Discretizing Continuous Attributes in AdaBoost for Text Categorization

Pio Nardiello[1], Fabrizio Sebastiani[2], and Alessandro Sperduti[3]

[1] MercurioWeb SNC
Via Appia
85054 Muro Lucano (PZ), Italy
pionardiello@mercurioweb.net

[2] Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
fabrizio@iei.pi.cnr.it

[3] Dipartimento di Matematica Pura ed Applicata
Università di Padova
35131 Padova, Italy
sperduti@math.unipd.it

Abstract. We focus on two recently proposed algorithms in the family of "boosting"-based learners for automated text classification, AdaBoost.MH and AdaBoost.MH^{KR}. While the former is a realization of the well-known AdaBoost algorithm specifically aimed at multi-label text categorization, the latter is a generalization of the former based on the idea of learning a committee of classifier sub-committees. Both algorithms have been among the best performers in text categorization experiments so far.

A problem in the use of both algorithms is that they require documents to be represented by binary vectors, indicating presence or absence of the terms in the document. As a consequence, these algorithms cannot take full advantage of the "weighted" representations (consisting of vectors of continuous attributes) that are customary in information retrieval tasks, and that provide a much more significant rendition of the document's content than binary representations.

In this paper we address the problem of exploiting the potential of weighted representations in the context of AdaBoost-like algorithms by discretizing the continuous attributes through the application of entropy-based discretization methods. We present experimental results on the Reuters-21578 text categorization collection, showing that for both algorithms the version with discretized continuous attributes outperforms the version with traditional binary representations.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; I.2.3 [**Artificial Intelligence**]: Learning—*Induction*

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Machine learning, text categorization, text classification, discretization, continuous attributes, boosting

1 Introduction

In the last ten years an impressive array of learning techniques have been used in text categorization (TC) research, including probabilistic methods, regression methods, decision tree and decision rule learners, neural networks, batch and incremental learners of linear classifiers, example-based methods, genetic algorithms, hidden Markov models, support vector machines, and classifier committees (see [16] for a review). Among these, the two classes of methods that most seem to have caught the attention of TC researchers are boosting (a subclass of the classifier committees class) and support vector machines. The reasons for this attention are twofold, in the sense that both classes exhibit strong justifications in terms of computational learning theory and superior effectiveness once tested on TC benchmarks of realistic size and difficulty. It is on the former class of methods that this paper focuses.

Classifier *committees* (aka *ensembles*) are based on the idea that, given a task that requires expert knowledge to perform, k experts may be better than one if their individual judgments are appropriately combined. In TC, this means applying k different classifiers $\Phi_1, \ldots, \Phi_k$ to the same task of deciding whether a document d_j belongs or not to category c_i, and then combining their outcome appropriately. Boosting is a method for generating a highly accurate classifier (also called *final hypothesis*) by combining a set of moderately accurate classifiers (also called *weak hypotheses*). In this paper we will make use of two algorithms, called AdaBoost.MH [15] and AdaBoost.MH^{KR} [17], which are based on the notion of *adaptive boosting*, a version of boosting in which members of the committee can be sequentially generated after learning from the classification mistakes of previously generated members of the same committee. AdaBoost.MH [15] is a realization of the well-known AdaBoost algorithm, which is specifically aimed at multi-label TC[4], and which uses *decision stumps* (i.e. decisions trees composed of a root and two leaves only) as weak hypotheses. AdaBoost.MH^{KR} [17] is instead a generalization of AdaBoost.MH based on the idea of learning a committee of classifier sub-committees; in other words, AdaBoost.MH^{KR} weak hypotheses are themselves committees of decision stumps. So far, both algorithms have been among the best performers in text categorization experiments run on standard benchmarks.

A problem in the use of both algorithms is that they require documents to be represented by binary vectors, indicating presence or absence of the terms in the document. As a consequence, these algorithms cannot take full advantage of the "weighted" representations (consisting of vectors of continuous attributes) that are customary in information retrieval tasks, and that provide a much more significant rendition of the document's content than binary representations.

In this paper we address the problem of exploiting the potential of weighted representations in the context of AdaBoost-like algorithms by discretizing the continuous attributes through the application of entropy-based discretization methods. These algorithms attempt to *optimally* split the interval on which

[4] Given a set of categories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$, *multilabel text categorization* is the task in which any number $0 \leq n_j \leq |\mathcal{C}|$ of categories may be assigned to each $d_j \in \mathcal{D}$.

these attributes range into a sequence of disjoint subintervals. This split engenders a new vector (binary) representation for documents, in which a binary term indicates that the original non-binary weight belongs or does not belong to a given sub-interval. We present experimental results on the Reuters-21578 text categorization collection, showing that for both algorithms the version with discretized continuous attributes outperforms the version with traditional binary representations.

