An Improved Boosting Algorithm and its Application to Text Categorization

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

Alessandro Sperduti
Dipartimento di Informatica
Università di Pisa
56125 Pisa, Italy
E-mail: `perso@di.unipi.it`

Nicola Valdambrini*
Dipartimento di Informatica
Università di Pisa
56125 Pisa, Italy
E-mail: `valdambr@cli.di.unipi.it`

May 3, 2000

Abstract

Boosting is a classification method which has successfully been applied to many different domains, and that has proven one of the best performers in text categorization (TC) 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 classification of the training documents which have proven harder for the previously trained classifiers. In this paper we describe an improved boosting algorithm, called AdaBoost.MH^{RK}, and its application to text categorization. AdaBoost.MH^{RK} 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 that, at this 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^{RK} is both more efficient to train and more effective than the original AdaBoost.MH^{R} boosting algorithm.

1 Introduction

Text categorization (TC – also known as *text classification*, or *topic spotting*) is the activity of automatically building, by means of machine learning (ML) techniques,

* Corresponding author.

automatic text classifiers, i.e. programs capable of labelling natural language texts with thematic categories from a predefined set $C = \{c_1, \ldots, c_m\}$. TC has many applications, ranging from automated document indexing for Boolean information retrieval systems [6] to document filtering [1], document organization for supporting information retrieval [15], resolution of natural language ambiguities [24], author identification for documents of unknown or controversial paternity [3], and even automatic essay grading [14].

The construction of an automatic text classifier requires the availability of a corpus $Co = \{\langle d'_1, C_1\rangle, \ldots, \langle d'_s, C_s\rangle\}$ 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* $Te = \{\langle d'_{g+1}, C_{g+1}\rangle, \ldots, \langle d'_s, C_s\rangle\} = Co - Tr$ and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the corpus[2]. This method has many advantages over the manual approach of building text classifiers by means of knowledge engineering techniques [8]. These advantages include greatly reduced effort in the construction of the classifier, in its updating to reflect possible shifts in the meaning of the categories, and in its porting to new category sets.

A wealth of different ML methods have been applied to TC [31], including probabilistic classifiers [19], decision trees [20], decision rules [2], regression methods [7], batch [10] and incremental linear methods [21], neural networks [29], example-based methods [33], and support vector machines [12]. 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, k independent experts may be better than one if their individual judgments are appropriately combined. In TC, the idea is to apply k 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) [30], or in terms of the ML method by means of which they have been built [9, 16, 22, 23], or both. A classifier committee is then characterised by (i) a choice of k classifiers, and (ii) a choice of a combination function.

The *boosting* method [4, 5, 25, 26, 27] occupies a special place in the classifier committees literature, since the k classifiers $\Phi_1, \ldots, \Phi_k$ (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 k weak hypotheses should be trained not in a conceptually parallel and independent way, as in the other classifier 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

[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 we use for our experiments. All the techniques we discuss in this paper can be straightforwardly adapted to the other case in which each document belongs to exactly one category.

al. [28] have shown the AdaBoost.MH boosting algorithm (an adaptation of Freund and Schapire's [5] AdaBoost algorithm 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 [2]. Further experiments by Schapire and Singer [27] showed AdaBoost.MH to outperform, aside from Sleeping Experts, a Naïve Bayes classifier, a standard Rocchio classifier, and Joachims' PrTFIDF classifier [11]. Weiss et al. [32] 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 [18]. Boosting has also been used in [22], 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 [12], the currently best performers in the TC arena (see [31, 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^{RK}, and describe the experimental results we have obtained on the Reuters-21578 text categorization benchmark. AdaBoost.MH^{RK} is based on a different method to create weak hypotheses. At each iteration, AdaBoost.MH^{RK} outputs not a single hypothesis, bu a sub-committee of the k hypotheses that, at this 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 5 briefly discusses our parallel implementations of AdaBoost.MH and AdaBoost.MH^{RK}. Section 6 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 Table 1) is a boosting algorithm proposed by Schapire and Singer [27] for the text categorization task and derived from AdaBoost, Freund and Schapire's [5] general purpose boosting algorithm. 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 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 $\Phi_s : \mathcal{D} \times C \rightarrow \Re$, 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 not. We instead interpret the absolute value of $\Phi_s(d_j, c_i)$ (indicated as $|\Phi_s(d_j, c_i)|$) 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 pairs $\langle d'_j, c_i\rangle$. The weight $D_{s+1}(d'_j, c_i)$ is meant to capture how difficult it was for $\Phi_1, \ldots, \Phi_s$ to correctly decide whether the training document d'_j belongs to the category c_i or not. By passing (together with the training set Tr)

