Knowledge-Based Systems 103 (2016) 104–117

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

RFBoost: An improved multi-label boosting algorithm and its application to text categorisation

Bassam Al-Salemi, Shahrul Azman Mohd Noah*, Mohd Juzaiddin Ab Aziz

Knowledge Technology Research Group, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 14 January 2015
Revised 23 March 2016
Accepted 29 March 2016
Available online 6 April 2016

Keywords:
RFBoost
Boosting
AdaBoost.MH
Text categorisation
Labeled Latent Dirichlet Allocation
Multi-label classification

ABSTRACT

The AdaBoost.MH boosting algorithm is considered to be one of the most accurate algorithms for multi-label classification. AdaBoost.MH works by iteratively building a committee of weak hypotheses of decision stumps. In each round of AdaBoost.MH learning, all features are examined, but only one feature is used to build a new weak hypothesis. This learning mechanism may entail a high degree of computational time complexity, particularly in the case of a large-scale dataset. This paper describes a way to manage the learning complexity and improve the classification performance of AdaBoost.MH. We propose an improved version of AdaBoost.MH, called *RFBoost*. The weak learning in RFBoost is based on filtering a small fixed number of ranked features in each boosting round rather than using all features, as AdaBoost.MH does. We propose two methods for ranking the features: One Boosting Round and Labeled Latent Dirichlet Allocation (LLDA), a supervised topic model based on Gibbs sampling. Additionally, we investigate the use of LLDA as a feature selection method for reducing the feature space based on the maximal conditional probabilities of words across labels. Our experimental results on eight well-known benchmarks for multi-label text categorisation show that RFBoost is significantly more efficient and effective than the baseline algorithms. Moreover, the LLDA-based feature ranking yields the best performance for RFBoost.

1. Introduction

Boosting is an ensemble method for improving the performance of any learning algorithm; it is considered to be one of the most powerful machine learning methods introduced in the last two decades [19]. The primary idea behind boosting is to combine the outputs of multiple simple classifiers called *weak learners* (a.k.a. weak hypotheses) into a powerful composite classifier called a *strong classifier* (a.k.a. a final hypothesis) [6,34]. Boosting was first proposed in the computational learning theory literature [15]. The most common boosting algorithm produced in theory and in practice, by Yoav [16], is Adaptive Boosting (AdaBoost). AdaBoost has evolved over the last several years and has been widely used in many applications as an accurate ensemble method, for example, in clustering [32] and in instance selection [17]. Many boosting algorithms have extended AdaBoost to solve problems in binary and multi-class classification. A complete review of boosting algorithms appears in [13].

Text categorisation (TC) is the task of automatically assigning a given document to predefined categories based on its content [1]. The rapid daily increase in online texts makes TC an interesting area for developing accurate classifiers and systems to automatically categorise texts. Most state-of-the-art classifiers handle single-label multi-class classification problems. However, by their very nature, texts may belong to more than one category. Many studies have proposed ways to adapt traditional classifiers for solving the multi-label classification problem; these include multi-label naïve Bayes classification [43], ensemble learning [25,44], sparse logistic regression [26], ML-KNN [44], and MLCC [42]. Another classifier that has proved effective for multi-label classification is AdaBoost.MH [36], a boosting algorithm that is an extension of AdaBoost for multi-label classification. AdaBoost.MH works by iteratively building a committee of weak decision stump hypotheses and returning a composite strong classifier with a small Hamming loss error. Using the idea of *domain partitioning* for weak learning, all features are examined in each boosting round of AdaBoost.MH learning to determine whether they are absent or present in each category. To minimise the Hamming loss, only one feature, called the *pivot* feature, is then chosen from among the examined features and used to build a new weak hypothesis. This mechanism makes AdaBoost.MH's learning time highly sensitive to

* Corresponding author. Tel.: +60389216343, fax: +60 389256732.
E-mail addresses: bassamalsalmi@yahoo.com (B. Al-Salemi), shahrul@ukm.edu.my (S.A. Mohd Noah), juzaiddin@ukm.edu.my (M.J. Ab Aziz).

http://dx.doi.org/10.1016/j.knosys.2016.03.029
0950-7051/

the number of features and may lead to a high degree of learning complexity, particularly for large-scale datasets.

Much research has been devoted to improving AdaBoost.MH's effectiveness and managing its complexity since it first appeared [7,9,11,22,23,37,47–49]. There are essentially three ways to accelerate AdaBoost.MH, which can be grouped as follows.

Dimension-reduction-based acceleration: Dimension reduction methods involve a pretask which reduces the feature space of the training examples using feature selection techniques before learning. This is a general method for any supervised machine learning classification algorithm [8,12,41].

Representation-based acceleration: Most work on AdaBoost.MH for TC concerns the bag-of-words (BOW) representation model. Al-Salemi et al. [3] proposed LDA-AdaBoost.MH, in which a Latent Dirichlet Allocation (LDA) [5] topics model is used instead of BOW to extract latent topics as features to represent texts. Experimental results proved that representing texts using a small number of topics significantly accelerates AdaBoost.MH learning and improves its performance. This framework has been extended to involve the most well-known multi-label boosting algorithms for multi-label TC [2]. Although topics-based representation has proved to be efficient in improving boosting algorithms in general, its classification performance is poor compared to BOW for imbalanced and large-scale datasets [2]. Furthermore, the LDA topic model was essentially proposed for textual data [5], and the topics-based representation cannot be extended to other multi-label classification problems. Therefore, in this work we focus on improving the boosting algorithm using the BOW representation, as this is the simplest representation model for general multi-label classification tasks.