The paper is organized as follows. In Section 2 we briefly introduce AdaBoost.MH and AdaBoost.MH^{KR}, while in Section 3 we describe the modified, improved version of AdaBoost.MH^{KR} that we have used in all the experiments described in this paper. In Section 4 we introduce the issue of discretizing continuous attributes, and we propose two discretization algorithms based on information-theoretic intuitions. In Section 5 we describe the experimental results we have obtained by applying both discretization algorithms to both AdaBoost.MH and AdaBoost.MH^{KR}; the benchmark used is Reuters-21578, the standard benchmark of text categorization research. In Section 6 we give our concluding remarks.

2 Boosting Algorithms for Text Categorization

In the following we briefly recall AdaBoost.MH and AdaBoost.MH^{KR}, two boosting algorithms that have been specially developed for text categorization applications[5]. For more details on both algorithms, see [17].

2.1 AdaBoost.MH

AdaBoost.MH is a boosting algorithm proposed by Schapire and Singer [15] for multilabel text categorization applications and derived from AdaBoost, Freund and Schapire's general purpose boosting algorithm [3]. The input to the algorithm is a training set $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_{|Tr|}, C_{|Tr|}\rangle\}$, where $C_j \subseteq C$ is the set of categories to each of which d_j belongs.

AdaBoost.MH works by iteratively calling a *weak learner* to generate a sequence $\Phi_1, \ldots, \Phi_S$ of weak hypotheses; at the end of the iteration the final hypothesis Φ is obtained by a summation

$$\Phi(d_j, c_i) = \sum_{s=1}^{S} \Phi_s(d_j, c_i) \tag{1}$$

of these weak hypotheses. A weak hypothesis is a function $\Phi_s : \mathcal{D} \times C \rightarrow \mathbb{R}$, where $\mathcal{D}$ is the set of all possible documents. We interpret the sign of $\Phi_s(d_j, c_i)$

[5] In this paper we concentrate on AdaBoost.MH *with real-valued predictions*, one of three variants of AdaBoost.MH discussed in [15], since it is the one that, in the experiments of [15], has been experimented most thoroughly and has given the best results, and since it is the one used in the AdaBoost.MH^{KR} system of [17]. The methods that we discuss in Section 4 straightforwardly apply also to the other two variants.

as the decision of Φ_s on whether d_j belongs to c_i (i.e. $\Phi_s(d_j, c_i) > 0$ means that d_j is believed to belong to c_i while $\Phi_s(d_j, c_i) < 0$ means it is believed not to belong to c_i), and the absolute value of $\Phi_s(d_j, c_i)$ (indicated by $|\Phi_s(d_j, c_i)|$) as the strength of this belief.

At each iteration s AdaBoost.MH applies the newly generated weak hypothesis Φ_s to the training set and uses the results to update a distribution D_s of weights on the training pairs $\langle d_j, c_i\rangle$. The weight $D_{s+1}(d_j, c_i)$ is meant to capture how effective $\Phi_1, \ldots, \Phi_s$ were in correctly deciding whether the training document d_j belongs to category c_i or not. By passing (together with the training set Tr) this distribution to the weak learner, AdaBoost.MH forces this latter to generate a new weak hypothesis Φ_{s+1} that concentrates on the pairs with the highest weight, i.e. those that had proven harder to classify for the previous weak hypotheses.

The initial distribution D_1 is uniform. At each iteration s all the weights $D_s(d_j, c_i)$ are updated to $D_{s+1}(d_j, c_i)$ according to the rule

$$D_{s+1}(d_j, c_i) = \frac{D_s(d_j, c_i) \exp(-C_j[c_i] \cdot \Phi_s(d_j, c_i))}{Z_s} \tag{2}$$

where $C_j[c_i]$ is defined to be 1 if $c_i \in C_j$ and -1 otherwise, and

$$Z_s = \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|Tr|} D_s(d_j, c_i) \exp(-C_j[c_i] \cdot \Phi_s(d_j, c_i)) \tag{3}$$

is a normalization factor.

Each document d_j is represented by a vector $\langle w_{1j}, \ldots, w_{|T|j}\rangle$ of $|T|$ binary weights, where $T = \{t_1, \ldots, t_{|T|}\}$ is the set of terms. The weak hypotheses AdaBoost.MH uses are *real-valued decision stumps*, i.e. functions of the form

$$\Phi_s(d_j, c_i) = \begin{cases} a_{0i} \text{ if } w_{kj} = 0 \\ a_{1i} \text{ if } w_{kj} = 1 \end{cases} \tag{4}$$

where $t_k \in T$, a_{0i} and a_{1i} are real-valued constants. The choices for t_k, a_{0i} and a_{1i} are in general different for each iteration, and are made according to a policy that attempts to minimize Z_s, since this is known to be an error-minimization policy (although not an optimal one) [14].

