MP-Boost: A Multiple-Pivot Boosting Algorithm and its Application to Text Categorization

Andrea Esuli and Tiziano Fagni
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Via G Moruzzi, 1 – 56124 Pisa, Italy
{andrea.esuli,tiziano.fagni}@isti.cnr.it

Fabrizio Sebastiani
Dipartimento di Matematica Pura e Applicata
Università di Padova
Via GB Belzoni, 7 – 35131 Padova, Italy
fabrizio.sebastiani@unipd.it

ABSTRACT
AdaBoost.MH is a popular supervised learning algorithm for building multi-label (aka n-of-m) text classifiers. AdaBoost.MH belongs to the family of "boosting" algorithms, and works by iteratively building a committee of "decision stump" classifiers, where each such classifier is trained to especially concentrate on the document-class pairs that previously generated classifiers have found harder to correctly classify. Each decision stump hinges on a specific "pivot term", checking its presence or absence in the test document in order to take its classification decision. In this paper we propose an improved version of AdaBoost.MH, called MP-Boost, obtained by selecting, at each iteration of the boosting process, not one but *several* pivot terms, one for each category. The rationale behind this choice is that this provides highly individualized treatment for each category, since each iteration thus generates, for each category, the best possible decision stump. The result of the learning process is thus not a single classifier committee, but a set of such committees, one for each category. We present the results of experiments showing that MP-Boost is much more effective than AdaBoost.MH. In particular, the improvement in effectiveness is spectacular when few boosting iterations are performed, and (only) high for many such iterations. The improvement is especially significant in the case of macroaveraged effectiveness, which shows that MP-Boost is especially good at working with hard, infrequent categories.

1. INTRODUCTION
Given a set of textual documents D and a predefined set of *categories* (aka *labels*) $C = \{c_1, \ldots, c_m\}$, *multi-label* (aka n-of-m) *text classification* is the task of approximating, or estimating, an unknown *target function* $\Phi : D \times C \rightarrow \{-1, +1\}$, that describes how documents ought to be classified, by means of a function $\hat{\Phi} : D \times C \rightarrow \{-1, +1\}$, called the *classifier*, such that Φ and $\hat{\Phi}$ "coincide as much as possible"[1]. Here, "multi-label" indicates that the same document can belong to zero, one, or several categories at the same time.

AdaBoost.MH [13] is a popular supervised learning algorithm for building multi-label text classifiers. AdaBoost.MH belongs to the family of "boosting" algorithms (see [8] for a review), which have enjoyed a wide popularity in the text categorization and filtering community [1, 3, 4, 7, 9, 10, 13, 14, 15, 16] because of their state-of-the-art effectiveness and of the strong justifications they have received from computational learning theory. AdaBoost.MH works by iteratively building a committee of "decision stump" classifiers[2], where each such classifier is trained to especially concentrate on the document-category pairs that previously generated classifiers have found harder to correctly classify. Each decision stump hinges on a specific "pivot term", and takes its classification decision based on the presence or absence of the pivot term in the test document.

We here propose an improved version of AdaBoost.MH, called MP-Boost, obtained by selecting, at each iteration of the boosting process, not one but *several* pivot terms, one for each category. The rationale behind this choice is that this provides highly individualized treatment for each category, since each iteration generates, for each category, the best possible decision stump. The result of the learning process is thus not a single classifier committee, but a set of such committees, one for each category. We present the results of experiments showing that MP-Boost is much more effective than AdaBoost.MH. In particular, the improvement in effectiveness is spectacular when few boosting iterations are performed, and (only) high for many such iterations. The improvement is especially significant in the case of macroaveraged effectiveness, which shows that MP-Boost is especially good at working with hard, infrequent categories. This ultimately means that the same level of effectiveness can be obtained by MP-Boost with much fewer iterations than AdaBoost.MH requires.

The paper is structured as follows. In Section 2 we concisely describe boosting and the AdaBoost.MH algorithm. Section 3 describes in detail our MP-Boost algorithm and the

[1]Consistently with most mathematical literature we use the caret symbol (ˆ) to indicate estimation.
[2]A *decision stump* is a decision tree of depth one, i.e. consisting of a root node and two or more leaf nodes.

rationale behind it. Section 4 discusses the computational cost of MP-Boost relative to that of AdaBoost.MH. In Section 5 we present experimental results comparing AdaBoost.MH and MP-Boost. Section 6 concludes.

2. AN INTRODUCTION TO BOOSTING AND ADABOOST.MH

AdaBoost.MH [13] (see Figure 1) is a *boosting* algorithm, i.e. an algorithm that generates a highly accurate classifier (also called *final hypothesis*) by combining a set of moderately accurate classifiers (also called *weak hypotheses*). The input to the algorithm is a training set $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_g, C_g\rangle\}$, where $C_i \subseteq C$ is the set of categories to each of which d_i belongs.

AdaBoost.MH works by iteratively calling a *weak learner* to generate a sequence $\hat{\Phi}_1, \ldots, \hat{\Phi}_S$ of weak hypotheses; at the end of the iteration the final hypothesis $\hat{\Phi}$ is obtained as a sum $\hat{\Phi} = \sum_{s=1}^{S} \hat{\Phi}_s$ of these weak hypotheses. A weak hypothesis is a function $\hat{\Phi}_s : D \times C \rightarrow \mathbb{R}$. We interpret the sign of $\hat{\Phi}_s(d_i, c_j)$ as the prediction of $\hat{\Phi}_s$ on whether d_i belongs to c_j, i.e. $\hat{\Phi}_s(d_i, c_j) > 0$ means that d_i is believed to belong to c_j while $\hat{\Phi}_s(d_i, c_j) < 0$ means it is believed not to belong to c_j. We instead interpret the absolute value of $\hat{\Phi}_s(d_i, c_j)$ (indicated by $|\hat{\Phi}_s(d_i, c_j)|$) as the strength of this belief.