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 $D_1(d'_j, c_i) = \frac{1}{mg}$ 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 $\alpha_s \in \Re$;
- set $D_{s+1}(d'_j, c_i) = \frac{D_s(d'_j, c_i)\exp(-\alpha_s \cdot C_j[c_i] \cdot \Phi_s(d'_j, c_i))}{Z_s}$
  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 $\Phi(d, c) = \sum_{s=1}^{S} \alpha_s \Phi_s(d, c)$

Table 1: The AdaBoost.MH algorithm.

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

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

where $C_j[c_i]$ is defined to be 1 if $c_i \in C_j$ and -1 otherwise, and Z_s is a normalization constant (see Table 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 misclassified $\langle d'_j, c_i\rangle$ pair is increased, as for such a pair $C_j[c_i]$ and $\Phi_s(d'_j, c_i)$ have the same sign and the factor $C_j[c_i] \cdot \Phi_s(d'_j, c_i)$ is thus positive; likewise, the weight assigned to a correctly classified pair 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 have the form

$$\Phi_s(d'_j, c_i) = \begin{cases} c_{0i} & \text{if } w_{kj} = 0 \\ c_{1i} & \text{if } w_{kj} = 1 \end{cases} \tag{2}$$

where t_k is a term chosen from $\{t_1, \ldots, t_r\}$ at iteration s, and c_{0i} and c_{1i} are numerical constants. Schapire and Singer [26] 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 thus a measure of its classification effectiveness. Therefore, in order to minimize the error of the final hypothesis Φ, each weak hypothesis Φ_s (and thus its parameters t_k, c_{0i} and c_{1i}) and each parameter α_s need to be chosen in such a way as to minimize the normalization factor Z_s. Schapire and Singer [27] define three different methods for choosing weak hypotheses for AdaBoost.MH, called

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 [27], 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 methods. AdaBoost.MH^R chooses weak hypotheses of the form described in Equation 2 by a two step-process:

1. for each term $t_k \in \{t_1, \ldots, t_r\}$ it pre-selects, among all weak hypotheses $\Phi_1^k, \ldots, \Phi_s^k$ that hinge on t_k, the one (Φ_{best}^k) for which Z_s is minimum;
2. among all the hypotheses $\Phi_{best}^1, \ldots, \Phi_{best}^r$ pre-selected for the different terms, it selects the one (Φ_{best}^k) for which Z_s is minimum.

Concerning Step 1, Schapire and Singer [26] have proven that, given term t_k and category c_i, Φ_{best}^k is obtained when $\alpha_t = 1$ and $c_{xi} = \frac{1}{2} \ln \left(\frac{W_1^{xik}}{W_{-1}^{xik}}\right)$, where

$$W_b^{xik} = \sum_{j=1}^{g} D_t(d'_j, c_i) \cdot [\![w_{kj} = x]\!] \cdot [\![C_j[c_i] = b]\!] \quad (3)$$

for $i \in \{1, \ldots, m\}$, $x \in \{0, 1\}$ 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, we obtain

$$Z_s = 2 \sum_{i=1}^{m} \sum_{x=0}^{1} (W_1^{xik} W_{-1}^{xik})^{\frac{1}{2}} \quad (4)$$

