparison to the MCE criterion in Equation 3 is that the model is adapted while evaluating the target. It is clear, however, that Equation 4 no longer amounts to measuring the document cross-entropy $H(X,M_c,\mathbf{d})$ with respect to model M_c, since a different model is used at each position of the sequence $\mathbf{d}$.

Intuitively, the description length increase $\Delta L(D,c,\mathbf{d})$ measures the "*surprise*" at observing d under the hypothesis $c(d) = c$, which is proportional to the (im)probability of d under the model for c. The intuition behind adapting the model M_c is that it *conditionally* continues to learn about c from the target document: If the hypothesis $c(d) = c$ actually holds and an improbable pattern is found in the initial part of d, then the probability that the pattern reoccurs in the remainder of the document should increase. Although we do not know whether the hypothesis $c(d) = c$ is true or not, it is assumed to be true for the purpose of testing its tenability.

Let us conclude this section with an illustrative example as to why the MDL classification criterion might be preferable to the MCE approach. Consider a hypothetical spam filtering problem in which a machine learning researcher uses a compression-based classifier to filter spam from his email. In addition to research-related email, our researcher also receives an abundant amount of spam that advertises prescription drugs. At some point, he receives an email on machine learning methods for drug discovery. This is a legitimate email, but it contains many occurrences of two particular terms that the filter strongly associates with spam: "medicine" and "drugs". In this scenario, the prevalence of these two terms might cause the MCE criterion to label the email as spam, but the MDL criterion would probably consider the email legitimate. This is because while the first occurrence of the terms "medicine" and "drugs" are surprising under the hypothesis "document is legitimate", subsequent occurrences are less surprising. They are, in a sense, redundant. The classifier will learn this as a direct consequence of allowing the model to adapt to the target.

It is interesting to note that Benedetto et al. (2002), who consider the use of the LZ77 compression algorithm (zip) for language identification and authorship attribution, notice that LZ77 adapts to the target text and take measures to prevent this behavior. We, on the other hand, believe this effect is beneficial, which is supported in the results of our experiments.

6. Experimental Setup and Evaluation Methodology

Our primary concern is the use of compression models in spam filtering. This problem differs from classical text categorization tasks in a number of ways:

- The cost of misclassification is highly unbalanced. Although the exact tradeoff will vary in different deployment environments, it tends to be biased toward minimizing false positives (i.e., misclassified legitimate messages).
- Messages in an email stream arrive in chronological order and must be classified upon delivery. It is also common to deploy a filter without *any* training data. Although previous studies typically use cross validation experiments, the appropriateness of cross validation is questionable in this setting.
- Many useful features may be gleaned from various message headers, formats and encodings, punctuation patterns and structural features. It is therefore desirable to use raw, unobfuscated messages with accompanying meta data intact for evaluation.

These unique characteristics of the spam filtering problem are reflected in the design of our experiments and the choice of measures that were used for classifier evaluation. This section gives an overview of the test corpora and evaluation methodology used to obtain our results.

6.1 Online Spam Filter Evaluation

An online learning scheme that lends itself well to typical usage of spam filters was adopted as the primary means of classifier evaluation. In this setup, messages are presented to the classifier in chronological order. For each message, the classifier must produce a score as to how likely it is that the message is spam, after which it is communicated the gold standard judgment. This allows the classifier to update its model before assessing the next message.

The setup aims to simulate a typical setting in personal email filtering, which is usually based on online user feedback, with the additional assumption that the user promptly corrects the classifier after every misclassification. The same evaluation method was used in the large-scale spam filter evaluation at TREC 2005, an overview of which can be found in the TREC proceedings (Cormack and Lynam, 2005).

The performance of different compression models and classification criteria were evaluated using the described scheme. We also compared compression models to a selection of established spam filters in this manner. Standard cross validation experiments on predefined splits were performed to compare compression models to previously published results which were also obtained with cross validation.

6.2 Evaluation Measures

Special care must be taken in the choice of evaluation measures for spam filtering. Classification accuracy, that is, the total proportion of misclassified messages, is a poor performance measure in this application domain, since all errors are treated on equal footing (Androutsopoulos et al., 2000).

In the binary spam filtering problem, spam messages are usually associated with the positive class, since these are the messages filtered by the system. Legitimate messages are thus designated to the negative class. If p is the total number of positive examples in the test set and n is the total number of negative examples, four classification outcomes are defined by the standard binary contingency table. Legitimate message may be incorrectly labeled as spam (fp – false positives) or correctly identified as legitimate (tn – true negatives). Similarly, spam messages may be incorrectly labeled as legitimate (fn – false negatives) or correctly identified as spam (tp – true positives). The false positive rate (FPR) and spam misclassification rate (SMR) are then defined as follows:

$$\text{FPR} = \frac{fp}{n}, \qquad \text{SMR} = \frac{fn}{p}.$$

FPR and SMR measures are intuitive and appealing, however, it is difficult to compare systems based on these measures alone, since one of them can always be improved at the expense of the other.

It is assumed that the scores produced by a learning system are comparable across messages, so that a fixed filtering threshold can be used to balance between spam misclassification and false positive rates. Such scores lend themselves well to Receiver Operating Characteristic (ROC) curve analysis, which was the primary means of classifier evaluation in the study. The ROC curve is a plot of spam accuracy (1 – SMR) on the Y axis, as a function of the false positive rate on the X axis.[5] Each point on the curve corresponds to an actual (FPR, SMR) pair achieved by the classifier at a certain threshold value. The curve thus captures the behavior of the system at all possible filtering thresholds.