At each iteration s AdaBoost.MH tests the effectiveness of the newly generated weak hypothesis $\hat{\Phi}_s$ on the training set and uses the results to update a distribution D_s of weights on the training pairs $\langle d_i, c_j\rangle$. The weight $D_{s+1}(d_i, c_j)$ is meant to capture how effective $\hat{\Phi}_1, \ldots, \hat{\Phi}_s$ have been in correctly predicting whether the training document d_i belongs to category c_j 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 $\hat{\Phi}_{s+1}$ 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 $D_s(d_i, c_j)$ are updated to $D_{s+1}(d_i, c_j)$ according to the rule

$$D_{s+1}(d_i, c_j) = \frac{D_s(d_i, c_j) \exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j))}{Z_s} \quad (1)$$

where

$$Z_s = \sum_{i=1}^{g} \sum_{j=1}^{m} D_s(d_i, c_j) \exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j)) \quad (2)$$

is a normalization factor chosen so that D_{s+1} is in fact a distribution, i.e. so that $\sum_{i=1}^{g} \sum_{j=1}^{m} D_{s+1}(d_i, c_j) = 1$. Equation 1 is such that the weight assigned to a pair $\langle d_i, c_j\rangle$ misclassified by $\hat{\Phi}_s$ is increased, as for such a pair $\Phi(d_i, c_j)$ and $\hat{\Phi}_s(d_i, c_j)$ have different signs and the factor $\Phi(d_i, c_j) \cdot \hat{\Phi}_s(d_i, c_j)$ is thus negative; likewise, the weight assigned to a pair correctly classified by $\hat{\Phi}_s$ is decreased.

2.1 Choosing the weak hypotheses

In AdaBoost.MH each document d_i is represented as a vector $\langle w_{1i}, \ldots, w_{ri}\rangle$ of r binary weights, where $w_{ki} = 1$ (resp. $w_{ki} = 0$) means that term t_k occurs (resp. does not occur) in d_i; $T = \{t_1, \ldots, t_r\}$ 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.

In AdaBoost.MH the weak hypotheses generated by the weak learner at iteration s are decision stumps of the form

$$\hat{\Phi}_s(d_i, c_j) = \begin{cases} a_{0j} & \text{if } w_{ki} = 0 \\ a_{1j} & \text{if } w_{ki} = 1 \end{cases} \quad (3)$$

where t_k (called the *pivot term* of $\hat{\Phi}_s$) belongs to $\{t_1, \ldots, t_r\}$, and a_{0j} and a_{1j} are real-valued constants. The choices for t_k, a_{0j} and a_{1j} are in general different for each iteration s, and are made according to an error-minimization policy described in the rest of this section.

Schapire and Singer [12] have proven that the *Hamming loss* of the final hypothesis $\hat{\Phi}$, defined as the percentage of pairs $\langle d_i, c_j\rangle$ for which $sign(\Phi(d_i, c_j)) \neq sign(\hat{\Phi}(d_i, c_j))$, is at most $\Pi_{s=1}^{S} Z_s$. The Hamming loss of a hypothesis is a measure of its classification (in)effectiveness; therefore, a reasonable (although suboptimal) way to maximize the effectiveness of the final hypothesis $\hat{\Phi}$ is to "greedily" choose each weak hypothesis $\hat{\Phi}_s$ (and thus its parameters t_k, a_{0j} and a_{1j}) in such a way as to minimize the normalization factor Z_s.

Schapire and Singer [13] 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 [13], 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 (from next section on simply called AdaBoost.MH) chooses weak hypotheses of the form described in Equation 3 by the following algorithm.

Algorithm 1 (The AdaBoost.MH weak learner).

1. *For each term $t_k \in \{t_1, \ldots, t_r\}$, select, among all the weak hypotheses $\hat{\Phi}$ that have t_k as the "pivot term", the one (indicated by $\hat{\Phi}_{best(k)}$) for which Z_s is minimum.*
2. *Among all the hypotheses $\hat{\Phi}_{best(1)}, \ldots, \hat{\Phi}_{best(r)}$ selected for the r different terms in Step 1, select the one (indicated by $\hat{\Phi}_s$) for which Z_s is minimum.*

Input: A training set $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_g, C_g\rangle\}$
where $C_i \subseteq C = \{c_1, \ldots, c_m\}$ for all $i = 1, \ldots, g$.

Body: Let $D_1(d_i, c_j) = \frac{1}{gm}$ for all $i = 1, \ldots, g$ and for all $j = 1, \ldots, m$
For $s = 1, \ldots, S$ do:

- pass distribution $D_s(d_i, c_j)$ to the weak learner;
- get the weak hypothesis $\hat{\Phi}_s$ from the weak learner;
- set $D_{s+1}(d_i, c_j) = \frac{D_s(d_i, c_j)\exp(-\Phi(d_i, c_j)\cdot\hat{\Phi}_s(d_i, c_j))}{Z_s}$
  where $Z_s = \sum_{i=1}^{g}\sum_{j=1}^{m} D_s(d_i, c_j)\exp(-\Phi(d_i, c_j)\cdot\hat{\Phi}_s(d_i, c_j))$
  is a normalization factor chosen so that $\sum_{i=1}^{g}\sum_{j=1}^{m} D_{s+1}(d_i, c_j) = 1$

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

Figure 1: The AdaBoost.MH algorithm.