AdaBoost.MH^R chooses weak hypotheses of the form described in Equation 4 by a two step-process:

1. For each term $t_k \in T$ it pre-selects, among all weak hypotheses that have t_k as the "pivot term", the one (indicated by Φ^k_{best}) for which Z_s is minimum. From a computational point of view this is the easier step, since Schapire and Singer [14] provide the provably optimal values for Φ^k_{best} as a function of $D_t(d_j, c_i)$, w_{kj} and $C_j[c_i]$.
2. Among all the hypotheses $\Phi^1_{best}, \ldots, \Phi^{|T|}_{best}$ pre-selected for the $|T|$ different terms, it selects the one (indicated by Φ_s) for which Z_s is minimum. From a computational point of view this is the harder step, since this is $O(|T|)$, which in text categorization applications is typically in the tens of thousands.

2.2 AdaBoost.MH^{KR}

In [17] a variant of AdaBoost.MH^R, called AdaBoost.MH^{KR} (for AdaBoost.MH *with K-fold real-valued predictions*), has been proposed. This algorithm differs from AdaBoost.MH^R in the policy according to which weak hypotheses are chosen, since it is based on the construction, at each iteration s of the boosting process, of a *complex weak hypothesis* (CWH) consisting of a sub-committee of *simple weak hypotheses* (SWHs) $\Phi_s^1, \ldots, \Phi_s^{K(s)}$, each of which has the form described in Equation 4. These are generated by means of the same process described in Section 2.1, but for the fact that at iteration s, instead of selecting and using only the best term t_k (i.e. the one which brings about the smallest Z_s), it selects the best $K(s)$ terms and use them in order to generate $K(s)$ SWHs $\Phi_s^1, \ldots, \Phi_s^{K(s)}$. The CWH is then produced by grouping $\Phi_s^1, \ldots, \Phi_s^{K(s)}$ into a sub-committee

$$\Phi_s(d_j, c_i) = \frac{1}{K(s)} \sum_{q=1}^{K(s)} \Phi_s^q(d_j, c_i) \tag{5}$$

that uses the simple arithmetic mean as the combination rule. For updating the distribution it still applies Equations 2 and 3, where Φ_s is now defined by Equation 5. The final hypothesis is computed by plugging Equation 5 into Equation 1, thus obtaining

$$\Phi(d_j, c_i) = \sum_{s=1}^{S} \alpha_s \frac{1}{K(s)} \sum_{q=1}^{K(s)} \Phi_s^q(d_j, c_i) \tag{6}$$

The number $K(s)$ of SWHs to include in the sub-committee is obtained using a simple heuristics which adds a constant C to K every N iterations, using a fixed value for N and using a value of 1 for $K(1)$, i.e. $K(s) = 1 + C\lfloor \frac{s-1}{N} \rfloor$.

3 An improved version of AdaBoost.MH^{KR}

In this work we have implemented and experimented a new version of AdaBoost.MH^{KR} in which each SWH in a sub-committee is weighted by a value which is inversely proportional to its score. This means replacing Equation 5 with

$$\Phi_s(d_j, c_i) = \frac{1}{K(s)} \sum_{q=1}^{K(s)} \frac{Z_q^{-1}}{\sum_{j=1}^{K(s)} Z_j^{-1}} \cdot \Phi_s^q(d_j, c_i) \tag{7}$$

The rationale of this choice is that the weight associated to a SWH that contributes more to maximizing effectiveness should be higher. This is especially important in the first iterations since, as noted in [17], it is here that the variance of the Z_s values of members of the same sub-committee is higher.

In experiments that we do not report for reasons of space we have observed that this new version consistently improves the performance of the basic version of AdaBoost.MH^{KR} of more than 1%. From now on when referring to AdaBoost.MH^{KR} we thus mean this improved version.

4 Term Discretization

The aim of this section is to define a *discretization* procedure that allows to exploit the rich information conveyed by the non-binary term weights produced by traditional IR weighting techniques (such as $tf * idf$ or others) while at the same time allowing the use of algorithms derived by the original BoosTexter algorithm [15](such as AdaBoost.MH and AdaBoost.MH^{KR}) that requires discrete (in this case: binary) input. An alternative way to directly exploit non-binary term weights would have been to use the traditional AdaBoost algorithm in conjunction with weak hypotheses able to deal with continuous features. We preferred to avoid this more direct solution since it would have been computationally more onerous.

In machine learning, the basic idea that underlies a discretization procedure is that of segmenting the continuous interval $[\alpha, \beta]$ on which an attribute (in our case: a term) t_k ranges on, into an ordered sequence $I = \langle[\alpha = \gamma_0, \gamma_1], (\gamma_1, \gamma_2], \ldots, (\gamma_{|I|-1}, \gamma_{|I|} = \beta]\rangle$ of disjoint sub-intervals such that

- in the vector representation term t_k is replaced by $|I|$ different terms $\{t_{k1}, t_{k2}, \ldots, t_{k|I|}\}$, where w_{krj} (the weight that term t_{kr} has in document d_j) is computed as

$$w_{krj} = \begin{cases} 1 \text{ if } w_{kj} \in (\gamma_{r-1}, \gamma_r] \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

- among all the possible replacements, the one chosen maximizes effectiveness.