A good performance is characterized by a curve that reaches well into the upper left quadrant of the graph. The area under the ROC curve (AUC) is then a meaningful statistic for comparing filters. If we assume that high score values are associated with the positive class, the area under the curve equals the probability that a random positive example receives a higher score than a random negative example:

$$\text{AUC} = P(score(\mathbf{x}) > score(\mathbf{y}) \mid c(\mathbf{x}) = positive,\, c(\mathbf{y}) = negative).$$

Typical spam filters achieve very high values in the AUC statistic. For this reason, we report on the complement of the AUC value, that is, the area *above* the curve (1-AUC). Bootstrap resampling was used to compute confidence intervals for logit-transformed AUC values and to test for significance in paired comparisons. Where suitable, we also report SMR at filtering thresholds that result in “acceptable” false positives rates (0.01%, 0.1% and 1%). This measure is easier to interpret and gives insight in the kind of performance one can expect from a spam filter.

6.3 Data Sets

We report experimental results on five publicly available data sets and a private collection of email compiled by one of the authors. The basic statistics for all six corpora are given in Table 1.

The TREC public[6] corpus contains messages received by employees of the Enron corporation over a one year period. The original Enron data was carefully augmented with the addition of

[5] It is traditional to name the axes of an ROC plot 1-specificity (X axis) and sensitivity (Y axis). Sensitivity is the proportion of correctly identified positive examples and specificity is the proportion of correctly identified negative examples.

[6] The TREC corpus is available for download at `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`.

Data Set	*Messages*	*Spam*	*Legitimate*	*Spam proportion*
TREC public	92189	52790	39399	57.3%
MrX	49086	40048	9038	81.6%
SpamAssassin	6033	1884	4149	31.2%
Ling-Spam	2893	481	2412	16.6%
PU1	1090	480	610	44.0%
PU3	4130	1820	2310	44.1%

Table 1: Basic statistics for the evaluation data sets.

approximately 50,000 spam messages, so that they appear to be delivered to the Enron employees in the same time period as the legitimate email.

The MrX data set contains email messages received by a single email user over a period of 8 months. This data set and the TREC corpus were recently used for the spam filter evaluation track in TREC 2005. Results from the TREC evaluation that are most relevant to our study are reproduced in this paper.

The SpamAssassin[7] data set contains legitimate and spam email collected from the SpamAssassin developer mailing list. This data set is arguably the most widely used resource in popular evaluations of publicly available spam filters, often conducted by enthusiasts or system authors.

Ling-Spam[8] is a collection of email messages posted to a linguistics newsgroup, which were augmented with spam received by the authors of the data set. The messages are stripped of all attachments and headers, except for the subject field. A fair number of research studies report results on the Ling-Spam corpus. We used the “bare” version of this data set in our evaluation.

The PU1 and PU3[9] data sets are relatively small personal email collections. In order to preserve privacy, the words in the messages are replaced with numerical identifiers and punctuation is discarded. Non-textual message headers, sender and recipient fields, attachments and HTML tags are not included in these data sets. Duplicate spam messages received on the same day are also removed.

We used the online evaluation scheme described in the previous subsection to evaluate performance on the TREC public, MrX and SpamAssassin corpora. We performed cross validation experiments on the remaining three data sets, as was done in previous studies. The Ling-Spam, PU1 and PU3 data sets contain predefined 10-fold cross validation splits, which we used in our experiments. These data sets do not contain message headers, so the original chronological order required for online evaluation could not be recovered.

6.4 Implementation and Parameters of DMC and PPM Models

All results reported in the study were achieved using our own implementations of DMC and PPM compression models. Classifiers based on the DMC and PPM compression models were developed independently by the authors and differ in preprocessing strategies and certain implementation details.

[7] The SpamAssassin data set is available at `http://spamassassin.org/publiccorpus/`.

[8] The Ling-Spam data set is available at `http://www.aueb.gr/users/ion/data/`.

[9] The PU1 and PU3 data sets are available for download at `http://www.iit.demokritos.gr/skel/i-config/downloads/PU123ACorpora.tar.gz`.

The DMC model was primed with an initial braid structure (Cormack and Horspool, 1987), corresponding to an order-7 binary Markov model. DMC uses two parameters that control its state cloning mechanism. These parameters were set somewhat arbitrarily to $(2,2)$, since such values were known by the authors to perform well for data compression. The initial transition counts were set to 0.2, following a similar argument. The DMC implementation does not include MIME decoding. It also truncates all messages to 2500 bytes.

The PPM implementation used an order-6 PPM-D model in all trials. Order-4 and order-8 models were also tested in the TREC evaluation, from which it was concluded that performance is robust to the choice of this parameter (Bratko and Filipič, 2005). In data compression, an order-6 model would also be considered suitable for compression of English text. The source alphabet for PPM was restricted to 72 ASCII characters including alphanumerical symbols and commonly used punctuation. This alphabet was complemented with an additional symbol that was used for all other ASCII codes found in the text. Our PPM-based classifier decodes base64-encoded message parts and discards all non-text attachments before evaluation.