Step 1 is clearly the key step, since there are a non-enumerable set of weak hypotheses with t_k as the pivot term. Schapire and Singer [12] have proven that, given term t_k and category c_j,

$$\hat{\Phi}_{best(k)}(d_i, c_j) = \begin{cases} \frac{1}{2}\ln\frac{W_{+1}^{0jk}}{W_{-1}^{0jk}} & \text{if } w_{ki} = 0 \\ \frac{1}{2}\ln\frac{W_{+1}^{1jk}}{W_{-1}^{1jk}} & \text{if } w_{ki} = 1 \end{cases} \quad (4)$$

where

$$W_b^{xjk} = \sum_{i=1}^{g} D_s(d_i, c_j)\cdot[\![w_{ki} = x]\!]\cdot[\![\Phi(d_i, c_j) = b]\!] \quad (5)$$

for $b \in \{-1, +1\}$, $x \in \{0, 1\}$, $j \in \{1, \ldots, m\}$ 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 a_{xj} we obtain

$$Z_s = 2\sum_{j=1}^{m}\sum_{x=0}^{1}(W_{+1}^{xjk}W_{-1}^{xjk})^{\frac{1}{2}} \quad (6)$$

Choosing $\frac{1}{2}\ln\frac{W_{+1}^{xjk}}{W_{-1}^{xjk}}$ as the value for a_{xj} has the effect that $\hat{\Phi}_s(d_i, c_j)$ outputs a positive real value in the two following cases:

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

In all the other cases $\hat{\Phi}_s$ outputs a negative real value. Here, "majority" has to be understood in a weighted sense, i.e. by bringing to bear the weight $D_s(d_i, c_j)$ associated to the training pair $\langle d_i, c_j\rangle$. The larger this majority is, the higher the absolute value of $\hat{\Phi}_s(d_i, c_j)$ is; this means that this absolute value represents a measure of the confidence that $\hat{\Phi}_s$ has in its own prediction [12].

In practice, the value $a_{xj} = \frac{1}{2}\ln\frac{W_{+1}^{xjk}+\epsilon}{W_{-1}^{xjk}+\epsilon}$ is chosen in place of $a_{xj} = \frac{1}{2}\ln\frac{W_{+1}^{xjk}}{W_{-1}^{xjk}}$, 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

$$\hat{\Phi}(d_i, c_j) = \sum_{s=1}^{S}\hat{\Phi}_s(d_i, c_j) \quad (7)$$

obtained by summing the outputs of the weak hypotheses.

2.2 Implementing AdaBoost.MH

Following [15], in our implementation of AdaBoost.MH we have further optimized the final hypothesis $\hat{\Phi}(d_i, c_j) = \sum_{s=1}^{S}\hat{\Phi}_s(d_i, c_j)$ by "compressing" the weak hypotheses $\hat{\Phi}_1, \ldots, \hat{\Phi}_S$ according to their pivot term t_k. In fact, note that if $\{\hat{\Phi}_1, \ldots, \hat{\Phi}_S\}$ contains a subset $\{\hat{\Phi}_1^{(k)}, \ldots, \hat{\Phi}_{q(k)}^{(k)}\}$ of weak hypotheses that all hinge on the same pivot term t_k and are of the form

$$\hat{\Phi}_r^{(k)}(d_i, c_j) = \begin{cases} a_{0j}^r & \text{if } w_{ki} = 0 \\ a_{1j}^r & \text{if } w_{ki} = 1 \end{cases} \quad (8)$$

[3] In [13] 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 $\epsilon = \frac{1}{gm}$, which we adopt in this work.

for $r = 1, \ldots, q(k)$, the collective contribution of $\hat{\Phi}_1^{(k)}, \ldots, \hat{\Phi}_{q(k)}^{(k)}$ to the final hypothesis is the same as that of a "combined hypothesis"

$$\hat{\Phi}^{(k)}(d_i, c_j) = \begin{cases} \sum_{r=1}^{q(k)} a_{0j}^r & \text{if } w_{ki} = 0 \\ \sum_{r=1}^{q(k)} a_{1j}^r & \text{if } w_{ki} = 1 \end{cases} \tag{9}$$

In the implementation we have thus replaced $\sum_{s=1}^{S} \hat{\Phi}_s(d_i, c_j)$ with $\sum_{k=1}^{\Delta} \hat{\Phi}^{(k)}(d_i, c_j)$, where Δ is the number of different terms that act as pivot for the weak hypotheses in $\{\hat{\Phi}_1, \ldots, \hat{\Phi}_S\}$.

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 on REUTERS-21578 with ADABOOST.MH when $S = 1000$ consists of 1000 weak hypotheses, but the number of different pivot terms is only 766 (see Section 5.2). The reduction in the size of the final hypothesis which derives from this modification is usually larger when high reduction factors have been applied in a feature selection phase, since in this case the number of different terms that can be chosen as the pivot is smaller.

3. MP-BOOST, AN IMPROVED BOOSTING ALGORITHM WITH MULTIPLE PIVOT TERMS

We here propose an improved version of ADABOOST.MH, that we call ADABOOST.MH *with multiple pivot terms* (here nicknamed MP-BOOST), that basically consists in modifying the form of weak hypotheses and how they are generated.

Looking at Equation 3 we may note that, at each iteration s, choosing a weak hypothesis means choosing (i) a pivot term t_k, *the same for all categories*, and (ii) for each category c_j, a pair of constants $\langle a_{0j}, a_{1j}\rangle$. We contend that the fact that, at iteration s, the same term t_k is chosen as the pivot term on which the binary classifiers for all categories hinge, is clearly suboptimal. At this iteration term t_k may be a very good discriminator for category c', but a very poor discriminator for category c'', which means that the weak hypothesis generated at this iteration would contribute very little to the correct classification of documents under c''. We claim that choosing, at every iteration s, a different pivot term $t_{\langle s,j\rangle}$ for each category c_j allows the weak hypothesis to provide customized, improved treatment to each individual category.