Choosing $\frac{1}{2} \ln \left(\frac{W_1^{xik}}{W_{-1}^{xik}}\right)$ as the value for c_{xi} has the effect that $\Phi_s(d_j, c_i)$ ouputs a positive 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 ouputs a negative real value. Here, "majority" has to be understood in a weighted sense, i.e. remembering that each pair $\langle d_j, c_i\rangle$ has an associated weight $D_t(d_j, c_i)$. The larger this majority is, the higher the absolute value of the output is; this means that the absolute value of the output of $\Phi_s(d_j, c_i)$ represents a measure of the confidence that Φ_s has in its decision. In practice,

Table 2: In each plot, terms are ranked on the X axis to obtain ascending curves. These plots show how the term scores asymptotically tend to 1 as the number of iterations grows.

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.

The output of the final hypothesis is the value $\Phi(d_j, c_i) = \sum_{s=1}^{S} \alpha_s \Phi_s(d_j, c_i)$ obtained by summing the outputs of the weak hypotheses.

3 An improved boosting algorithm and its application to text categorization

While the AdaBoost.MH^R algorithm is currently at the top of the text classification methods in terms of performance, it is possible to improve it by modifying the choice of the weak hypotheses. We here propose a new method, called AdaBoost.MH$^R K$, to choose the hypotheses at each iteration of the boosting process, a method which is based on the construction of a small committee of weak hypotheses. Instead of calculating the function Φ_s according to Equation 2 by using the term t_k that minimizes Z_s, we take into account the k most important terms, i.e. the k terms for which Z_s is smaller. The value of k depends on the iteration we are at. Once we have calculated the k hypotheses that correspond to these terms, we update the distribution and carry on as usual. The difference with the original algorithm is that, as a contribution of each iteration, we can add to the final hypothesis the information deriving from more than one term. In this way the computational cost to build a final hypothesis made up of S weak hypotheses depends on the value of k. Anyway, if k is not too high, the time spent to perform one iteration is comparable to the time spent to perform one iteration in the original AdaBoost.MH^R algorithm.

The idea this new method relies on comes from the observation of the term scores obtained during different iterations of AdaBoost.MH^R. In this latter algorithm it can be noted that, while in the first iterations only a few terms obtain high scores, so that it is evident which of them to choose, in the other cases the scores are very close, and they tend to become equal to 1 (see Table 2). Therefore the score Z_s does not seem to discriminate well among different terms, and this is the reason why we choose the first k terms which have similar scores and which would probably have been choosen in the next k iterations by the AdaBoost.MH algorithm. By setting the right value for k on each iteration, we can build a final hypothesis composed by a lot of weak hypotheses performing a reduced number of computations. In fact, most of the computation time of the boosting process is spent to calculate the terms score. Building a committee of k hypotheses on each iteration we compute k scores using the same distribution on the training document with the same cost incurred by AdaBoost.MH to find only the first term.

The committee of k hypotheses is used to update the distribution on the training examples by making a geometrical mean of their outputs.

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 [18], as it is currently the most widely used benchmark in text categorization research[3]. Reuters-21578 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 Reuters-21578 experimentation.

In all the experiments discussed in this section, stop words have been removed using the stop list provided in [17, pages 117–118]. No stemming and number removal have been performed. Term weighting has been obtained by means of the standard "ltc" variant of the *tfidf* function, i.e.

$$tfidf(t_s, d_j) \quad = tf(t_s, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_s)}$$

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

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

where $\#(t_s, d_j)$ denotes the number of times t_s occurs in d_j. Weights have been further normalized by cosine normalization, i.e.

$$w_{sj} \quad = \quad \frac{tfidf(t_s, d_j)}{\sqrt{\sum_{s=1}^{r'} tfidf(t_s, d_j)^2}}$$

where r' is the set of terms resulting from feature selection (feature selection, when performed, obviously takes place before weighting).