The PPM-based classifier used a memory buffer of approximately 800MB, substantially less than the DMC implementation which was limited to 2GB of RAM. Both algorithms used the same retraining strategy when this memory limit was reached in online evaluation experiments. Specifically, half of the training data was discarded and models were retrained from the more recent part of the email stream. This mechanism was invoked up to twice during online evaluation on the two larger data sets (MrX and the TREC public corpus), but was not used in any of the other trials.

We realize that this setup does not facilitate a fair comparison between the two compression algorithms in the online experiments (on raw email data), as the different preprocessing schemes were found to have an effect on performance in some of these experiments. However, the aim of this paper is the evaluation of compression models against existing spam filtering techniques, as well as a comparison of the two classification criteria discussed in Section 5. We are satisfied with the general observation that both algorithms exhibit similar performance, which strengthens our confidence in the applicability of the proposed methods for the spam filtering problem.

6.5 Reference Systems Used for Comparative Evaluation

A number of freely available open source spam filters have been introduced in recent years, motivated mainly by the influential essays of Graham (2004) and Robinson (2003). A wide variety of learning algorithms, training strategies, preprocessing schemes and recipes for feature engineering are employed in these systems. It is interesting to note that most publications that address spam filtering do not compare their proposed methods to these established alternatives.

We evaluate the performance of compression models against six popular open source filters. We also summarize results obtained in other studies that use the Ling-Spam, PU1 and PU3 corpora, and from which it is possible to determine misclassification rates from the published results. Table 2 lists all systems that were included in any of the comparisons.

7. Results

In this section, we report the main results of our evaluation. We first evaluate the performance of the MCE and MDL classification criteria, that is, the effect of adapting the model to the target, for both compression algorithms. This is followed by an extensive evaluation of compression-based classifiers in comparison to established spam filters and results published in other studies. We

Label		**Description**
Bogofilter^a	⋆	Version 0.94.0, default parameters (`http://www.bogofilter.org`).
Bogofilter^b	●	Bogofilter version 0.95.2 as configured for TREC 2005 by the track organizers.
CRM114^a	⋆	Version 20041231, default parameters (`http://crm114.sourceforge.net`).
CRM114^b	●	CRM114 specially configured by Assis et al. (2005) for TREC. Labeled *“CRMSPAM2”* at TREC 2005.
dbacl^a	⋆	Version 1.91, default parameters (`http://dbacl.sourceforge.net`).
dbacl^b	●	A custom version of dbacl prepared by the author for evaluation at TREC (Breyer, 2005). Labeled *“lbSPAM2”* at TREC 2005.
SpamAssassin^a	⋆	Version 3.0.2, combination of rule-based and learning components (`http://spamassassin.apache.org`).
SpamAssassin^b	●	Version 3.0.2, learning component only, as configured for TREC 2005 by the track organizers.
SpamBayes^a	⋆	Version 1.03, default parameters (`http://spambayes.sourceforge.net`).
SpamBayes^b	●	SpamBayes specially configured by Meyer (2005) for TREC. Labeled *“tamSPAM1”* at TREC 2005.
SpamProbe	●⋆	Version 1.0a, default parameters (`http://spamprobe.sourceforge.net`).
a-Bayes	◇	Naive Bayes, multi-variate Bernoulli model with binary features (Androutsopoulos et al., 2000).
a-FlexBayes	◇	Flexible naive Bayes—uses kernel density estimation for estimating class-conditional probabilities of continuous valued attributes (Androutsopoulos et al., 2004).
a-LogitBoost	◇	LogitBoost (variant of boosting) with decision stumps as base classifiers (Androutsopoulos et al., 2004).
a-SVM	◇	Linear kernel support vector machines (Androutsopoulos et al., 2004).
b-Stack	◇	Stacking of linear support vector machine classifiers built from different message fields (Bratko and Filipič, 2006).
c-AdaBoost	◇	Boosting of decision trees with real-valued predictions (Carreras and Márquez, 2001).
gh-Bayes	◇	Naive Bayes (exact model unknown) with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
gh-SVM	◇	Linear support vector machine with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
h-Bayes	◇	Multinomial naive Bayes (Hovold, 2005).
ks-Bayes	◇	Multinomial naive Bayes (Schneider, 2003).
p-Suffix	◇	Pattern matching of character sequences based on the suffix tree data structure and various heuristic scoring functions (Pampapathi et al., 2006).
m-Filtron	◇	Support vector machines with linear kernels (Michelakis et al., 2004).
s-Stack	◇	Stacking of naive Bayes and *k*-nearest neighbors (Sakkis et al., 2001).
s-kNN	◇	*k*-nearest neighbors with attribute and distance weighting (Sakkis et al., 2003).
SVM	⋆	An adaptation of the SVM^{light} package (Joachims, 1998) for the PU1 data set due to Tretyakov (2004), linear kernel with C = 1.
Perceptron	⋆	Implementation of the perceptron algorithm due to Tretyakov (2004).