In MP-BOOST the weak hypotheses generated by the weak learner at iteration s are thus of the form

$$\hat{\Phi}_s(d_i, c_j) = \begin{cases} a_{0j} & \text{if } w_{\langle s,j\rangle i} = 0 \\ a_{1j} & \text{if } w_{\langle s,j\rangle i} = 1 \end{cases} \tag{10}$$

where term $t_{\langle s,j\rangle}$ is the pivot term chosen for category c_j at iteration s.

To see how MP-BOOST chooses weak hypotheses of the form described in Equation 10, let us first define a *weak c_j-hypothesis* as a function

$$\hat{\Phi}^j(d_i) = \begin{cases} a_{0j} & \text{if } w_{ki} = 0 \\ a_{1j} & \text{if } w_{ki} = 1 \end{cases} \tag{11}$$

that is only concerned with classifying documents under c_j; a weak hypothesis is the union of weak c_j-hypotheses, one for each category $c_j \in C$. At each iteration s, MP-BOOST chooses a weak hypothesis $\hat{\Phi}_s$ by means of the following variation of Algorithm 1.

ALGORITHM 2 (THE MP-BOOST WEAK LEARNER).

1. *For each category c_j and for each term $t_k \in \{t_1, \ldots, t_r\}$, select, among all weak c_j-hypothesis $\hat{\Phi}^j$ that have t_k as the pivot term, the one (indicated by $\hat{\Phi}^j_{best(k)}$) which minimizes*

$$Z_s^j = \sum_{i=1}^{g} D_s(d_i, c_j) \exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}^j(d_i)) \tag{12}$$

2. *For each category c_j, among all the hypotheses $\hat{\Phi}^j_{best(1)}$, ..., $\hat{\Phi}^j_{best(r)}$ selected in Step 1 for the r different terms, select the one (indicated by $\hat{\Phi}^j_s$) for which Z_s^j is minimum;*

3. *Choose, as the weak hypothesis $\hat{\Phi}_s$, the "union", across all $c_j \in C$, of the weak c_j-hypotheses selected in Step 2, i.e. the function such that $\hat{\Phi}_s(d_i, c_j) = \hat{\Phi}^j_s(d_i)$.*

Note the difference between Algorithm 1, as described in Section 2.1, and Algorithm 2; in particular, Step 2 of Algorithm 2 is such that weak c_j-hypotheses based on different pivot terms may be chosen for different categories c_j.

For reasons analogous to the ones discussed in Section 2.1, Step 1 is the key step; it is important to observe that $\hat{\Phi}^j_{best(k)}$ is still guaranteed to have the form described in Equation 4, since the weak hypothesis generated by Equation 10 is the same that Equation 3 generates when $m = 1$, i.e. when C contains one category only.

Note also that the "outer" algorithm of Figure 1 is untouched by our modifications, except for the fact that a normalization factor Z_s^j local to each category c_j is used (in place of the "global" normalization factor Z_s) in order to obtain the revised distribution D_{s+1}; i.e.

$$D_{s+1}(d_i, c_j) = \frac{D_s(d_i, c_j) \exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}^j(d_i))}{Z_s^j}$$

The main difference in the algorithm is thus in the "inner" part, i.e. in the weak hypotheses that are received from the weak learner, which now have the form of Equation 10, and in the way they are generated.

Concerning the optimizations discussed in Section 2.2, obtained by merging into a single weak hypothesis all weak hypotheses that share the same pivot term, note that in MP-BOOST these must be done on a category-by-category basis, i.e. by merging into a single weak c_j-hypothesis all weak c_j-hypotheses that share the same pivot term. The effect of this is that the different categories $c_1, \ldots, c_m$ may be associated to final hypotheses consisting of different numbers $\Delta_1, \ldots, \Delta_m$ of weak hypotheses.

Last, let us note that one consequence of switching from ADABOOST.MH to MP-BOOST is that *local* feature selection (i.e. choosing different reduced feature sets for different categories) can also be used in place of *global* feature selection (i.e. choosing the same reduced feature set for all categories). In fact, since in MP-Boost the choice of pivot terms is category-specific, the vectorial representations of documents can also be category-specific. This allows the designer to select, ahead of the learning phase and by means of standard feature selection techniques, the terms that are the most discriminative for a given category c_j, and are thus highly likely to be chosen as pivot terms for the c_j-hypotheses. This can be done separately for each individual category, and thus allows the use of even higher reduction factors; from the standpoint of efficiency this is advantageous, given that the computational cost of MP-Boost has a linear dependence on the number of features used (see Section 4).

4. THE COMPUTATIONAL COST OF MP-BOOST

We now analyse the computational costs of AdaBoost.MH and MP-Boost, and show that the latter has the same computational complexity of the former, despite generating (unlike AdaBoost.MH) classifiers that provide individualized treatment for each category. In Section 5.2 we also report and discuss actual running times recorded during the experiments.

