An Improved Boosting Algorithm and its Application to Text Categorization

Fabrizio Sebastiani
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
56100 Pisa, Italy
fabrizio@iei.pi.cnr.it

Alessandro Sperduti
Dipartimento di Informatica
Università di Pisa
56125 Pisa, Italy
perso@di.unipi.it

Nicola Valdambrini
Dipartimento di Informatica
Università di Pisa
56125 Pisa, Italy
valdambr@supereva.it

ABSTRACT

We describe AdaBoost.MH^{KR}, an improved boosting algorithm, and its application to text categorization. Boosting is a method for supervised learning which has successfully been applied to many different domains, and that has proven one of the best performers in text categorization exercises so far. Boosting is based on the idea of relying on the collective judgment of a committee of classifiers that are trained sequentially. In training the i-th classifier special emphasis is placed on the correct categorization of the training documents which have proven harder for the previously trained classifiers. AdaBoost.MH^{KR} is based on the idea to build, at every iteration of the learning phase, not a single classifier but a sub-committee of the K classifiers which, at that iteration, look the most promising. We report the results of systematic experimentation of this method performed on the standard Reuters-21578 benchmark. These experiments have shown that AdaBoost.MH^{KR} is both more efficient to train and more effective than the original AdaBoost.MH^{R} algorithm.

1. INTRODUCTION

Text categorization (TC) is the activity of automatically building, by means of machine learning (ML) techniques, *automatic text classifiers*, i.e. programs capable of labelling natural language texts with thematic categories from a predefined set $C = \{c_1, \ldots, c_m\}$. The construction of an automatic text classifier requires the availability of a corpus $Co = \{\langle d'_1, C_1\rangle, \ldots, \langle d'_h, C_h\rangle\}$ of preclassified documents[1], where a pair $\langle d'_j, C_j\rangle$ indicates that document d'_j belongs to all and only the categories in $C_j \subseteq C$. A general inductive process (called the *learner*) automatically builds a classifier for the set C by learning the characteristics of C from a *training set* $Tr = \{\langle d'_1, C_1\rangle, \ldots, \langle d'_g, C_g\rangle\} \subset Co$ of documents. Once a classifier has been built, its effectiveness (i.e. its capability to take the right categorization decisions) may be tested by applying it to the *test set* $Te = \{\langle d'_{g+1}, C_{g+1}\rangle, \ldots, \langle d'_h, C_h\rangle\} = Co - Tr$ and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the corpus[2].

A wealth of different ML methods have been applied to TC, including probabilistic classifiers, decision trees, decision rules, regression methods, batch and incremental linear methods, neural networks, example-based methods, and support vector machines (see [17] for a review). In recent years, the method of classifier *committees* (or *ensembles*) has also gained popularity in the TC community. This method is based on the idea that, given a task that requires expert knowledge to be performed, S independent experts may be better than one if their individual judgments are appropriately combined. In TC, the idea is to apply S different classifiers $\Phi_1, \ldots, \Phi_S$ to the same task of deciding under which set $C_j \subseteq C$ of categories document d_j should be classified, and then combine their outcome appropriately. Usually, the classifiers are different either in terms of the *indexing* approach followed (i.e. the method by which document representations are automatically obtained) [16], or in terms of the ML method by means of which they have been built [5, 9, 11], or both. A classifier committee is then characterised by (i) a choice of S classifiers, and (ii) a choice of a combination function.

The *boosting* method [2, 3, 12, 13, 14] occupies a special place in the classifier committees literature, since the S classifiers $\Phi_1, \ldots, \Phi_S$ (here called the *weak hypotheses*) forming the committee are obtained by the same learning method (here called the *weak learner*) and work on the same text representation. The key intuition of boosting is that the S weak hypotheses should be trained not in a conceptually parallel and independent way, as in the other classifier

[1] In the following we use variables $d_1, d_2, \ldots$ to indicate generic documents and variables $d'_1, d'_2, \ldots$ to indicate preclassified documents.

[2] In this paper we make the general assumption that a document d_j can in principle belong to zero, one or many of the categories in C; this assumption is indeed verified in the Reuters-21578 benchmark that we use for our experiments. All the techniques we discuss in this paper can be straightforwardly adapted to the case in which each document belongs to exactly one category.

committees described above, but sequentially, one after the other. In this way, the training of hypothesis Φ_i may take into account how hypotheses $\Phi_1, \ldots, \Phi_{i-1}$ perform on the training documents, and may concentrate on getting right those training documents on which $\Phi_1, \ldots, \Phi_{i-1}$ have performed worst.