Different techniques for discretizing a continuous attribute have been proposed in the field of machine learning. These include 1R [5], ChiMerge and Chi2 [6], plus several entropy-based algorithms [1, 2, 7, 10–12]. In this paper, we adopt a very simple discretization algorithm based on *(multiclass) information gain*, which is defined in terms of the well-known *(multiclass) entropy* measure [9]; for reasons discussed later in this section, we take $|I|$ to be equal to 2. This approach is based on the following rationale. Given a term t_k which can take values from a continuous interval $[\alpha, \beta]$, we look for a value γ^k (which we call the *split value*, or simply *split*) that partitions $[\alpha, \beta]$ into two sub-intervals $[\alpha, \gamma^k]$ and $(\gamma^k, \beta]$ such that the dichotomy induced on the training set (i.e. the dichotomy between the examples for which $t_k \in [\alpha, \gamma^k]$ and the examples for which $t_k \in (\gamma^k, \beta]$) generates two training subsets which are expected to be globally easier to classify (and thus lead to better classifiers) by just using the membership of t_k in the two subintervals as discriminant. Specifically, information gain is a measure that quantifies how easier it is to classify the examples when using the split γ^k (i.e. how well the two newly generated terms separate the positive from the negative examples), and selecting the split that maximizes information gain thus means selecting the split that maximizes separability.

4.1 Algorithm A

Let us fix some notation: if t_k is the term under consideration,

- Let $\alpha > 0$ and β be the upper and lower bounds of the range of values (except 0) that all the terms in T can take (which we assume to be equal for all terms), and let γ^k be a split.
- Let Tr_{in} be the set of training examples d_j for which $w_{kj} \in [\alpha, \beta]$.
- Let Tr_{in}^c be the set of training examples belonging to category c for which $w_{kj} \in [\alpha, \beta]$.
- Let $Tr_{in1}(\gamma^k)$ and $Tr_{in2}(\gamma^k)$ be the sets of training examples d_j for which $w_{kj} \in [\alpha, \gamma^k]$ and $w_{kj} \in (\gamma^k, \beta]$, respectively.

Definition 1. *The* (multiclass) entropy *of Tr with respect to an interval $[\alpha, \beta]$ is defined as*

$$H(Tr, \alpha, \beta) = -\sum_{c \in \mathcal{C}} \frac{|Tr_{in}^c|}{|Tr_{in}|} log_2 \frac{|Tr_{in}^c|}{|Tr_{in}|} \tag{9}$$

The (multiclass) information gain *of a value with respect to Tr is defined as*

$$\begin{aligned} IG(Tr, \alpha, \beta, \gamma^k) = H(Tr, \alpha, \beta) - \\ \frac{|Tr_{in1}(\gamma^k)|}{|Tr|} H(Tr, \alpha, \gamma^k) - \frac{|Tr_{in2}(\gamma^k)|}{|Tr|} H(Tr, \gamma^k, \beta) \end{aligned} \tag{10}$$

□

The basic idea of our discretization process is to find, for each term t_k and among all possible splits γ^k, the split $\hat{\gamma}^k$ that maximizes the information gain. On the basis of this split, a term t_k is replaced by two new terms t_{k0} and t_{k1}, and the weights they have in the training documents are computed according to Equation 8. The fact that these weights are binary will allow us to work with both AdaBoost.MH and AdaBoost.MH^{KR} while at the same time exploiting the rich information present in the non-binary weights of the term t_k which has originated the (binary-weighted) terms t_{k0} and t_{k1}. It should be stressed that, because of $\alpha > 0$, if term t_k is not present in a document, both t_{k0} and t_{k1} will assume zero value.

In general, the application of this discretization strategy to all the terms will double the number of terms. This may not be advisable if the ratio $\frac{|Tr|}{|T|}$ between the number $|Tr|$ of training documents and the number $|T|$ of terms used to represent the documents is not high enough. In fact, it is well known that a low such ratio will in general give rise to overfitting, since in the classifier each term corresponds to a free parameter of the learning problem, and this parameter is set by the training algorithm based on the training examples available, which correspond to constraints on the problem. Therefore, if the $\frac{|Tr|}{|T|}$ ratio is low the learning problem is underconstrained, and the probability that an overspecialized classifier is learnt is high. This is the reason why we have limited the number of intervals into which a term is discretized to two.

For pretty much the same reason, we also introduce a selection procedure on the terms to be discretized: among all the pairs $(t_k, \hat{\gamma}^k)$ generated by our algorithm, we select for discretization only the $p\%$ with the highest information gain. Thus, after this selection, terms which have not been selected will not be discretized and, for input to our learners, will be considered as binary-valued (i.e. only presence/absence of the term in the document will be considered). Each selected term will instead be replaced by two new terms, as previously described.

A full description of this algorithm, that in the following we will refer to as Algorithm A, is given in Figure 1.