Table 2: Reference systems and results of previous studies reproduced for comparison. Entries are delimited by primary authors. Symbols indicate the source of reported results:
⋆ – this study   ● – TREC 2005 evaluation   ◇ – reproduced from other studies

conclude the section with experiments that study the effect of noise introduced in data by typical obfuscation tactics employed by spammers to evade filtering. Additional results from our evaluation are available in Online Appendix 1.[10]

7.1 Performance of MCE vs. MDL Classification Criteria

We evaluated the effect of adapting the compression model to the target document on the TREC public, MrX and SpamAssassin data sets. AUC scores achieved by the static and adaptive DMC and PPM models are listed in Table 3. The adaptive models clearly outperform their static counterparts on all data sets, sometimes strikingly so. The area above the ROC curve is more than halved in two of the six experiments and substantially improved in three of the remaining four trials. The improvement is smallest for the DMC model tested on the TREC public data set. As we shall see, even the baseline performance achieved by the static model is exceptionally good in this experiment, and thus hard to improve.

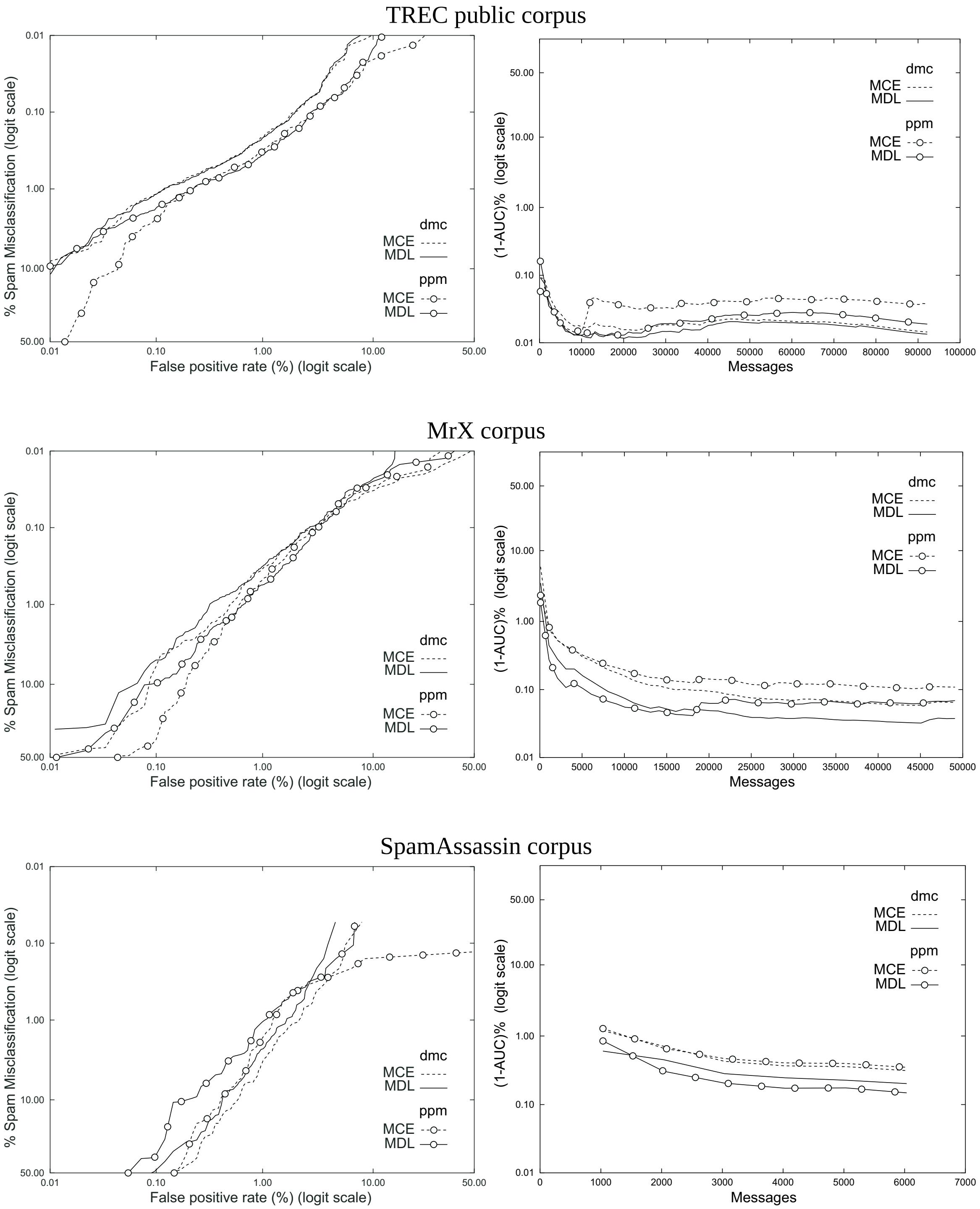
	DMC				**PPM**			
Data Set	MCE		MDL		MCE		MDL	
TREC	0.014	(0.010–0.020)	**0.013**	(0.010–0.018)	0.038	(0.027–0.052)	**0.019**†	(0.015–0.023)
MrX	0.065	(0.040–0.11)	**0.037**†	(0.026–0.053)	0.11	(0.073–0.16)	**0.069**†	(0.044–0.11)
SpmAssn	0.31	(0.21–0.47)	**0.20**†	(0.14–0.30)	0.35	(0.20–0.60)	**0.15**†	(0.086–0.26)

Table 3: Performance of DMC and PPM algorithms in combination with the MCE and MDL classification criteria on the TREC public, MrX and SpamAssassin data sets. Results are in the area above the ROC curve 1-AUC(%) statistic and include 95% confidence intervals for this measure. The best results for each algorithm/data set pair are in bold. Statistically significant differences are marked with a '†' sign ($p < 0.01$, one-tailed).