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. *Precision wrt* c_i (Pr_i) is defined as the probability that if a random document d_x is categorized under c_i, this decision is correct (i.e. it is a *true positive* for c_i). In what follows, TP, TN, FP and FN will denote the numbers of true positives, true negatives, false positives, and false negatives, respectively. *Recall wrt* c_i (Re_i) is instead defined as the probability that, if a random document d_x ought to be categorized under c_i, this decision is taken. Estimates of Pr_i and Re_i (indicated by $\hat{Pr}_i$ and $\hat{Re}_i$) may be obtained in the obvious way by counting occurrences on the test set. These category-relative values may in turn be averaged to obtain $\hat{Pr}$ and $\hat{Re}$, i.e. values global to the whole category set C, according to two alternative methods:

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

- *microaveraging* (indicated by the "μ" superscript): $\hat{Pr}$ and $\hat{Re}$ are obtained by globally summing over all individual decisions, i.e.:

$$\hat{Pr}^{\mu} = \frac{TP}{TP+FP} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i+FP_i)}$$
$$\hat{Re}^{\mu} = \frac{TP}{TP+FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i+FN_i)}$$

- *macroaveraging* (indicated by the "M" superscript): precision and recall are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories, i.e.:

$$\hat{Pr}^{M} = \frac{\sum_{i=1}^{m} Pr_i}{m} = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i+FP_i}}{m}$$
$$\hat{Re}^{M} = \frac{\sum_{i=1}^{m} Re_i}{m} = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i+FN_i}}{m}$$

In our experiments we have evaluated both microaveraged and macroaveraged precision and recall.

As a measure of effectiveness that combines the contributions of both $\hat{Pr}$ and $\hat{Re}$, we have used the well-known F_β function, defined as

$$F_\beta = \frac{(\beta^2+1)\cdot \hat{Pr}\cdot \hat{Re}}{\beta^2\cdot \hat{Pr}+\hat{Re}}$$

with $0 \leq \beta \leq +\infty$. Similarly to most other researchers we have used the parameter value $\beta = 1$, which places equal emphasis on $\hat{Pr}$ and $\hat{Re}$.

4.2 The experiments

Our experiments were conducted by varing some training parameters like the number of boosting iterations and the updating rule for k. We applied a 96% term space reduction to the training and test documents. This means that only the 4% (i.e. about 1000) of the terms that occur in the original documents were considered. Fore feature selection we used a "filtering" approach [13] and used χ^2 as our feature selection function, since it is known from the literature to be one of the best performing at aggressive reduction levels [34]. Table 3 summarizes the F_1 values we have obtained in correspondence of different numbers of boosting iterations and different updating rules for k. The updating rules we used add a constant C to k every N iterations, with a fixed value for N. This rule should instead be discussed in Section 3!!!!

We ran the AdaBoost.MH algorithm using the same data sets for the training and the test procedures. The results are shown in table 4.

During the first 500 iterations we can see that the new version of AdaBoost.MH behaves better than the original algorithm in the micro averaging case. In particular we achieve $F_1 = 0.743$ after only 50 iterations using $C = 1$ and $N = 12$. The time spent to do it were approximately the same spent by AdaBoost.MH to create a final hypothesis made up of only 50 weak hypotheses with $F_1 = 0.656$. With AdaBoost.MH we could not obtain $F_1 \geq 0.743$ in the first 16000 iterations.

We should remark that comparisons between these results and the results of [27] are difficult for at least two reasons: a) [27] uses a different variant of the Reuters

# of iterations	C	N	MicroAvg F_1	MacroAvg F_1
50	0	-	0.654655	0.497821
50	1	5	0.666667	0.513684
97	1	10	0.725000	0.565000
50	1	10	0.731658	0.549577
50	2	10	0.689655	0.512637
50	1	11	0.716192	0.565011
50	1	12	0.726818	0.558150
50	1	13	0.727083	0.550834
50	1	14	0.723666	0.540245
50	1	15	0.724629	0.532600
50	1	20	0.704566	0.511514
50	2	20	0.721946	0.531515
50	4	20	0.702740	0.515649
100	2	20	0.694929	0.544996
100	1	20	0.740828	0.557656
197	1	20	0.728946	0.572078
100	1	30	0.738643	0.558622
100	2	30	0.735571	0.555456
500	1	80	0.724133	0.591866
500	1	200	0.738584	0.570910

Table 3: Results obtained using AdaBoost.MH^{RK} with a rate of 96% for TSR applied with the χ^2_{max} method. We add C to the number k of hypotheses to generate on each iteration every N iterations.