Let us first discuss the cost of classifier training. The key steps of AdaBoost.MH are (i) computing, for each $t_k \in T$, the $Z_s(\hat{\Phi}_{best(k)})$ factor, and (ii) computing the minimum, over all t_k, of such $Z_s(\hat{\Phi}_{best(k)})$ factors. By inspecting Equations 5 and 6 we can clearly see that, for each t_k, Step (i) requires $O(gm)$ operations, where g is the number of training documents and m is the number of categories; since there are r such terms, the entire step requires $O(gmr)$ operations. Step (ii) requires $O(r)$ operations, so the entire AdaBoost.MH training process is $O(gmr)$. In the case of MP-Boost, the key steps are (i) computing, for each $t_k \in T$ and for each $c_j \in C$, the $Z_s^j(\hat{\Phi}_{best(k)})$ factor, and (ii) for each $c_j \in C$, computing the minimum, over all t_k, of such $Z_s^j(\hat{\Phi}_{best(k)})$ factors. This time, for each pair $\langle t_k, c_j \rangle$ Step (i) only requires $O(g)$ operations ; but since there are r terms and m categories, the entire step requires, again, $O(gmr)$ operations. Step (ii) requires $O(mr)$ operations, so the entire MP-Boost training process is, again, $O(gmr)$. This shows that, at training time, the computational costs of AdaBoost.MH and MP-Boost are of the same order of magnitude; this is confirmed by the experiments reported in Table 2.

At a first approximation, applying the final hypothesis generated by AdaBoost.MH to a test document consists in applying to it a committee of S decision stumps; the cost is thus $O(S)$. In the case of MP-Boost the cost is instead $O(mS)$ since there is a committee of S decision stumps for each of the m categories. In practice, since weak hypotheses are "compressed", as described in Section 2.2, for both learners the cost linearly depends on Δ, the number of distinct pivot terms selected during the training process (for MP-Boost, we take Δ to be an average of the category-specific Δ_i values). For a given value of S the value of Δ tends to be much smaller for MP-Boost than for AdaBoost.MH, since the "good" pivot terms for a specific category tend to be few. As a result, for the testing phase the differential in cost between the two algorithms is, in practice, much smaller than the upper bounds discussed above seem to suggest. Table 1 shows how, in our experiments, the value of Δ varies, for both learners, as S increases.

In terms of space, a weak hypothesis consists of 1 pivot term and $2m$ constants in AdaBoost.MH, and of m pivot terms and $2m$ constants in MP-Boost; the cost of storing the final hypothesis is thus $O(mS)$ (where S is the total number of boosting iterations) for both AdaBoost.MH and MP-Boost.

	Reuters-21578				RCV1-v2			
	ffs		rfs		ffs		rfs	
	AdaBoost.MH	MP-Boost	AdaBoost.MH	MP-Boost	AdaBoost.MH	MP-Boost	AdaBoost.MH	MP-Boost
5	5	4.5	5	4.5	5	4.9	5	4.9
10	10	8.4	10	8.4	10	9.7	10	9.7
20	20	15.0	20	15.1	20	19.1	20	19.2
50	50	30.6	50	31.3	50	46.2	50	46.6
100	98	50.1	98	51.1	95	88.6	95	89.0
200	192	79.8	192	81.4	191	163.7	191	165.4
500	433	136.6	433	134.7	479	338.9	483	344.0
1000	766	191.7	703	181.5	950	533.7	946	538.2

Table 1: Difference between the Δ values of AdaBoost.MH and MP-Boost; "ffs" and "rfs" stand for "full feature set" and "reduced feature set", respectively.

5. EXPERIMENTS

We have run a series of experiments for testing MP-Boost, using AdaBoost.MH as a baseline. Section 5.1 describes the setting of these experiments and Section 5.2 describes the results we have obtained.

5.1 Experimental setting

The corpora we have used in our experiments are Reuters-21578 and RCV1-v2.

The "Reuters-21578, Distribution 1.0" corpus is currently the most widely used benchmark in multi-label text categorization research[4]. It 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 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. In our experiments we have restricted our attention to the 115 categories with at least one positive training example.

The Reuters Corpus Volume 1 version 2 (RCV1-v2)[5] is a more recent text categorization benchmark made available

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

[5]`http://trec.nist.gov/data/reuters/reuters.html`

by Reuters and consisting of 804,414 news stories produced by Reuters from 20 Aug 1996 to 19 Aug 1997; all news stories are in English, and have 109 distinct terms per document on average [11]. In our experiments we have used the "LYRL2004" split, defined in [6], in which the (chronologically) first 23,149 documents are used for training and the other 781,265 are used for test. Out of the 103 "Topic" categories, in our experiments we have restricted our attention to the 101 categories with at least one positive training example.

In all the experiments discussed in this paper, stop words have been removed using the stop list provided in [5, pages 117–118], punctuation has been removed, all letters have been converted to lowercase, numbers have been removed, and stemming has been performed by means of Porter's stemmer. Feature selection has been performed by scoring features by means of *information gain*, defined as

$$IG(t_k, c_i) = \sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$$

The final set of features has been chosen according to Forman's *round robin* technique, which consists in picking, for each category c_i, the v features with the highest $IG(t_k, c_i)$ value, and pooling all of them together into a category-independent set [2]. This set thus contains at most vm features, where m is the number of categories; it usually contains strictly fewer than vm features, since some features are among the best v features for more than one category. We have set v to 48 (for Reuters-21578) and 177 (for RCV1-v2); these are the values that bring about feature set sizes of 2,012 (Reuters-21578) and 5,509 (RCV1-v2), thus achieving 90% reduction with respect to the original sets which consisted of of 20,123 (Reuters-21578) and 55,051 (RCV1-v2) terms.

As a measure of effectiveness that combines the contributions of *precision* (π) and *recall* (ρ) we have used the well-known F_1 function, defined as

$$F_1 = \frac{2\pi\rho}{\pi + \rho} = \frac{2TP}{2TP + FP + FN} \qquad (13)$$

and which corresponds to the harmonic mean of precision and recall (here, TP stands for true positives, FP for false positives, and FN for false negatives). Note that F_1 is undefined when $TP = FP = FN = 0$; in this case we take F_1 to equal 1.0, since the classifier has correctly classified all documents as negative examples.