Input: $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_{|Tr|}, C_{|Tr|}\rangle\}$, the training set of documents
$T = \{t_1, \ldots, t_{|T|}\}$, the set of terms
$\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$, the set of categories
p, the percentage of computed splits to be actually selected

Body:
1. $Optimal\_splits \leftarrow \emptyset$
2. For each $t_k \in T$ do
   - Sort the values that term t_k actually takes in Tr into an ordered sequence $\langle v_1, v_2, \ldots\rangle$ and compute the set of possible splits (see [2])
   $$Splits_k = \{\gamma_r^k | \gamma_r^k = \frac{v_r + v_{r+1}}{2}\}$$
   - Let
   $$\hat{\gamma}^k = \arg \max_{\gamma_q^k \in Splits_k} IG(Tr, \alpha, \beta, \gamma_q^k)$$
   - $Optimal\_splits \leftarrow Optimal\_splits \cup \{(t_k, \hat{\gamma}^k)\}$

Output: Return the p% splits $(t_k, \hat{\gamma}^k)$ in $Optimal\_splits$ with highest $IG(Tr, \alpha, \beta, \hat{\gamma}^k)$

Fig. 1. Algorithm A.

4.2 Algorithm B

Given a term t_k, a variant of Algorithm A can instead be obtained by first of all computing the best split with respect to a single category c_i. This is done by computing the information gain over a training set in which a document belonging to c_i is considered positive, otherwise it is considered negative. Then, again, we select the p% splits with highest information gain.

A full description of the algorithm, that in the following we will refer to as Algorithm B, is given in Figure 2.

5 Experimental results

5.1 Experimental setting

We have conducted a number of experiments to test the validity of the two algorithms proposed in Section 4. For these experiments we have used the well-

Input: $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_{|Tr|}, C_{|Tr|}\rangle\}$, the training set of documents
$T = \{t_1, \ldots, t_{|T|}\}$, the set of terms
$\mathcal{C}$, the set of categories
p, the percentage of computed splits to be actually selected

Body:

1. For each $c_i \in \mathcal{C}$ define a training set Tr_i in which documents belonging to category c_i are positive examples and documents not belonging to category c_i are negative examples
2. $Optimal\_splits \leftarrow \emptyset$
3. For each $t_k \in T$ do
   - Sort the values that term t_k actually takes in Tr into an ordered sequence $\langle v_1, v_2, \ldots\rangle$ and compute the set of possible splits (see [2])

     $$Splits_k = \{\gamma_r^k | \gamma_r^k = \frac{v_r + v_{r+1}}{2}\}$$
   - $Candidate\_splits_k \leftarrow \emptyset$
   - For each category $c_i \in \mathcal{C}$ that has at least one positive training example d_j such that $w_{kj} > 0$ do
     - compute

       $$\hat{\gamma}_i^k = \arg \max_{\gamma_q^k \in Splits_k} IG(Tr_i, \alpha, \beta, \gamma_q^k)$$
     - $Candidate\_splits_k \leftarrow Candidate\_splits_k \cup \{(t_k, \hat{\gamma}_i^k)\}$
   - Let $(t_k, \hat{\gamma}^k) \in Candidate\_splits_k$ be the split with highest IG for t_k
   - $Optimal\_splits \leftarrow Optimal\_splits \cup \{(t_k, \hat{\gamma}^k)\}$
4. For each $(t_k, \hat{\gamma}_k) \in Optimal\_splits$ compute $IG(Tr, \alpha, \beta, \hat{\gamma}^k)$

Output: Return the best $p\%$ splits $(t_k, \hat{\gamma}^k)$ in $Optimal\_splits$

Fig. 2. Algorithm B.

known "Reuters-21578, Distribution 1.0" corpus[6], which consists of a set of 12,902 news stories, partitioned (according to the "ModApté" split we have adopted) into a training set of 9,603 documents and a test set of 3,299 documents. The documents have an average length of 211 words (that become 117 after stop word removal). We have run our experiments on the set of 115 categories that have at least 1 positive training example; the average number of categories per document is 1.08, ranging from a minimum of 0 to a maximum of 16. The number of positive training examples per category ranges from a minimum of 1 to a maximum of 3964.

As the set of terms T we use the set of words occurring at least once in the training set. This set is identified by previously removing punctuation and then removing stop words. Neither stemming nor explicit number removal have been

[6] The Reuters-21578 corpus is freely available for experimentation purposes from `http://www.daviddlewis.com/resources/testcollections/~reuters21578/`

performed. As a result, the number of different terms is 17,439. Starting from this set of terms, *term space reduction* has been applied using χ^2_{max} as term selection function and a reduction factor of 90%, since this proved one among the best choices in the thorough experiments of [18].

Classification effectiveness has been measured in terms of the classic IR notions of precision (π) and recall (ρ) adapted to the case of text categorization. In our experiments we have evaluated both the "microaveraged" and the "macroaveraged" versions of π and ρ. As a measure of effectiveness that combines the contributions of both π and ρ, we have used the well-known F_1 function [8], in both the microaveraged and the macroaveraged versions.