The ROC curves and 1-AUC learning curves of adaptive and static DMC and PPM models are depicted in Figure 2. Let us first comment on the ROC curves, which reveal an interesting and remarkably consistent pattern. Note that the ROC graphs are plotted in logarithmic scale for clarity, so a non-convex ROC area is not necessarily unexpected. Although adaptive models dominate throughout the curve in most experiments, the gain in the 1-AUC statistic can mostly be attributed to the fact that the adaptive models perform better at the extreme ends of the curves. Performance is comparable when FPR and SMR are balanced. The logarithmic scale should again be taken into consideration when examining the magnitude of this effect. This suggests that the adaptive model makes less gross mistakes, which are costly in terms of the AUC measure. Performance at the extreme ends of the curve is especially important when the cost of misclassification is unbalanced, so this is certainly a desirable property for spam filtering.

The learning curves in Figure 2 depict accumulated 1-AUC scores sampled at 1000 message intervals during the online learning experiments. They are again plotted in logarithmic scale to facilitate evaluation of the asymptotic performance of classifiers. The main observation offered by the learning curves is that the adaptive models do not achieve better overall performance at the price of slower learning rates. Their performance is superior throughout the runs. The difference in 1-AUC scores is in fact greater in the earlier stages of learning for two of the three data sets. This is intuitive, since the effect of adapting the models will be greater for simpler models built from

[10] Available at `http://ai.ijs.si/andrej/papers/jmlr2006/`.



TREC public corpus

MrX corpus

SpamAssassin corpus

Figure 2: ROC curves (left) and 1-AUC learning curves (right) on the MrX, SpamAssassin and TREC public corpora. MCE and MDL classification criteria estimated with the DMC and PPM algorithms are compared.

limited training data. In Online Appendix 2,[11] we address an apparent anomaly which occurs in the learning curve of the static version of the PPM classifier at around 12,000 messages on the TREC public corpus. The analysis presented in the appendix exposes the differences between the MCE and MDL classification criteria, but is beyond the scope of the current discussion.

7.2 Comparison to Open Source Spam Filters

The results of our experimental comparison of compression models and established open source filters are summarized in Table 4. Adaptive versions of the PPM and DMC classifiers were used for this comparison. In terms of the 1-AUC score, both compression models uniformly outperformed all of the competing filters, with the exception of the PPM classifier on the MrX corpus. The performance of DMC and PPM models on the TREC public corpus is particularly notable. Spam misclassification rates at hypothetical filtering thresholds that result in a low proportion of false positives are also shown. Although definitive conclusions are harder to draw from these measures, we find that both DMC and PPM feature prominently, outperforming other methods in two of the tree tradeoff points on every data set. ROC curves and learning curves of compression models and open source filters on the TREC public corpus are shown in Figure 3.

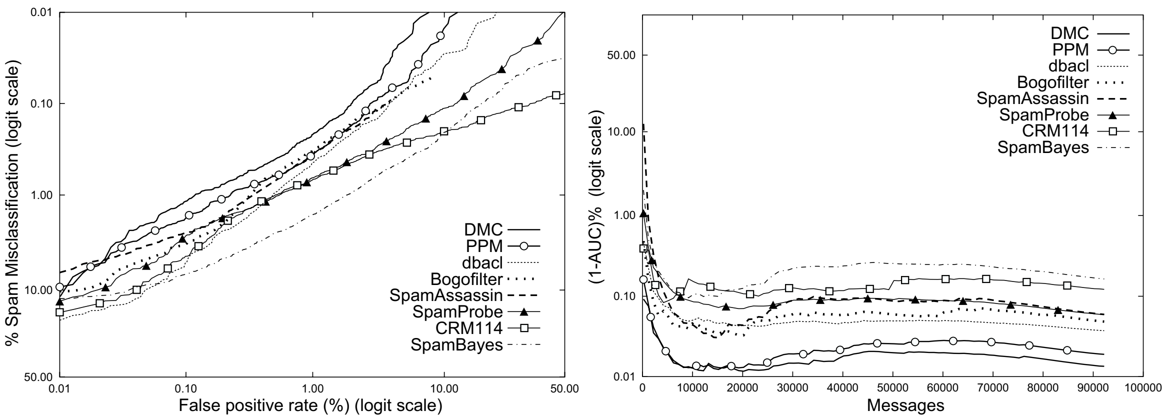


Figure 3: ROC curves (left) and 1-AUC learning curves (right) for compression models and a selection of established spam filters on the TREC public corpus.

7.3 Comparison with Published Results

We conducted standard cross validation experiments to evaluate classification performance of PPM and DMC (adaptive versions) on the Ling-Spam, PU1 and PU3 data sets. Implementations of the perceptron and SVM classifiers, as well as Bogofilter, a well-performing open source filter from previous experiments, were also tested in this manner. Results of these experiments are presented in Figures 4 and 5, in which we also reproduce results of previous studies on the same data. The simplified ROC-style graphs plot the number of misclassified spam messages against the number of false positives.

[11] Available at `http://ai.ijs.si/andrej/papers/jmlr2006/`.