We compute both microaveraged F_1 (denoted by F_1^{μ}) and macroaveraged F_1 (F_1^M). F_1^{μ} is obtained by (i) computing the category-specific values TP_i, (ii) obtaining TP as the sum of the TP_i's (same for FP and FN), and then (iii) applying Equation 13. F_1^M is obtained by first computing the F_1 values specific to the individual categories, and then averaging them across the c_i's.

5.2 Results

The results of our experiments are reported in Table 2 for some key values of the number of iterations S; Figure 2 reports the same results in graphical form for any value of S comprised in the [1..1000] interval. It is immediate to observe that, for any value of S, MP-Boost always improves on AdaBoost.MH, in terms of both F_1^{μ} and F_1^M.

Let us discuss the results obtained on Reuters-21578 (the ones obtained on RCV1-v2 are qualitatively similar). For small values of S the improvement in effectiveness of MP-Boost wrt AdaBoost is spectacular: F_1^{μ} goes up by +69.47% for $S = 5$, by +57.07% for $S = 10$, and by +30.07% for $S = 20$. As the value of S grows, the margin between the two learners narrows: we obtain +4.34% for $S = 1,000$ and +4.20% for $S = 10,000$. This fact may be explained by noting that in AdaBoost.MH, if the final hypothesis consists of a few weak hypotheses only, it is likely that only few categories have been properly addressed (i.e. that the pivot terms used in the committee have a high discrimination power for few categories only). When the number of weak hypotheses gets larger, it is more likely that many (or most of the) categories have been properly catered for. Conversely, MP-Boost has already used the best pivot terms for each category right from the very first iterations; this explains the fact that MP-Boost is highly effective even for small values of S.

Note that the improvement brought about by the individualized treatment of categories implemented by MP-Boost is not recovered by AdaBoost.MH even by pushing S to the limit. For instance, note that not even in 10,000 iterations AdaBoost.MH manages to obtain the F_1^{μ} values obtained by MP-Boost in just 50 iterations: MP-Boost with $S = 50$ obtains a slightly superior effectiveness (+1.4%) than AdaBoost.MH with $S = 10,000$, in less than 1% the training time and in about 10% the testing time of this latter.

These effectiveness improvements are even more significant when considering macroaveraged effectiveness. In this case, we obtain a relative improvement in F_1^M that ranges from a minimum of +21.13% (obtained for $S = 10,000$) to a maximum of +124,70% (obtained for $S = 5$). Again, not even in 10,000 iterations AdaBoost.MH obtains the F_1^M values obtained by MP-Boost in just 5 iterations. This may be explained by recalling the well-known fact that macroaveraged effectiveness especially rewards those classifiers that perform well also on infrequent categories (i.e. categories with few positive training examples); indeed, unlike AdaBoost.MH, MP-Boost places equal emphasis on all categories, regardless of their frequency, thus picking the very best pivot terms for the infrequent categories too right from the first iterations.

Let us now discuss the relative efficiency of the two learners. As expected, for both learners the time required to generate the final committees grows linearly with the number of boosting iterations S. We also observed an almost constant ratio between the running times of the two learners, with MP-Boost being about 9% slower than AdaBoost.MH. A profiling session on the applications has pointed out that this difference is due to the larger (by a constant factor) size of weak hypotheses in MP-Boost (see Section 4), which generates a small overhead in memory management. In terms of testing time, instead, it turns out that MP-Boost is, for equal numbers S of boosting iterations, from one to four times slower than AdaBoost.MH (see Table 2).