5.2 The experiments

We have tested our discretization algorithms on the implementations of both AdaBoost.MH and AdaBoost.MH^{KR} described in [17], running them in the same experimental conditions in experiments with or without discretization. An alternative method might have been to just run the experiments with discretization and compare the results with the ones of the experiments without discretization published in [15] and [17]. We decided to avoid this latter method because of a number of reasons that would have made this comparison difficult:

- [15] uses an older version of the Reuters benchmark, called Reuters-22173. This benchmark is known to suffer from a number of problems that make its results difficult to interpret, and the research community universally prefers the better version Reuters-21578. No experiments using both collections have been reported in the literature, so there is no indication as to how results obtained on these two different collections might be compared.
- Apart from single words, [15] uses also bigrams (i.e. statistical phrases of length 2) as terms, while we use unigrams (i.e. single words) only.
- The experiments presented in [17] were run with a suboptimal version of the text preprocessing algorithm[7], which would make the comparison unfair towards [17].
- The experiments presented here use the optimized version of AdaBoost.MH^{KR} described in Section 3, and not the original one described in [17].

Our experiments were conducted by applying both AdaBoost.MH and AdaBoost.MH^{KR} to the data:

- data1: original binary representations, i.e. obtained without discretization by just checking the presence/absence of a term in the document;
- data2: binary representations obtained by discretizing previously obtained non-binary representations by means Algorithm A;
- data3: same as data2, but with Algorithm B in place of Algorithm A.

[7] Among other things, that version of the text preprocessing algorithm mistakenly did not make use of the TITLE field and of part of the BODY field of Reuters-21578 articles, which means that information important for the categorization task went unused.

The non-binary representations used in data2 and data3 were obtained by means of the $tfidf$ function in its standard "ltc" variant [13], i.e.

$$tfidf(t_k, d_j) = tf(t_k, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)} \tag{11}$$

where $\#_{Tr}(t_k)$ denotes the number of documents in Tr in which t_k occurs at least once and

$$tf(t_k, d_j) = \begin{cases} 1 + \log \#(t_k, d_j) & \text{if } \#(t_k, d_j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in d_j. Weights obtained by Equation 11 are normalized by cosine normalization to yield the final weights, i.e.

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|} tfidf(t_s, d_j)^2}} \tag{12}$$

In all the experiments with AdaBoost.MH^{KR} we have used the parameter settings $C = 1$ and $N = 20$, since in [17] these choices had been empirically found to give the best results.

Preliminary experiments involving the application of both AdaBoost.MH and AdaBoost.MH^{KR} to data preprocessed by methods data2 and data3 with different values for the p parameter had shown that, for both learners, the best performance is obtained for $p = 20$ [8].

In Figure 3 we have thus reported both the microaveraged F_1 (top) and the macroaveraged F_1 (bottom) curves obtained by AdaBoost.MH when applied for 5,000 iterations to data1, data2, and data3 (with $p = 20$ used for obtaining data2 and data3). From the results it is clear that

- The use of discretization algorithms (either A or B) brings about an improvement in both microaveraged and macroaveraged F_1 with respect to the application of AdaBoost.MH without discretization. This is relevant, since AdaBoost.MH has been shown to be one of the top performers in text categorization experiments so far.
- The improvement obtained by Algorithm B is more significant than that of Algorithm A on microaveraged F_1, while on macroaveraged F_1 the performances of the two algorithms are comparable. This indicates that Algorithm A performs well also on scarcely populated categories, while Algorithm B is more at home with densely populated ones.

The results of analogous experiments for the AdaBoost.MH^{KR} learner are shown in Figure 4. Even in this case we may observe that the representations

[8] The values we have tested for parameter p are 10, 20, 35, 50, 75, 100. For values of p higher than 20 we observed overfitting, while the performance obtained for $p = 10$ was intermediate between the one obtained without discretization and the one obtained for $p = 20$.