TREC public corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
DMC	†**0.013** (0.010 – 0.018)	0.22%	1.17%	14.47%
PPM	†**0.019** (0.015 – 0.023)	0.36%	1.78%	9.89%
dbacl[b]	**0.037** (0.031 – 0.045)	0.45%	5.19%	19.77%
Bogofilter[b]	**0.048** (0.038 – 0.062)	0.33%	3.41%	10.39%
SpamAssassin[b]	**0.059** (0.044 – 0.081)	0.37%	2.56%	7.81%
SpamProbe	**0.059** (0.049 – 0.071)	0.65%	2.77%	15.30%
CRM114[b]	**0.122** (0.102 – 0.145)	0.68%	4.52%	17.17%
SpamBayes[b]	**0.164** (0.142 – 0.189)	1.63%	6.92%	12.55%

MrX corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
DMC	†**0.037** (0.026 – 0.053)	0.32%	5.08%	36.16%
Bogofilter[b]	**0.045** (0.032 – 0.063)	0.57%	3.90%	31.04%
CRM114[b]	**0.051** (0.035 – 0.075)	0.43%	9.65%	47.76%
PPM	**0.069** (0.044 – 0.107)	0.56%	9.72%	94.42%
dbacl[b]	**0.083** (0.054 – 0.130)	0.43%	10.24%	99.09%
SpamAssassin[b]	**0.097** (0.070 – 0.135)	0.77%	6.19%	83.06%
SpamProbe	**0.097** (0.063 – 0.150)	0.35%	15.54%	95.08%
SpamBayes[b]	**0.138** (0.111 – 0.171)	1.10%	6.51%	45.65%

SpamAssassin corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
PPM	**0.148** (0.086 – 0.256)	1.06%	38.67%	66.42%
DMC	**0.202** (0.136 – 0.301)	2.39%	48.86%	64.93%
Bogofilter[a]	**0.209** (0.140 – 0.312)	3.08%	57.67%	99.10%
SpamAssassin[a]	**0.254** (0.173 – 0.373)	4.93%	24.77%	100.00%
dbacl[a]	**0.262** (0.169 – 0.404)	2.65%	58.36%	79.26%
SpamProbe	**0.296** (0.195 – 0.450)	2.18%	74.43%	99.47%
CRM114[a]	**1.143** (0.902 – 1.446)	6.26%	57.82%	83.02%
SpamBayes[a]	**1.391** (1.036 – 1.867)	11.35%	92.73%	99.26%

Table 4: Performance of DMC, PPM and a selection of established spam filters on the TREC public, MrX and SpamAssassin data sets. Filters are ordered by decreasing performance in the 1-AUC statistic. Significant differences between AUC scores achieved by the compression models and the best competing filter are marked with a '†' sign ($p < 0.05$, one-tailed).

On the Ling-Spam data set, both compression models are comparable to the suffix tree approach of Pampapathi et al. (2006). These three classifiers dominate the other methods at all filtering thresholds. Both the suffix tree classifier and the compression models considered in this paper operate on character-level or binary sequences. All other methods use the standard bag-of-words representation for modeling text. This suggests that methods based on sub-word symbol sequences are more suitable for the Ling-Spam data set, and we believe this to be the case for spam filtering in general. However, experimental results on the PU1 and PU3 corpora show that character or binary-level modeling of text is not the only advantage offered by the compression models.

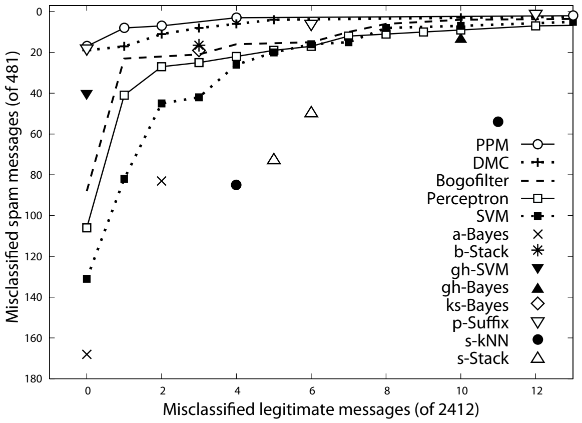


Figure 4: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and previously published results on the Ling-Spam data set.

The PU1 and PU3 data sets contain pre-tokenized messages in which the original words are replaced with numeric identifiers. We converted these messages to binary format by replacing token identifiers with their 16 bit binary equivalents. This representation was used to evaluate the performance of the DMC classifier on the two data sets. We believe this is fair, since DMC uses a binary alphabet and is hurt by the artificial tokenization of these data sets. The PPM classifier was tested on the unprocessed original version of the data sets. Digits in numeric identifiers were converted to alphabetical characters for testing with Bogofilter, since the filter handles digits differently from alphabetical character strings.

The graphs in Figure 5 show classification performance on the PU1 and PU3 data sets and are perhaps the most surprising result reported in this paper. Both compression models outperform other classifiers almost uniformly across the range of ‘interesting’ filtering thresholds, despite the fact that the data sets were produced for tokenization-based filters. The SVM is competitive on the PU1 data set. Other methods are further behind the SVM in both trials. We attribute the good performance of compression models in these tests to the fact that tokens are *not* considered independently. Their probability is always evaluated with respect to the local context, which was already found to be beneficial for word-level models in previous studies (Peng et al., 2004). Compression models offer the additional advantage over language models considered by Peng et al. (2004) in their effective strategy for adapting the model structure incrementally. By design, the compression models can discover phrase-level patterns just as naturally as sub-word patterns.