In experiments conducted over three different TC test collections, Schapire et al. [15] have shown the AdaBoost.MH boosting algorithm (an adaptation of Freund and Schapire's AdaBoost algorithm [3] using a one-level decision tree as the weak learner) to outperform Sleeping Experts, a classifier that had proven quite effective in the experiments of [1]. Further experiments by Schapire and Singer [14] showed AdaBoost.MH to outperform, aside from Sleeping Experts, a Naïve Bayes classifier, a standard Rocchio classifier, and Joachims' PrTFIDF classifier [6]. Weiss et al. [18] have used boosting with slightly more complex decision trees as the weak learners, and in doing so have outperformed all other text categorization approaches on Reuters-21578, the standard benchmark of text categorization. Boosting has also been used in [11], where the authors have reported a significant (13%) improvement in effectiveness over the pure weak learner. Boosting approaches are thus, on a par with support vector machine classifiers [7], the currently best performers in the TC arena (see [17, Table 6] for a comparative list of published results on Reuters-21578 and other TC benchmarks). Improving on the results of boosting is thus a challenging problem for text categorization.

In this work we present an improved boosting algorithm, that we call AdaBoost.MH^{KR}, and describe the experimental results we have obtained on the Reuters-21578 text categorization benchmark. AdaBoost.MH^{KR} is based on a different method to create weak hypotheses. At each iteration s, AdaBoost.MH^{KR} outputs not a single hypothesis, but a sub-committee of the $K(s)$ hypotheses which, at that iteration, look the most promising. The rest of the paper is structured as follows. In Section 2 we describe in detail the AdaBoost.MH algorithm, which may be considered representative of the "standard" way of doing boosting and which we will use as our baseline. In Section 3 we turn to describing our improved boosting algorithm. The results of its experimentation on Reuters-21578 are described in Section 4; Section 4.3 briefly discusses our parallel implementations of AdaBoost.MH and AdaBoost.MH^{KR}. Section 5 concludes.

2. BOOSTING AND ADABOOST.MH

Boosting is a method for generating a highly accurate classification rule (also called *final hypothesis*) by combining a set of moderately accurate hypotheses (also called *weak hypotheses*).

AdaBoost.MH (see Figure 1) is a boosting algorithm proposed by Schapire and Singer [14] for the text categorization task and derived from AdaBoost, Freund and Schapire's general purpose boosting algorithm [3]. The input to the algorithm is a training set $Tr = \{\langle d'_1, C_1\rangle, \ldots, \langle d'_g, C_g\rangle\}$, 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 linear combination $\Phi = \sum_{s=1}^{S} \alpha_s \Phi_s$ of these weak hypotheses (the choice of the α_s parameters will be discussed later). 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 $\Phi_s(d_j, c_i)$ 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. We instead interpret 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 tests the effectiveness of the newly generated weak hypothesis Φ_s on 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

$$D_{s+1}(d'_j, c_i) = \frac{D_s(d'_j, c_i) \exp(-\alpha_s \cdot C_j[c_i] \cdot \Phi_s(d'_j, c_i))}{Z_s} \quad (1)$$

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

$$Z_s = \sum_{i=1}^{m} \sum_{j=1}^{g} D_s(d'_j, c_i) \exp(-\alpha_s \cdot C_j[c_i] \cdot \Phi_s(d'_j, c_i)) \quad (2)$$

is a normalization factor chosen so that $\sum_{i=1}^{m} \sum_{j=1}^{g} D_{s+1}(d'_j, c_i) = 1$. If α_s is positive (this will indeed be the case, as discussed below), Equation 1 is such that the weight assigned to a pair $\langle d'_j, c_i\rangle$ misclassified by Φ_s is increased, as for such a pair $C_j[c_i]$ and $\Phi_s(d'_j, c_i)$ have different signs and the factor $C_j[c_i] \cdot \Phi_s(d'_j, c_i)$ is thus negative; likewise, the weight assigned to a pair correctly classified by Φ_s is decreased.

2.1 Choosing the weak hypotheses

In AdaBoost.MH each document d_j is represented as a vector $\langle w_{1j}, \ldots, w_{rj}\rangle$ of r binary weights, where $w_{kj} = 1$ means that term t_k occurs in document d_j and $w_{kj} = 0$ means that it does not; $\{t_1, \ldots, t_r\}$ is the set of terms that occur in at least one document in Tr. Of course, AdaBoost.MH does not make any assumption on what constitutes a term; single words, stems of words, or phrases are all plausible choices.

The weak hypotheses AdaBoost.MH deals with have the form

$$\Phi_s(d'_j, c_i) = \begin{cases} c_{0i} & \text{if } w_{kj} = 0 \\ c_{1i} & \text{if } w_{kj} = 1 \end{cases} \quad (3)$$

where $t_k \in \{t_1, \ldots, t_r\}$ and c_{0i} and c_{1i} are real-valued constants. The choices for t_k, c_{0i} and c_{1i} are in general different for each iteration, and are made according to an error-minimization policy described in the rest of this section.

Schapire and Singer [13] have proven that the *Hamming loss* of the final hypothesis Φ, defined as the percentage of pairs $\langle d'_j, c_i\rangle$ for which $sign(C_j[c_i]) \neq sign(\Phi(d'_j, c_i))$, is at most $\Pi_{s=1}^{S} Z_s$. The Hamming loss of a hypothesis Φ_s is a

Input: A training set $Tr = \{\langle d'_1, C_1\rangle, \ldots, \langle d'_g, C_g\rangle\}$
where $C_j \subseteq C = \{c_1, \ldots, c_m\}$ for all $j = 1, \ldots, g$.