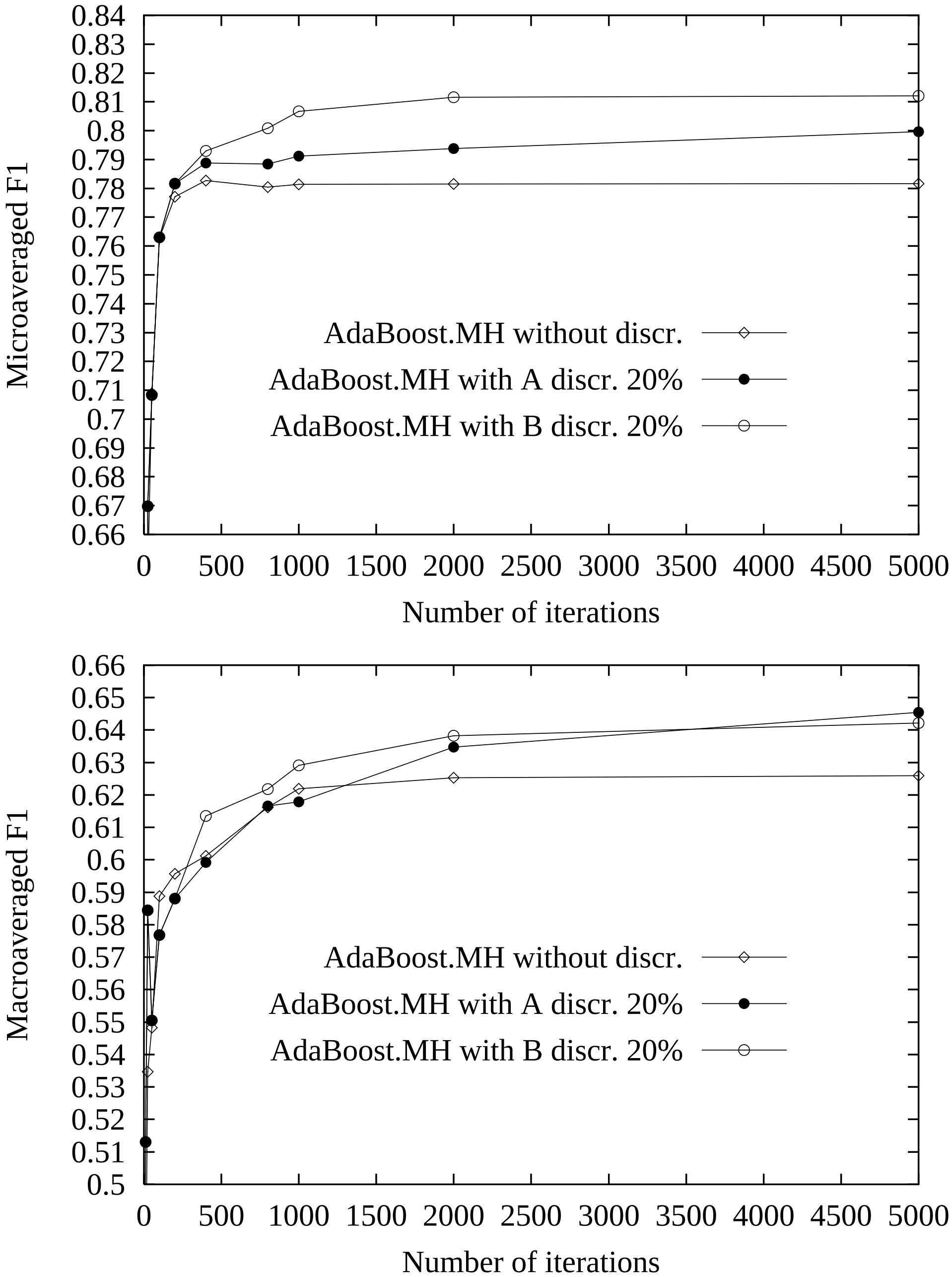

Fig. 3. F_1 curves for AdaBoost.MH applied to data without discretization and with $p = 20\%$ discretization (by both Algorithm A and Algorithm B).