7.4 Sensitivity to Noise in the Data

One of the perceived advantages of statistical data compression models over standard text categorization algorithms is that they do not require any preprocessing of the data. Classification is not based on word tokens or manually constructed features, which is in itself appealing from the implementation standpoint. For spam filtering, this property is especially welcome, since preprocessing steps are error-prone and are often exploited by spammers in order to evade filtering. A typical strategy is to distort words with common spelling mistakes or character substitutions, which may

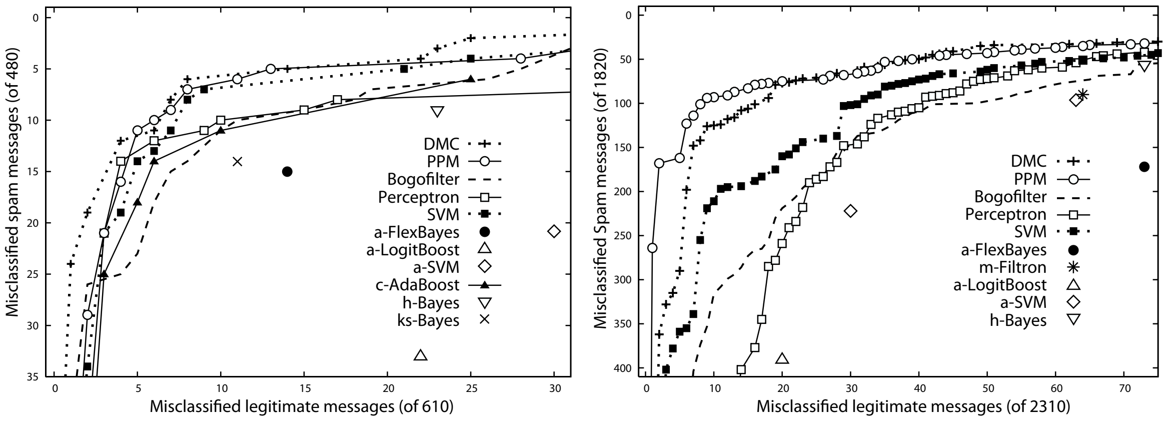


Figure 5: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and previously published results on the PU1 (left) and PU3 (right) data sets.

confuse an automatic filter. We believe that compression models are much more robust to such tactics than methods which require tokenization.

To support this claim, we conducted an experiment in which all messages in the SpamAssassin data set were distorted by substituting characters in the original text with random alphanumeric characters and punctuation. The characters in the original message that were subject to such a substitution were also chosen randomly. By varying the probability of distorting each character, we evaluated the effect of such noise on classification performance.

For this experiment, all messages were first decoded and stripped of non-textual attachments. Noise was added to the subject and body parts of messages. Adaptive compression models and Bogofilter, a representative tokenization-based filter, were evaluated on the resulting data set. We then stripped the messages of all headers except subjects and repeated the evaluation. The results of these experiments are depicted in Figure 6. They show that compression models are indeed very robust to noise. Even after 20% of all characters are distorted, rendering messages practically illegible, they retain a respectable performance. The advantage of compression models on noisy data is particularly pronounced in the second experiment, where the classifier must rely solely on the (distorted) textual contents of the messages. It is interesting to note that the performance of PPM increases slightly at the 5% noise level if headers are kept intact. We have no definitive explanation for this phenomenon. We suspect that introducing noise in the text implicitly increases the influence of the non-textual message headers, and that this effect is beneficial.

8. Conclusion

In comparison to tokenization-based classification methods, compression models offer a number of advantages that are especially relevant for spam filtering. By operating directly on sequences, tokenization, stemming and other tedious and error-prone preprocessing steps are omitted altogether. It is precisely these volatile preprocessing steps that are often exploited by spammers in order to evade filtering. Also, characteristic sequences of punctuation and other special characters, which are generally thought to be useful in spam filtering, are naturally included in the model. The algo-

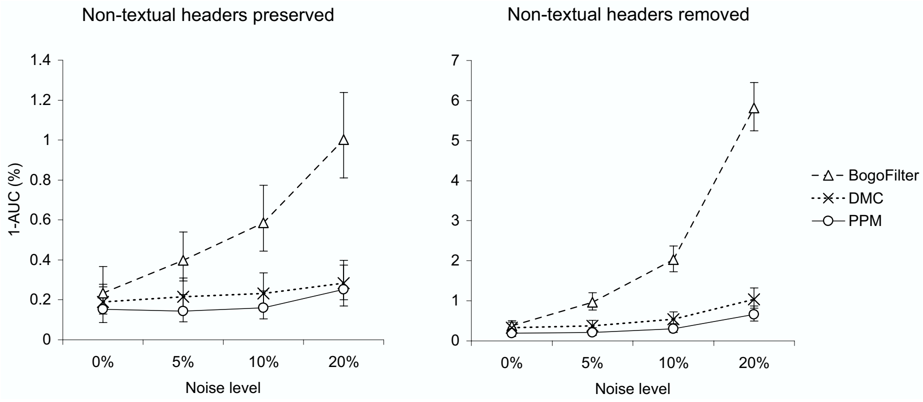


Figure 6: Effect of noise on classification performance on the SpamAssassin data set. The graph shows 1-AUC scores with 95% confidence intervals for PPM, DMC and the tokenization-based Bogofilter system at different levels of artificial random noise in the data.

rithms are efficient, with training and classification times linear in the amount of data. The models are incrementally updateable, which is often a requirement for practical spam filters that support online learning based on user feedback.