Body: Let $D_1(d'_j, c_i) = \dfrac{1}{mg}$ for all $j = 1, \ldots, g$ and for all $i = 1, \ldots, m$
For $s = 1, \ldots, S$ do:

- pass distribution $D_s(d'_j, c_i)$ to the weak learner;
- get the weak hypothesis Φ_s from the weak learner;
- choose $\alpha_s \in \mathbb{R}$;
- set $D_{s+1}(d'_j, c_i) = \dfrac{D_s(d'_j, c_i)\exp(-\alpha_s \cdot C_j[c_i] \cdot \Phi_s(d'_j, c_i))}{Z_s}$

  where $Z_s = \sum_{i=1}^{m}\sum_{j=1}^{g} D_s(d'_j, c_i)\exp(-\alpha_s \cdot C_j[c_i] \cdot \Phi_s(d'_j, c_i))$

  is a normalization factor chosen so that $\sum_{i=1}^{m}\sum_{j=1}^{g} D_{s+1}(d'_j, c_i) = 1$

Output: A final hypothesis $\Phi(d, c) = \sum_{s=1}^{S} \alpha_s \Phi_s(d, c)$

Figure 1: The AdaBoost.MH algorithm.

measure of its classification effectiveness; therefore, a reasonable (although suboptimal) way to maximize the effectiveness of the final hypothesis Φ is to "greedily" choose each weak hypothesis Φ_s (and thus its parameters t_k, c_{0i} and c_{1i}) and each parameter α_s in such a way as to minimize the normalization factor Z_s.

Schapire and Singer [14] define three different variants of AdaBoost.MH, corresponding to three different methods for making these choices:

1. AdaBoost.MH *with real-valued predictions* (here nicknamed AdaBoost.MH^R);
2. AdaBoost.MH *with real-valued predictions and abstaining* (AdaBoost.MH^{RA});
3. AdaBoost.MH *with discrete-valued predictions* (AdaBoost.MH^D).

In this paper we concentrate on AdaBoost.MH^R, since it is the one that, in the experiments of [14], has been experimented most thoroughly and has given the best results; the modifications to AdaBoost.MH^R that we discuss in Section 3 straightforwardly apply also to the other two variants. AdaBoost.MH^R chooses weak hypotheses of the form described in Equation 3 by a two step-process:

1. for each term $t_k \in \{t_1, \ldots, t_r\}$ 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;
2. among all the hypotheses $\Phi^1_{best}, \ldots, \Phi^r_{best}$ pre-selected for the r different terms, it selects the one (indicated by Φ_s) for which Z_s is minimum.

Step 1 is clearly the key step, since there are a non-enumerable set of weak hypotheses that have t_k as the pivot term. Schapire and Singer [13] have proven that, given term t_k and category c_i, Φ^k_{best} is obtained when $\alpha_s = 1$ and $c_{xi} = \frac{1}{2}\ln\left(\frac{W^{xik}_1}{W^{xik}_{-1}}\right)$, where

$$W^{xik}_b = \sum_{j=1}^{g} D_t(d'_j, c_i) \cdot [\![w_{kj} = x]\!] \cdot [\![C_j[c_i] = b]\!] \qquad (4)$$

for $b \in \{1, -1\}$, $x \in \{0, 1\}$, $i \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, r\}$, and where $[\![\pi]\!]$ indicates the characteristic function of predicate π (i.e. the function that returns 1 if π is true and 0 otherwise). For these values of α_s and c_{xi} we obtain

$$Z_s = 2\sum_{i=1}^{m}\sum_{x=0}^{1}(W^{xik}_1 W^{xik}_{-1})^{\frac{1}{2}} \qquad (5)$$

Choosing $\frac{1}{2}\ln\left(\frac{W^{xik}_1}{W^{xik}_{-1}}\right)$ as the value for c_{xi} has the effect that $\Phi_s(d_j, c_i)$ outputs a positive real value in the two following cases:

1. $w_{kj} = 1$ (i.e. t_k occurs in d_j) and the majority of the training documents in which t_k occurs belong to c_i;
2. $w_{kj} = 0$ (i.e. t_k does not occur in d_j) and the majority of the training documents in which t_k does not occur belong to c_i.

In all the other cases Φ_s outputs a negative real value. Here, "majority" has to be understood in a weighted sense, i.e. by bringing to bear the weight $D_t(d'_j, c_i)$ associated to the training pair $\langle d'_j, c_i\rangle$. The larger this majority is, the higher the absolute value of $\Phi_s(d_j, c_i)$ is; this means that this absolute value represents a measure of the confidence that Φ_s has in its own decision.

In practice, the value $c_{xi} = \frac{1}{2}\ln\left(\frac{W_{1}^{xik}+\epsilon}{W_{-1}^{xik}+\epsilon}\right)$ is chosen in place of $c_{xi} = \frac{1}{2}\ln\left(\frac{W_{1}^{xik}}{W_{-1}^{xik}}\right)$, since this latter may produce outputs with a very large or infinite absolute value when the denominator is very small or zero[3].