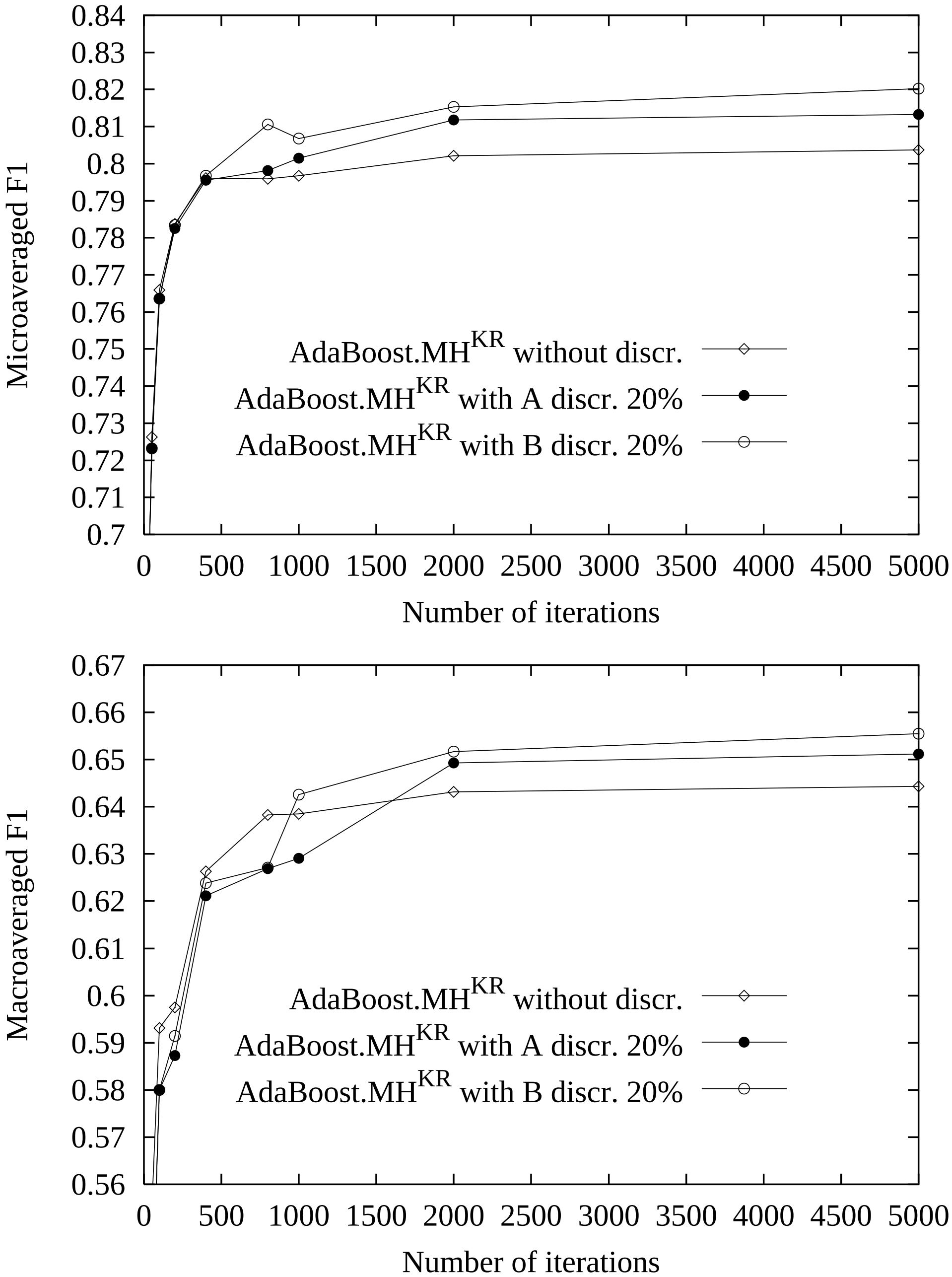

Fig. 4. F_1 curves for AdaBoost.MH^{KR} applied to data without discretization and with $p = 20\%$ discretization (by both Algorithm A and Algorithm B).

obtained by the discretization algorithms (either Algorithm A or B) consistently outperform the original binary representations. This time, however, Algorithm B is superior to Algorithm A both in terms of microaveraged and macroaveraged F_1, but the differential between the two is smaller than with AdaBoost.MH.

Finally, we note that AdaBoost.MH^{KR} proves superior to AdaBoost.MH also with representations obtained by discretization (either Algorithm A or B), and for both microaveraged and macroaveraged F_1, thus confirming the results obtained in [17] in which the two learners had been experimentally compared on binary representations obtained without any discretization.

6 Conclusion

We have presented two algorithms for the discretization of non-binary term weights, a task that addresses the problem of exploiting the rich information contained in the non-binary weights produced by standard statistical or probabilistic term weighting techniques, in the context of high performance learners requiring binary input. Although these algorithms can also be used in connection with learners not belonging to the "boosting" family, we have focused on experimenting them with two boosting-based learners, AdaBoost.MH and AdaBoost.MH^{KR}, since these had delivered top-notch performance in previous text categorization experiments.

These experiments have shown that binary representations obtained by discretizing previously obtained non-binary ($tf * idf$) representations by means of any of our two algorithms, outperform the original binary representations. This improvement is especially significant in the case of microaveraged F_1, for which an improvement of more than 3% was observed for AdaBoost.MH. This is significant, since AdaBoost.MH is in the restricted lot of the peak text categorization performers nowadays, a lot where the margins for performance improvement are slimmer and slimmer.

Acknowledgements

We thank Luigi Galavotti for making available his RealCAT text classification software environment [4], which greatly simplified our experimental work. We also thank Marco Danelutto for giving us access to the Backus cluster of PCs.

References

1. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceeding of ICML-95, 12th International Conference on Machine Learning*, pages 194–202, Lake Tahoe, US, 1995.
2. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of IJCAI-93, 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, Sidney, AU, 1993.

3. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
4. L. Galavotti, F. Sebastiani, and M. Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In J. L. Borbinha and T. Baker, editors, *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries*, pages 59–68, Lisbon, PT, 2000.
5. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.
6. R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of AAAI-92, 10th Conference of the American Association for Artificial Intelligence*, pages 123–128, San Jose, US, 1998.
7. R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of KDD-96, 2nd International Conference on Knowledge Discovery and Data Mining*, pages 114–119, Portland, US, 1996.
8. D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995.
9. T. M. Mitchell. *Machine learning*. McGraw Hill, New York, US, 1996.
10. L. C. Molina Félix, S. Oliveira Rezende, M. C. Monard, and C. Wellington Caulkins. Some experiences with the discretization process: from regression to classification. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 155–166, Buenos Aires, AR, 1999.
11. B. Pfahringer. Compression-based discretization of continuous attributes. In *Proceeding of ICML-95, 12th International Conference on Machine Learning*, pages 339–350, Lake Tahoe, US, 1995.
12. J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.
13. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
14. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
15. R. E. Schapire and Y. Singer. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
16. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
17. F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to automated text categorization. In A. Agah, J. Callan, and E. Rundensteiner, editors, *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, McLean, US, 2000.
18. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997.