# of iterations	MicroAvg F_1	MacroAvg F_1
30	0.597000	0.486978
50	0.656354	0.483030
100	0.697740	0.523138
167	0.702300	0.786115
500	0.724986	0.565490
760	0.726772	0.575426
1000	0.728390	0.584169
2000	0.725088	0.597137
3000	0.721169	0.605509
5000	0.722731	0.618689
6000	0.721077	0.621494
7000	0.720186	0.623543
8000	0.717481	0.619853
9000	0.717524	0.621415
10000	0.716788	0.622729
11000	0.715583	0.621209
12000	0.715418	0.622485
13000	0.714828	0.622671
14000	0.714640	0.622056
15000	0.714380	0.619946
16000	0.713720	0.619019

Table 4: Results obtained using the AdaBoost.MH algorithm. We set the TSR rate to 96% and we applied the χ^2_{max} method.

# of iterations	C	N	MicroAvg F_1	MacroAvg F_1
50	1	12	0.726213	0.511327
50	1	13	0.728315	0.525439
50	1	15	0.723751	0.525064
50	1	16	0.717628	0.527743
100	1	30	0.752109	0.532758
200	1	20	0.764550	0.558366
500	1	80	0.752034	0.568556
500	1	200	0.757198	0.533829

Table 5: Results obtained using AdaBoost.MH^{RK} with a rate of 90% for TSR applied with the χ^2_{max} method. We add C to the number k of hypotheses to generate on each iteration every N iterations.

# of iterations	MicroAvg F_1	MacroAvg F_1
50	0.652111	0.458944
100	0.708171	0.484279
200	0.737705	0.508982
500	0.753718	0.540283

Table 6: Results obtained using the AdaBoost.MH algorithm. We set the TSR rate to 90% and we applied the χ^2_{max} method.

collection, called Reuters-21450; b) [27] uses also bigrams, apart from single words, as terms, while we uses unigrams only.

We can therefore say that the new version of the AdaBoost.MH can achieve better results in less time. It must be noted that our experiments were conducted with a very high rate for the term space reduction process. Higher values for the F_1 measure can be obtained considering more terms.

5 A parallel implementation of AdaBoost.MH andAdaBoost.MH^{RK}

6 Conclusion

Further researches me might do are

- Check if the bug that Galavotti has found in the implementation of the χ^2 formula

# of iterations	C	N	MicroAvg F_1	MacroAvg F_1
50	1	12	0.724138	0.514714
50	1	13	0.722528	0.532095
50	1	15	0.723751	0.525064
100	1	30	0.740409	0.522325
200	1	20	0.764550	0.558366
500	1	200	0.750773	0.519780

Table 7: Results obtained using AdaBoost.MH^{RK} without the TSR process. We add C to the number k of hypotheses to generate on each iteration every N iterations.

# of iterations	MicroAvg F_1	MacroAvg F_1
50	0.643192	0.460704
100	0.704545	0.521491
200	0.726456	0.534055
500	0.742699	0.514699

Table 8: Results obtained using AdaBoost.MH without the TSR process.

resulted in significantly different feature sets. In this case rerun the experiments, and possibly rerun the experiments on the older versions of AdaBoost that we had previously proposed.

- Run experiments using the most significant bigrams that Fernanda Caropreso found in her experiments. These could be the intersection of the sets that she has found by using different feature selection techniques.
- Trying to maximize F_1 instead of minimizing Hamming loss (unless it turns out to be the same as minimizing Hamming loss). After all, F_1 is the evaluation measure we have chosen.
- Modify the choice of the weak hypotheses in such a way to take into account non-binary weights of the documents. This could be achieved by segmenting the [0,1] interval into n chunks and building one-level n-ary decision trees as a result.
- Study (theoretically) the bounds on Z_s for our algorithm.
- Find a better motivated combination rule than geometrical mean for the production of the sub-committee.