The output of the final hypothesis is the value

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

obtained by summing the outputs of the weak hypotheses.

3. AN IMPROVED BOOSTING ALGORITHM AND ITS APPLICATION TO TEXT CATEGORIZATION

We here propose a new method, called AdaBoost.MH^{KR} (for AdaBoost.MH *with K-fold real-valued predictions*) that differs from AdaBoost.MH^{R} in the policy according to which weak hypotheses are chosen. AdaBoost.MH^{KR} 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 3. 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), we select the best $K(s)$ terms and use them in order to generate $K(s)$ SWHs $\Phi_s^1, \ldots, \Phi_s^{K(s)}$. Our CWH is then produced by grouping $\Phi_s^1, \ldots, \Phi_s^{K(s)}$ into a sub-committee

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

that uses the simple arithmetic mean as the combination rule. For updating the distribution we still apply Equations 1 and 2, where Φ_s is now defined by Equation 7. The final hypothesis is computed by plugging Equation 7 into Equation 6, thus obtaining

$$\Phi(d_j, c_i) = \sum_{s=1}^{S} \alpha_s \frac{1}{K(s)} \sum_{q=1}^{K(s)} \Phi_s^q(d_j, c_i) \tag{8}$$

The idea of using the $K(s)$ best terms, instead of simply using the top-ranked one, comes from the analysis of the scores assigned to terms $t_1, \ldots, t_r$ at different iterations of AdaBoost.MH^{R}. Here, by the *score* of a term t_k at iteration s we mean the value that Z_s would take if t_k were chosen as the pivot term for iteration s; as described in Section 2.1, at each iteration the term with the *lowest* score is thus chosen as the pivot. Figure 2 plots, for four sample iterations (3, 10, 50 and 99) of AdaBoost.MH^{R}, the score of each term as a function of the rank position the term has obtained for that iteration in our experiments on Reuters-21578. It can be noted that, while in the first iterations (especially Iteration 3) the best terms have a score

[3]In [14] the value for ϵ is chosen by 3-fold cross validation on the training set, but this procedure is reported to give only marginal improvements with respect to the default choice of $\epsilon = \frac{1}{mg}$.

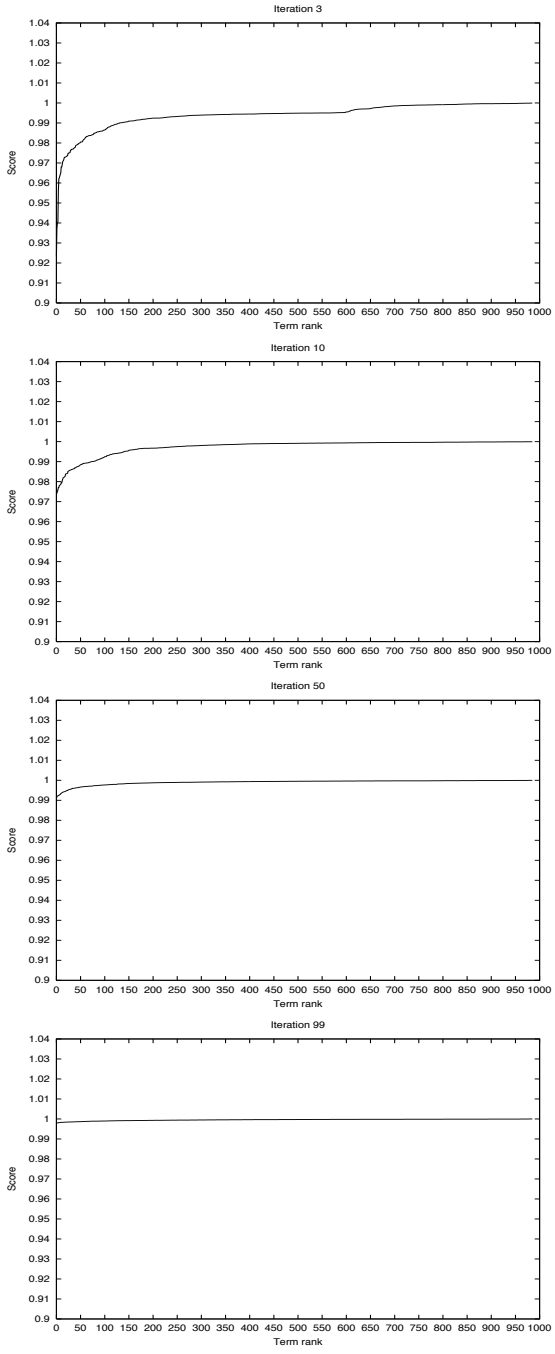

Figure 2: Plots, obtained at four sample iterations, of Z_s as a function of the rank position that the term has obtained for that iteration (terms are ranked on the X axis; higher values of X thus correspond to the worst-scoring terms). These plots show how the term scores Z_s asymptotically tend to 1, even for the best terms, as the number s of iterations grows.

markedly different from each other and from the worst ones (as indicated by the initial steep ascent of the curve), in the last iterations the differences among scores are very small (as indicated by the very flat profile of the curve) and the scores tend to become equal to 1 for all terms. This means that, as boosting progresses, the score Z_s is increasingly unable to discriminate well among different terms.