		AdaBoost.MH				MP-Boost				MP-Boost wrt AdaBoost.MH			
	S	F_1^μ	F_1^M	$\tau(Tr)$	$\tau(Te)$	F_1^μ	F_1^M	$\tau(Tr)$	$\tau(Te)$	F_1^μ	F_1^M	$\tau(Tr)$	$\tau(Te)$
Reuters-21578 full feature set	5	0.416	0.235	12.1	0.1	0.704	0.529	13.2	0.2	+69.47%	+124.70%	+9.09%	+100.0%
	10	0.483	0.271	24.2	0.1	0.759	0.556	26.4	0.3	+57.07%	+105.52%	+9.09%	+183.3%
	20	0.611	0.325	48.4	0.1	0.795	0.586	52.8	0.5	+30.07%	+80.44%	+9.09%	+266.6%
	50	0.723	0.392	96.8	0.2	0.822	0.589	105.7	1.1	+13.79%	+50.44%	+9.19%	+324.0%
	100	0.776	0.454	193.6	0.4	0.837	0.608	211.3	1.7	+7.91%	+34.06%	+9.14%	+326.8%
	200	0.798	0.461	387.1	0.8	0.843	0.600	422.7	3.1	+5.68%	+30.16%	+9.20%	+297.4%
	500	0.811	0.485	774.2	2.0	0.848	0.604	845.3	6.3	+4.51%	+24.62%	+9.18%	+216.1%
	1000	0.811	0.482	1548.4	3.7	0.846	0.603	1690.6	9.2	+4.34%	+25.06%	+9.18%	+150.1%
	10000	0.810	0.497	15483.9	10.0	0.844	0.602	16906.2	20.6	+4.20%	+21.13%	+9.18%	+106.0%
RCV1-v2 full feature set	5	0.361	0.037	34.5	21.8	0.519	0.306	37.3	54.0	+43.89%	+720.57%	+8.12%	+147.9%
	10	0.406	0.070	69.1	25.5	0.588	0.367	74.7	91.8	+44.80%	+421.88%	+8.10%	+260.7%
	20	0.479	0.131	138.1	32.7	0.646	0.418	149.4	148.5	+34.96%	+218.09%	+8.18%	+354.7%
	50	0.587	0.239	276.2	54.6	0.700	0.455	298.7	286.2	+19.24%	+90.63%	+8.15%	+423.8%
	100	0.650	0.333	552.4	87.5	0.726	0.474	597.5	472.5	+11.75%	+42.33%	+8.16%	+439.8%
	200	0.701	0.396	1104.8	161.5	0.745	0.487	1194.9	837.0	+6.20%	+23.00%	+8.16%	+418.3%
	500	0.735	0.435	2209.7	516.1	0.761	0.495	2389.9	1698.3	+3.58%	+13.74%	+8.15%	+229.1%
	1000	0.745	0.442	4419.3	1014.4	0.768	0.496	4779.7	2478.6	+2.99%	+12.21%	+8.16%	+144.4%
	10000	0.754	0.459	44192.3	2831.4	0.765	0.485	47796.2	5772.4	+1.46%	+5.66%	+8.16%	+103.9%
Reuters-21578 red. feature set	5	0.416	0.235	9.3	0.1	0.704	0.515	10.2	0.2	+69.23%	+119.15%	+9.68%	+133.3%
	10	0.483	0.271	18.5	0.1	0.760	0.560	20.4	0.3	+57.35%	+106.64%	+10.27%	+200.0%
	20	0.611	0.325	37.1	0.1	0.794	0.567	40.7	0.5	+29.95%	+74.46%	+9.70%	+307.7%
	50	0.723	0.392	74.1	0.2	0.826	0.596	81.4	1.0	+14.25%	+52.04%	+9.85%	+325.0%
	100	0.773	0.457	148.3	0.4	0.839	0.614	162.9	1.7	+8.54%	+34.35%	+9.84%	+315.0%
	200	0.790	0.474	296.5	0.7	0.845	0.623	325.8	2.9	+6.96%	+31.43%	+9.88%	+288.0%
	500	0.811	0.485	593.0	1.9	0.846	0.617	651.5	5.8	+4.32%	+27.22%	+9.87%	+202.1%
	1000	0.806	0.484	1186.0	3.2	0.839	0.619	1303.0	8.2	+4.09%	+27.89%	+9.87%	+153.2%
RCV1-v2 red. feature set	5	0.361	0.037	28.2	21.1	0.519	0.307	30.5	49.6	+43.77%	+729.73%	+8.16%	+135.6%
	10	0.406	0.070	56.4	24.4	0.587	0.365	61.0	78.0	+44.58%	+421.43%	+8.16%	+219.3%
	20	0.479	0.131	112.7	31.2	0.646	0.416	122.1	125.6	+34.86%	+217.56%	+8.34%	+302.1%
	50	0.587	0.239	225.4	54.6	0.701	0.458	244.2	247.4	+19.42%	+91.63%	+8.34%	+352.8%
	100	0.650	0.333	450.9	84.6	0.727	0.478	488.4	442.3	+11.85%	+43.54%	+8.32%	+422.7%
	200	0.701	0.396	901.8	154.4	0.744	0.493	976.8	896.9	+6.13%	+24.49%	+8.32%	+481.0%
	500	0.734	0.431	1803.5	495.9	0.760	0.503	1953.5	2133.1	+3.54%	+16.71%	+8.32%	+330.2%
	1000	0.747	0.445	3607.0	974.7	0.764	0.505	3907.0	3500.6	+2.28%	+13.48%	+8.32%	+259.2%

Table 2: Comparative performance of AdaBoost.MH and MP-Boost on the Reuters-21578 and RCV1-v2 benchmarks, with (i) a full feature set and with (ii) a reduced feature set obtained with a round-robin technique and 90% reduction factor. S indicates the number of boosting iterations; F_1^μ and F_1^M indicate micro- and macro-averaged F_1, respectively; $\tau(Tr)$ and $\tau(Te)$ indicate the time (in seconds) required for training and testing, respectively.

This is due to the fact that AdaBoost.MH selects, for the same value S, a number Δ of distinct pivot terms smaller than the number $\sum_{i=1}^{m} \Delta_i$ that MP-Boost selects (see Section 2.2), and to the fact that the classifier tests all the values of these terms in the document. However, note that for MP-Boost this loss in testing efficiency is more than compensated by the large gain in effectiveness. Also, with MP-Boost trained on the full feature set with $S = 1000$ (a value at which effectiveness peaks) the time required for classifying all the 781,265 RCV1-v2 test documents is about 79 minutes, which is more than acceptable.

Last, let us note that the experiments run with the reduced feature set (see Table 2) have produced practically unchanged effectiveness results wrt those obtained with the full feature set, but (as expected – see Section 4) at the advantage of dramatically smaller training times and substantially smaller testing times. That feature selection does not reduce effectiveness might seem surprising in the context of a boosting algorithm, since feature selection brings about smaller degrees of freedom in the choice of the best pivot term; quite evidently, IG is very effective at discarding the terms that the boosting algorithm would not choose anyway as pivots.

6. CONCLUSION

We have presented MP-Boost, a novel algorithm for multi-label text categorization that improves upon the well-known AdaBoost.MH algorithm by selecting multiple pivot terms at each boosting iteration, we have provided (training time and testing time) complexity results for it, and we have thoroughly tested it on two well-known benchmarks, one of which consisting of more than 800,000 documents. The results allow us to conclude that MP-Boost is a largely superior alternative to AdaBoost.MH since, at the price of a tolerable decrease in classification efficiency, it yields speedier convergence, superior microaveraged effectiveness, and dramatically superior macroaveraged effectiveness. This latter fact makes it especially suitable to categorization problems in which the distribution of training examples across the categories is highly skewed.