Empirically, we demonstrate that compression models perform very well for spam filtering, consistently outperforming established spam filters and other methods proposed in previous studies. We also show that compression models are very robust to the type of noise introduced in the text by typical obfuscation tactics used by spammers. This should make them difficult for spammers to defeat, but also makes them attractive for other text categorization problems that contain noisy data, such as classification of text extracted with optical character recognition.

Finally, we find that updating compression models adaptively to the target document is beneficial for classification, particularly in improving the AUC measure. This is especially desirable when the cost of misclassification is uneven, as is the case in spam filtering. The modification has a natural interpretation in terms of the minimum description length principle. Although we are aware of no parallel to this in existing text classification research, the same approach could easily be adopted for the popular multinomial naive Bayes model (McCallum and Nigam, 1998) and possibly also for other incremental models. We believe this to be an interesting avenue for future research.

The large memory requirements of compression models are a major disadvantage of this approach. To this end, effective pruning strategies should be investigated in order to bring the models within limits that would be suitable for practical applications. Should compression models actually be employed in practice, the adversarial nature of spam filtering suggests spammers will react to these techniques. It remains to be seen whether their efforts could reduce the long-term efficacy of the proposed approach.

References

P. H. Algoet and T. M. Cover. A sandwich proof of the Shannon-McMillan-Breiman theorem. *Annals of Probability*, 16:899–909, 1988.

I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proc. of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, 2000.

I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. Technical Report 2004/2, NCSR “Demokritos”, October 2004.

F. Assis, W. Yerazunis, C. Siefkes, and S. Chhabra. CRM114 versus Mr. X: CRM114 notes for the TREC 2005 spam track. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

A. R. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

D. Benedetto, E. Caglioti, and Loreto V. Language trees and zipping. *Physical Review Letters*, 88 (4), 2002.

A. Bratko and B. Filipič. Spam filtering using character-level markov models: Experiments for the TREC 2005 Spam Track. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

A. Bratko and B. Filipič. Exploiting structural information for semi-structured document categorization. *Information Processing & Management*, 42(3):679–694, 2006.

L. A. Breyer. DBACL at the TREC 2005. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.

X. Carreras and L. Márquez. Boosting trees for anti-spam email filtering. In *Proc. of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001.

J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40 (2/3):67–75, 1997.

J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

G. V. Cormack and R. N. S. Horspool. Data compression using dynamic Markov modelling. *The Computer Journal*, 30(6):541–550, 1987.

G. V. Cormack and T. R. Lynam. TREC 2005 Spam Track overview. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209, Snowbird, US, 2000. IEEE Computer Society Press, Los Alamitos, US.

J. Goodman, D. Heckerman, and R. Rounthwaite. Stopping spam. *Scientific American*, 292(4): 42–88, April 2005.

P. Graham. *Hackers and Painters, Big Ideas from the Computer Age*, chapter 8, pages 121–130. O'Reilly, 2004.

P. Grünwald. A tutorial introduction to the minimum description length principle. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 3–81. MIT Press, 2005.

J. M. G. Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 615–620, Madrid, March 2002. ACM Press.

J. Hovold. Naive bayes spam filtering using word-position-based attributes. In *Proc. of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, Palo Alto, CA, July 2005.

P. G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems.* PhD thesis, Brown University, Providence, Rhode Island, 1993.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, 1998.

E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD '04: Proc. of the 10th ACM SIGKDD conference on Knowledge Discovery and Data mining*, pages 206–215, Seattle, WA, 2004. ACM Press.

P. Kontkanen, P. Myllymki, W. Buntine, J. Rissanen, and H. Tirri. An MDL framework for data clustering. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 863–872, 2003.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

T. A. Meyer. A TREC along the spam track with SpamBayes. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, July 2004.

R. Pampapathi, B. Mirkin, and M. Levene. A suffix tree approach to anti-spam email filtering. *Machine Learning*, 65(1):309–338, 2006.

F. Peng, D. Schuurmans, and S. Wang. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345, 2004.

I. Rigoutsos and T. Huynh. Chung-kwei: A pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, July 2004.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, 1984.

J. Rissanen. Complexity of strings in the class of Markov sources. *IEEE Transactions on Information Theory*, 32(4):526–532, 1986.

J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, 1996.

G. Robinson. A statistical approach to the spam problem. *Linux Journal*, 107:3, March 2003.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proc. 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 44–50, Pittsburgh, PA, 2001.

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.

K. M. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proc. of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

D. Sculley and C. E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Proc. of the 2006 Data Compression Conference (DCC 2006)*, pages 332–332, Snowbird, UT, 2006. IEEE Computer Society.

W. J. Teahan. Text classification and segmentation using minimum cross-entropy. In *Proceeding of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, Paris, 2000.

W. J. Teahan and D. J. Harper. Using compression-based language models for text categorization. In W. B. Croft and J. Lafferty, editors, *Language Modeling for Information Retrieval*, pages 141–166. Kluwer Academic Publishers, 2003.

K. Tretyakov. Machine learning techniques in spam filtering. Technical report, Institute of Computer Science, University of Tartu, 2004.

F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.