In AdaBoost.MH^{KR} we choose, at each iteration, $K(s)$ top-ranked terms that have similar scores and that would have been good candidates for selection in the next $K(s)$ AdaBoost.MH^{R} iterations. In this way we can build a final hypothesis composed of S' SWHs (of the form of Equation 3) grouped into S CWHs at a computational cost comparable to the one required by AdaBoost.MH^{R} to generate a committee of S SWHs, with $S' \gg S$. In fact, as obvious from what we said in Section 2.1, most of the computation required by the boosting process is devoted to calculating the term scores, and by using only the top-scoring term AdaBoost.MH^{R} exploits these hard-won scores only to a very small extent. On the contrary, AdaBoost.MH^{KR} tries to put these scores to maximum use by using more term scores, hence more information on how documents and categories are associated, immediately, without waiting for further iterations[4].

This analysis is valid only if the scores for the $K(s)$ terms are very close to each other (lest the original purpose of boosting is lost), so that it would not make a substantial difference for AdaBoost.MH^{R} to choose one or the other as the pivot. This is the reason why we require the number K of SWHs that form the CWH Φ_s to be a function of s. As evident from the plots in Figure 2, in the first iterations we will need K to be small, since the scores for the best terms are quite different among each other, while in the last iterations we can have larger values of K, since the differences in scores are minimal. In the experiments described in Section 4 we have used the simple heuristics of adding a constant C to $K(s)$ every N iterations, using a fixed value for N and using a value of 1 for $K(1)$, i.e. $K(s) = 1 + C\lfloor\frac{s-1}{N}\rfloor$.

Finally, note that the fact that the scores of the $K(s)$ terms are very close to each other substantially justifies our use of the arithmetic mean, rather than a weighted linear combination, as a combination rule.

4. EXPERIMENTAL RESULTS

4.1 Experimental setting

We have conducted a number of experiments to test the validity of the method proposed in Section 3. For these experiments we have used the "Reuters-21578, Distribution 1.0" corpus[5], 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) and are labelled by 118 categories; 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 examples per category ranges from a minimum of 1 to a maximum of 3964. We have run our experiments on the set of 90 categories that have both at least 1 positive training example and at least 1 positive test example, as this is the most widely used category set in the literature on Reuters-21578 experimentation.

As the set of terms $t_1, \ldots, t_r$ 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 performed. As a result, the number of different terms is 17,439.

Classification effectiveness has been measured in terms of the classic IR notions of precision (Pr) and recall (Re) adapted to the case of document categorization. In our experiments we have evaluated both the "microaveraged" and the "macroaveraged" versions of Pr and Re. As a measure of effectiveness that combines the contributions of both Pr and Re, we have used the well-known F_1 function [10].

4.2 The experiments

In order to compare the effectiveness of AdaBoost.MH^{R} and AdaBoost.MH^{KR} we have implemented both algorithms and run them in the same experimental conditions in a number of different experiments. An alternative method might have been to just experiment AdaBoost.MH^{KR} and compare the results with the ones published in [14]. We decided to avoid this latter method because of a number of reasons that would have made this comparison difficult:

- [14] 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 is now universally oriented towards the use of 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.

- [14] uses also bigrams (i.e. statistical phrases of length 2), apart from single words, as terms, while we use unigrams (i.e. single words) only.

Our experiments were conducted by varying a number of parameters, such as the number of iterations S in the boosting process, the C and N parameters in the AdaBoost.MH^{KR} updating rule for $K(s)$, and the *reduction factor* of the *term space reduction* process. Term space reduction refers to the process of identifying, prior to the invocation of the learning algorithm, a subset of the $r' \ll r$ terms that are deemed most useful for compactly representing the meaning of the documents. After such a reduction each (training or test) document d_j is represented by a vector $\langle w_{1j}, \ldots, w_{r'j}\rangle$ of weights shorter than the original; the value $\rho = \frac{r-r'}{r}$ is called the *reduction factor*. Feature selection is usually beneficial in that it tends to reduce both *overfitting* (i.e. the phenomenon by which a classifier tends to be better at classifying the data it has been trained on than at classifying other data) and the computational cost of training the classifier. We have used a "filtering" approach to term space

[4]The added cost of a single AdaBoost.MH^{KR} iteration is due to the need of ranking the r terms, which is an $O(r \log r)$ problem, rather than just finding the top-scoring one, which is an $O(r)$ problem. However, since $K(S)$ is typically smaller than $\log r$, it is cheaper to repeat $K(S)$ times the search for a top-scoring term, which means that we are still in $O(r)$.