Acknowledgements

We thank Luigi Galavotti for making available his RealCAT text classification software environment, which greatly simplified our experimental work.

References

[1] Gianni Amati and Fabio Crestani. Probabilistic learning for selective dissemination of information. *Information Processing and Management*, 35(5):633–654, 1999.

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

[3] Richard S. Forsyth. New directions in text categorization. In Alex Gammerman, editor, *Causal models and intelligent data management*, pages 151–185. Springer, Heidelberg, DE, 1999.

[4] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In Jude W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 170–178, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[5] Yoav Freund and Robert 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.

[6] Norbert Fuhr, Stephan Hartmann, Gerhard Knorz, Gerhard Lustig, Michael Schwantner, and Konstadinos Tzeras. AIR/X – a rule-based multistage indexing system for large subject fields. In André Lichnerowicz, editor, *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 606–623, Barcelona, ES, 1991. Elsevier Science Publishers, Amsterdam, NL.

[7] Norbert Fuhr and Ulrich Pfeifer. Probabilistic information retrieval as combination of abstraction inductive learning and probabilistic assumptions. *ACM Transactions on Information Systems*, 12(1):92–115, 1994.

[8] Philip J. Hayes and Steven P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In Alain Rappaport and Reid Smith, editors, *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–66. AAAI Press, Menlo Park, US, 1990.

[9] David A. Hull, Jan O. Pedersen, and Hinrich Schütze. Method combination for document filtering. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 279–288, Zürich, CH, 1996. ACM Press, New York, US.

[10] David J. Ittner, David D. Lewis, and David D. Ahn. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, US, 1995.

[11] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[12] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[13] George H. John, Ron Kohavi, and Karl 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.

[14] Leah S. Larkey. Automatic essay grading using text categorization techniques. In W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 90–95, Melbourne, AU, 1998. ACM Press, New York, US.

[15] Leah S. Larkey. A patent search and classification system. In Edward A. Fox and Neil Rowe, editors, *Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, pages 179–187, Berkeley, US, 1999. ACM Press, New York, US.

[16] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, CH, 1996. ACM Press, New York, US.

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

[18] David D. Lewis. Reuters-21578 text categorization test collection. Distribution 1.0, 1997.

[19] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[20] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[21] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996. ACM Press, New York, US.

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

[23] Ray Liere and Prasad Tadepalli. Active learning with committees for text categorization. In *Proceedings of AAAI-97, 14th Conference of the American Association for Artificial Intelligence*, pages 591–596, Providence, US, 1997. AAAI Press, Menlo Park, US.

[24] Dan Roth. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 806–813, Madison, US, 1998. AAAI Press, Menlo Park, US.

[25] Robert E. Schapire. Theoretical views of boosting. In Paul Fischer and Hans U. Simon, editors, *Proceedings of EuroCOLT-99, 4th European Conference on Computational Learning Theory*, pages 1–10, Nordkirchen, DE, 1999. Lecture Notes in Computer Science, number 1572, Springer Verlag, Heidelberg, DE.

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

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

[28] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. In W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of SIGIR-98,*

21st ACM International Conference on Research and Development in Information Retrieval, pages 215–223, Melbourne, AU, 1998. ACM Press, New York, US.

[29] Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, US, 1995. ACM Press, New York, US.

[30] Sam Scott and Stan Matwin. Feature engineering for text classification. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[31] Fabrizio Sebastiani. Machine learning in automated text categorisation. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999. Submitted for publication to *ACM Computing Surveys*.

[32] Sholom M. Weiss, Chidanand Apte, Fred J. Damerau, David E. Johnson, Frank J. Oles, Thilo Goetz, and Thomas Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[33] Yiming Yang. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[34] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.