7. REFERENCES

[1] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval*, pages 182–189, Toronto, CA, 2003.

[2] G. Forman. A pitfall and solution in multi-class feature selection for text classification. In *Proceedings of ICML-04,*

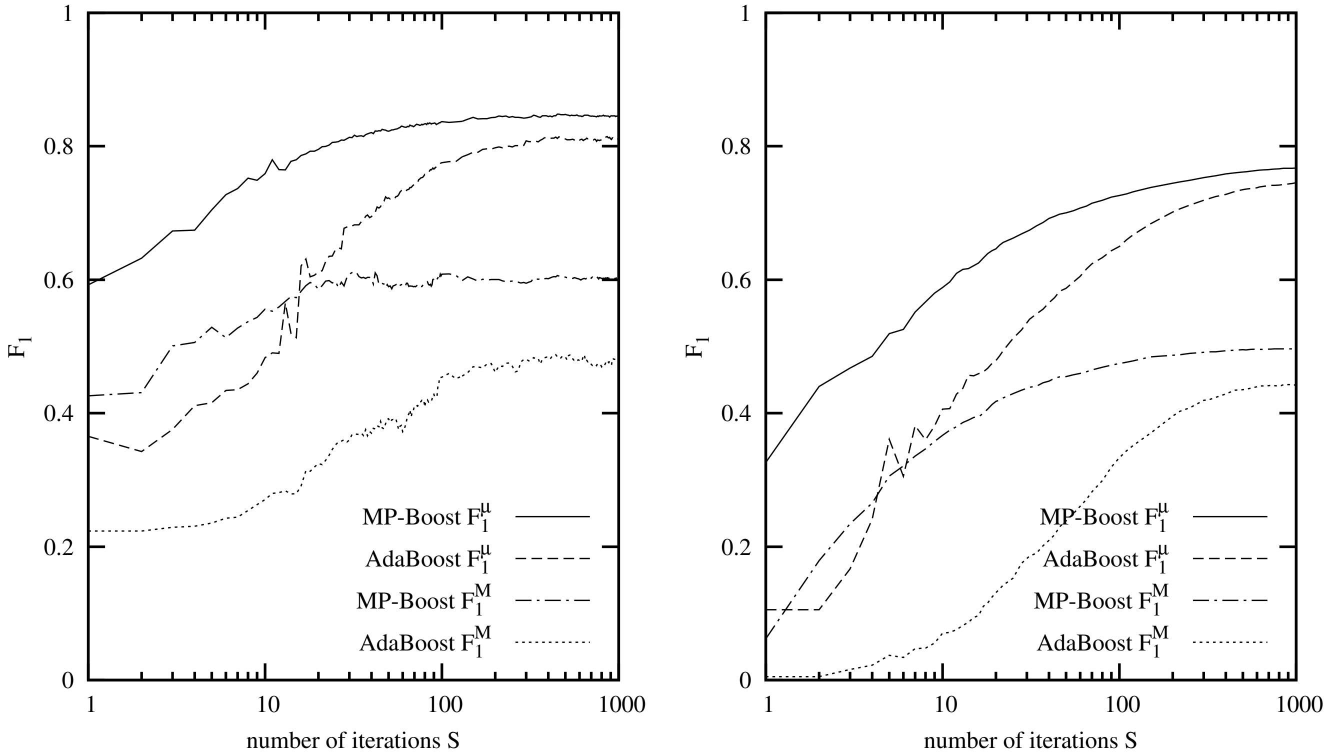

Figure 2: Effectiveness of AdaBoost.MH and MP-Boost on Reuters-21578 (left) and RCV1-v2 (right) as a function of the number S of iterations. The X axis is displayed on a logarithmic scale.

21st International Conference on Machine Learning, Banff, CA, 2004.

[3] R. D. Iyer, D. D. Lewis, R. E. Schapire, Y. Singer, and A. Singhal. Boosting for document routing. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 70–77, McLean, US, 2000.

[4] Y.-H. Kim, S.-Y. Hahn, and B.-T. Zhang. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 168–175, Athens, GR, 2000.

[5] D. D. Lewis. *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

[6] D. D. Lewis, F. Li, T. Rose, and Y. Yang. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, April 2004.

[7] Y. Liu, Y. Yang, and J. Carbonell. Boosting to correct the inductive bias for text classification. In *Proceedings of CIKM-02, 11th ACM International Conference on Information and Knowledge Management*, pages 348–355, McLean, US, 2002.

[8] R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. J. Smola, editors, *Advanced lectures on machine learning*, pages 118–183. Springer Verlag, Heidelberg, DE, 2003.

[9] K. Myers, M. Kearns, S. Singh, and M. A. Walker. A boosting approach to topic spotting on subdialogues. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 655–662, Stanford, US, 2000.

[10] P. Nardiello, F. Sebastiani, and A. Sperduti. Discretizing continuous attributes in AdaBoost for text categorization. In *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, pages 320–334, Pisa, IT, 2003.

[11] T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1: From yesterday's news to tomorrow's language resources. In *Proceedings of LREC-02, 3rd International Conference on Language Resources and Evaluation*, pages 827–832, Las Palmas, ES, 2002.

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

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

[14] 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.

[15] F. Sebastiani, A. Sperduti, and N. Valdambrini. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, McLean, US, 2000.

[16] H. Taira and M. Haruno. Text categorization using transductive boosting. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, pages 454–465, Freiburg, DE, 2001.