[5]The Reuters-21578 corpus may be freely downloaded for experimentation purposes from `http://www.research.att.com/~lewis/reuters21578.html`

reduction [8]; this consists in scoring each term by means of a *term evaluation function* and then selecting the r' features with the highest score. We have used

$$\chi^2_{max}(t_k) = \max_{i=1,\dots,m} \frac{g \cdot [P(t_k, c_i)P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i)P(\overline{t}_k, c_i)]^2}{P(t_k)P(\overline{t}_k)P(c_i)P(\overline{c}_i)} \quad (9)$$

as our term evaluation function[6], since it is known from the literature to be one of the best performers at high reduction factors [20]. We have conducted various experiments by using reduction factors ρ of .96, .90, .00 (a reduction factor $\rho = .00$ means that no term reduction has been performed).

The results obtained with AdaBoost.MH^{KR} for these three reduction factors are shown in Table 1, while the equivalent experiments for AdaBoost.MH^{R} are reported in Table 2. Note that the two tables do not have the same amount of rows; the reason for this is a lack of computational resources that prevented us from making more thorough experiments, especially in the cases that are computationally most demanding (i.e. the experiments with AdaBoost.MH^{R} or with low reduction factors). In Table 1, the 3rd and 4th columns indicate the values used for the parameters C and N used for determining $K(s)$, while the 4th column indicates the total number S' of generated SWHs, computed as $S' = \sum_{s=1}^{S} K(s) = \sum_{s=1}^{S}(1 + C\lfloor \frac{s-1}{N} \rfloor)$. For instance, Row 1 in Table 1 specifies that the F^{μ}_{β} and F^{M}_{β} results indicated were obtained by a final hypothesis built after 50 AdaBoost.MH^{KR} iterations in which $K(s)$ was increased by 1 every 5 iterations; this resulted in a total of 275 SWHs being generated.

By comparing the results obtained by the two algorithms for $\rho = .96$ (upper parts of Tables 1 and 2) we can observe that AdaBoost.MH^{KR} was able to reach an F^{μ}_1 value of 0.740 after only 100 iterations using $C = 1$ and $N = 20$. With the same number of iterations AdaBoost.MH^{R} achieved an F^{μ}_1 value of only 0.697, and for this number of iterations AdaBoost.MH^{KR} is superior also in terms of F^{M}_1. While this comparison can be considered fair from a computational point of view, since the computational cost of each iteration for the two algorithms is basically equivalent, it may be argued that the number of hypotheses selected by AdaBoost.MH^{KR} is actually 300 (as reported in the 5th column of Table 1) versus the 100 selected by AdaBoost.MH^{R} (one hypothesis for each iteration). However, to this respect it should be noted that the effectiveness AdaBoost.MH^{KR} achieves thanks to these 300 hypotheses is not obtained by AdaBoost.MH^{R} even by generating 16000 hypotheses! Moreover, the best value of F^{μ}_1 for AdaBoost.MH^{R} was obtained at Iteration 1000; this means that AdaBoost.MH^{KR} (with $C = 1$ and $N = 20$) reached a better performance with approximately ten times less computational effort. In the vast majority of the cases, with the same number of iterations AdaBoost.MH^{KR} outperformed AdaBoost.MH^{R}.

The results obtained with reduction factors $\rho = .90$ and $\rho = .96$ (middle and lower parts of Tables 1 and 2) confirm the above results. Thus, independently from the reduction factor used in term space reduction, AdaBoost.MH^{KR} seems to be characterised by a higher effectiveness and a significantly higher efficiency.

The fact that AdaBoost.MH^{KR} is more effective than AdaBoost.MH^{R} could be explained by the greedy approach followed by AdaBoost.MH^{R} for the selection of the best hypothesis to include in the final one. In fact, the greedy approach does not guarantee the optimality of the selection, and AdaBoost.MH^{R} does not have any possibility to explore the hypothesis space for the final hypothesis: given a set of training data, one and only one final hypothesis is generated for each possible iteration. On the contrary, AdaBoost.MH^{KR} allows the designer to explore, at least to a given degree, the hypothesis space for the final hypothesis by setting C and N to different values. Moreover, the use of CWHs that include an increasing number of SWHs reduces the variability associated to SWHs which are generated later in the learning process, thus reducing the impact on the distribution of hypotheses which may turn out to be too specific.

ρ	S	C	N	S'	F^{μ}_1	F^{M}_1
.96	50	1	5	275	0.666667	0.513684
.96	97	1	10	520	0.725000	0.565000
.96	50	1	10	150	0.731658	0.549577
.96	50	2	10	170	0.689655	0.512637
.96	50	1	11	140	0.716192	0.565011
.96	50	1	12	130	0.726818	0.558150
.96	50	1	13	122	0.727083	0.550834
.96	50	1	14	116	0.723666	0.540245
.96	50	1	15	110	0.724629	0.532600
.96	50	1	20	90	0.704566	0.511514
.96	50	2	20	110	0.721946	0.531515
.96	50	4	20	170	0.702740	0.515649
.96	100	2	20	340	0.694929	0.544996
.96	100	1	20	300	**0.740828**	0.557656
.96	197	1	20	1070	0.728946	0.572078
.96	100	1	30	220	0.738643	0.558622
.96	100	2	30	260	0.735571	0.555456
.96	500	1	80	1820	0.724133	**0.591866**
.96	500	1	200	900	0.738584	0.570910
.90	50	1	12	130	0.726213	0.511327
.90	50	1	13	122	0.728315	0.525439
.90	50	1	15	110	0.723751	0.525064
.90	50	1	16	104	0.717628	0.527743
.90	100	1	30	220	0.752109	0.532758
.90	200	1	20	1100	**0.764550**	0.558366
.90	500	1	80	1820	0.752034	**0.568556**
.90	500	1	200	900	0.757198	0.533829
.00	50	1	12	130	0.724138	0.514714
.00	50	1	13	122	0.722528	0.532095
.00	50	1	15	110	0.723751	0.525064
.00	100	1	30	220	0.740409	0.522325
.00	200	1	20	1100	**0.764550**	**0.558366**
.00	500	1	200	900	0.750773	0.519780

Table 1: Results obtained by using AdaBoost.MH^{KR} after performing χ^2_{max} feature selection (ρ indicates the reduction factor). The number $K(s)$ of SWHs generated at iteration s is increased by C every N iterations, which causes a total of S' SWHs to be generated in S iterations. The best result is indicated in boldface.

[6]In Equation 9 probabilities are interpreted on an event space of documents (e.g. $P(\overline{t}_k, c_i)$ indicates the probability that, for a random document x, term t_k does not occur in x and x belongs to category c_i), and are estimated by counting occurrences in the training set. In the same equation g indicates, as usual, the cardinality of the training set.

ρ	S	F_1^μ	F_1^M
.96	30	0.597000	0.486978
.96	50	0.656354	0.483030
.96	100	0.697740	0.523138
.96	167	0.702300	0.786115
.96	500	0.724986	0.565490
.96	760	0.726772	0.575426
.96	1000	**0.728390**	0.584169
.96	2000	0.725088	0.597137
.96	3000	0.721169	0.605509
.96	5000	0.722731	0.618689
.96	6000	0.721077	0.621494
.96	7000	0.720186	**0.623543**
.96	8000	0.717481	0.619853
.96	9000	0.717524	0.621415
.96	10000	0.716788	0.622729
.96	16000	0.713720	0.619019
.90	50	0.652111	0.458944
.90	100	0.708171	0.484279
.90	200	0.737705	0.508982
.90	500	**0.753718**	**0.540283**
.00	50	0.643192	0.460704
.00	100	0.704545	0.521491
.00	200	0.726456	0.534055
.00	**500**	**0.742699**	**0.514699**

Table 2: Results obtained by using ADABOOST.MH^R after performing χ^2_{max} feature selection (ρ indicates the reduction factor). The total number of generated SWHs is here equal to the number of iterations (i.e. one hypothesis is generated at each iteration). The best result is indicated in boldface.

4.3 A parallel implementation of ADABOOST.MH^R and ADABOOST.MH^{KR}

In order to speed up the computation we have realized a parallel implementation of both ADABOOST.MH^R and ADABOOST.MH^{KR} on a cluster of ten Pentium II 266MHz PCs. Although ADABOOST.MH is inherently a sequential algorithm, since each new weak hypothesis is selected on the basis of a distribution that depends on the previously selected hypotheses, it is possible to parallelize the computation required for the choice of the weak hypotheses.

For our parallel implementation we have adopted a FARM model of computation [4], as outlined in Figure 3. A process E partitions the set of r terms into M subsets and allocates each of them to one among M processors $W_1, W_2, \ldots, W_M$ (called *workers*). At each iteration s, each processor W_i finds the best hypothesis (ADABOOST.MH^R) or the first $K(s)$ best hypotheses (ADABOOST.MH^{KR}) among the ones that hinge on terms allocated to W_i, and forwards its output to process C. C collects and compares the outputs coming from $W_1, W_2, \ldots, W_M$, selects the best weak hypothesis Φ_s (ADABOOST.MH^R) or the best $K(s)$ hypotheses $\Phi_s^1, \ldots, \Phi_s^{K(s)}$ (ADABOOST.MH^{KR}), updates the distribution D accordingly, and sends the updated distribution to each worker, which will use it in the next round of computation.

In the implementation of ADABOOST.MH^R we have further optimized the final hypothesis $\Phi(d_j, c_i) = \sum_{s=1}^{S} \Phi_s(d_j, c_i)$ by "combining" the weak hypotheses $\Phi_1, \ldots, \Phi_S$ according to their pivot term t_k. In fact, note that if $\{\Phi_1, \ldots, \Phi_S\}$ contains a subset $\{\Phi_1^k, \ldots, \Phi_{q(k)}^k\}$ of weak hypotheses that hinge on t_k and are of the form

$$\Phi_r^k(d_j, c_i) = \begin{cases} c_{0i}^r & \text{if } w_{kj} = 0 \\ c_{1i}^r & \text{if } w_{kj} = 1 \end{cases} \quad (10)$$

for $r = 1, \ldots, q(k)$, the collective contribution of $\Phi_1^k, \ldots, \Phi_{q(k)}^k$ to the final hypothesis is the same as that of a "combined hypothesis"

$$\breve{\Phi}^k(d_j, c_i) = \begin{cases} \sum_{r=1}^{q(k)} c_{0i}^r & \text{if } w_{kj} = 0 \\ \sum_{r=1}^{q(k)} c_{1i}^r & \text{if } w_{kj} = 1 \end{cases} \quad (11)$$

In the implementation we have thus replaced $\sum_{s=1}^{S} \alpha_s \Phi_s(d_j, c_i)$ with $\sum_{k=1}^{\Delta} \breve{\Phi}^k(d_j, c_i)$, where Δ is the number of different terms that act as pivot for the weak hypotheses in $\{\Phi_1, \ldots, \Phi_S\}$. We have also done a similar optimization in the implementation of ADABOOST.MH^{KR}; the only difference is that in this latter case the factor $\frac{1}{K(s)}$ from Equation 8 needs to be taken into account.

This modification brings about a considerable efficiency gain in the application of the final hypothesis to a test example. For instance, the final hypothesis we obtained with ADABOOST.MH^{KR} with the parameters set to $\rho = .96$, $S = 100$, $C = 1$ and $N = 20$, consists of 300 SWHs, but the number of different pivot terms is only 168. The reduction in the size of the final hypothesis which derives from this modification is usually larger for high reduction factors, since in this case the number of different terms that can be chosen as the pivot is smaller.

Figure 3: The FARM model used for the parallel implementation of ADABOOST.MH^R and ADABOOST.MH^{KR}.

5. CONCLUSION

We have described ADABOOST.MH^{KR}, a boosting algorithm derived by ADABOOST.MH^R by modifying the policy for the choice of the weak hypotheses, and we have reported the results of its experimentation on Reuters-21578, the standard benchmark of text categorization research. The modification described applies straightforwardly to ADABOOST.MH^{RA} and ADABOOST.MH^D too. ADABOOST.MH^{KR} is substantially more efficient to train than ADABOOST.MH^R, and our experiments have shown that it is also more effective. This is even more significant once we note that we have adopted fairly unsophisticated policies (i) for the combination of the simple weak hypotheses into one complex weak hypothesis, and (ii) for the updating of the number $K(s)$ of simple weak hypotheses that should be selected at iteration s.

There are a number of ways in which we plan to continue our research. The first, obvious one is to explore more theoretically justified policies for performing tasks (i) and (ii) above. For instance, more refined policies for (ii) could involve e.g. using all the terms whose score differ by at most τ (with τ determined e.g. by k-fold cross-validation), or letting $K(s)$ be a function of the derivative of the curve $Z_s(t)$.

The second is to explore the possibility of using non-binary weights, such as the ones produced by standard $tf * idf$ term weighting techniques. The idea is that of segmenting the [0,1] interval, on which the non-binary weights typically range, into a fixed number Y of intervals, and searching a space of weak hypotheses that each realize a Y-ary branch (instead of the binary branch realized by the weak hypotheses of Equation 3).

The third, more challenging one is to explore variants of AdaBoost.MH^{KR} in which the evaluation of the weak hypotheses is not done in terms of Hamming distance, but in terms of F_1 itself. The reason for this is that in text categorization, unlike in many other machine learning applications, the number of negative examples of a given category c_i is usually overwhelmingly higher than the number of its positive examples. This means that, if Hamming distance is used as a yardstick of effectiveness, the trivial rejector (i.e. the classifier that "says no" to every $\langle d_j, c_i \rangle$ pair) may well turn out to be more "effective" than any other classifier induced by machine learning techniques [19]. This means in turn that learning approaches based on explicit error minimization, as AdaBoost.MH is, may well end up in training a text classifier to behave very similarly to the trivial rejector once "error" is understood in terms of Hamming distance. We conjecture that forcing AdaBoost.MH^{KR} to maximize the F_1 of the weak hypotheses generated, instead of minimizing their Hamming distance, should bring about higher recall at a comparatively small cost in terms of precision.

6. REFERENCES

[1] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.

[2] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 170–178, Madison, US, 1998.

[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] A. Hey. Experiments in MIMD parallelism. In *Proceedings of PARLE-89, European Conference on Parallel Architectures and Languages*, pages 28–42, Eindhoven, NL, 1989.

[5] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 279–288, Zürich, CH, 1996.

[6] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997.

[7] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998.

[8] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 121–129, New Brunswick, US, 1994.

[9] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, CH, 1996.

[10] D. D. Lewis. Evaluating and optmizing autonomous text classification systems. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995.

[11] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[12] R. E. Schapire. Theoretical views of boosting. In *Proceedings of EuroCOLT-99, 4th European Conference on Computational Learning Theory*, pages 1–10, Nordkirchen, DE, 1999.

[13] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[14] R. E. Schapire and Y. Singer. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[15] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU, 1998.

[16] S. Scott and S. Matwin. Feature engineering for text classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388, Bled, SL, 1999.

[17] F. Sebastiani. Machine learning in automated text categorisation: a survey. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999.

[18] S. M. Weiss, C. Apté, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[19] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[20] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997.