Weak-learning-based Acceleration: Weak-learning-based improvements of AdaBoost.MH can involve changing the number of pivot terms used to build the weak hypotheses, changing the base learner, and reducing the search space of pivot terms in each boosting round. Sebastiani et al. [37] proposed using a set of pivot terms in AdaBoost.MH^{KR} (*AdaBoost.MH with k-fold real-valued predications*). Rather than choosing only one pivot term in each boosting round, as AdaBoost.MH does, AdaBoost.MH^{KR} chooses the k top-ranked features which have similar scores as pivot terms to build a new weak hypothesis, and it uses the mean of their scores as a prediction for the hypothesis. Similarly, MP-Boost [11] selects multiple pivot terms, one for each label, in every iteration instead of only once, as AdaBoost.MH does. Regarding the base learner, De Comité et al. [9] used an *alternating decision tree* (ADTree) [14] instead of a decision stump as a base learner for AdaBoost.MH; each node of a multi-label ADTree is associated with a set of real values, one for each label. Busa-Fekete and Kégl [7] combined trees and products to model AdaBoost.MH's base learner as sequences of decisions over the smaller partitioning used for stumps. Recently, Kégl [22] adapted Hamming trees for learning in AdaBoost.MH to conduct multi-class classification without a one-against-all inner transformation.

To reduce the search space of pivot terms, Escudero et al. [10] proposed an accelerated version of AdaBoost.MH, LazyBoosting, in which the search space is reduced to a fixed number of feature terms which are selected randomly in each boosting round. However, random selection of the features without taking their importance into account may lead to poor performance. Busa-Fekete and Kégl [7] improved this algorithm by posing feature selection as a multi-armed bandit (MAB) problem and using the Upper Confidence Bound (UCB) bandit algorithm [4] to select only informative features while retaining some exploration of new features. The idea behind this method is to associate a bandit arm with each feature, viewing loss reduction as a reward.

This paper presents an improved version of AdaBoost.MH, called the *Rank-and-Filter-based Boosting algorithm* (RFBoost). RFBoost retains the simplicity and generality of AdaBoost with a new method of accelerating the weak learning based on reducing the search space of pivot terms in each boosting round. Using the suggestion of Mukherjee and Schapire [29] that the weak hypotheses of AdaBoost.MH should only return a single multi-class prediction per example, and similar to LazyBoosting, RFBoost filters a fixed number of feature terms in each boosting round to build a new weak hypothesis. The difference between RFBoost and LazyBoosting is that the features in LazyBoosting are selected randomly, whereas RFBoost first ranks the features and then, in each iteration, filters only the top k features among which the *pivot term* is most likely located to build a new weak hypothesis. The advantage of ranking the features as a separate task before weak learning is that the feature space is directly reduced in each iteration without using an inner feature selection method such as MABs [7]. We employ two methods to rank the features; one is based on the boosting weights of the features obtained in the first boosting round and the other uses a supervised topic model based on Gibbs sampling [38,39].

2. AdaBoost.MH

A multi-label classification algorithm assigns a set of target labels to each sample. Let χ be a finite set of examples and let $\mathcal{Y} = (y_1, \ldots y_m)$ be a finite set of labels such that each example $x_i \in \mathcal{X}$ belongs to multiple class labels in the set Y_i, where $Y_i \subseteq \mathcal{Y}$. Let a set of training examples $\boldsymbol{S} = \{(x_1, Y_1), \ldots, (x_n, Y_n)\}$ be given. The machine-learning task is to induce from $\boldsymbol{S}$ a *classifier* f: $\chi \to 2^y$ capable of estimating an unknown *target function* φ: $\chi \to 2^y$. The single-label multi-class classification problem is a special case of a multi-label problem in which each example is assigned to only one label.

To make this definition accord with the AdaBoost.MH formal description (as illustrated in Algorithm 1), the *target function* φ will be defined as $\varphi : \mathcal{X} \times \mathcal{Y} \to \{\pm 1\}^m$, in which $\varphi(x,y) = 1$ if y is the correct label for x and $\varphi(x, y) = -1$ otherwise. For each example, the output of φ is a set of length m of the values $+1$ and -1. This setup converts the multi-class problem into a *one-against-all binary classification problem*. For simplicity, we denote the label index for y in $\mathcal{Y}$ as l; then $\varphi(x, y)$ is written as $\varphi(x, l)$ for $l = 1, \ldots, m$.

The main goal of AdaBoost.MH is to infer a composite strong classifier (the final hypothesis) $H : \chi \times \mathcal{Y} \to \mathbb{R}$ from a set of weak hypotheses $(h^1(x), \ldots, h^R(x))$ with a small Hamming loss. The Hamming loss measuring the empirical risk (training error) is defined as follows:

$$\hat{R}_{HL}(H, \mathbf{W}) = \sum_{i=1}^{n} \sum_{l=1}^{m} w_{i,l} \; sign(H(x_i, l)) \neq \varphi(x_i, l), \tag{1}$$

where $\mathbf{W} = [w_{i,l}]_{n \times m}$ is a uniform distribution matrix over the training examples and labels, and . is the indicated function, which returns 1 if the inequality is true and 0 otherwise—the comparison is based on the fact that the output of the strong classifier $H(x_i, l)$ is a positive or negative real value and $\varphi(x_i, l)$ takes the value $+1$ or -1, depending on the predefined category assignment.

AdaBoost.MH builds the strong classifier with a small Hamming loss by minimising the *exponential-margin-based error*, defined as follows:

$$\hat{R}_{EXP}(H, \mathbf{W}) = \sum_{i=1}^{n} \sum_{l=1}^{m} w_{i,l} exp(- H(x_i, l) \varphi(x_i, l)). \tag{2}$$

Given a training dataset $\boldsymbol{S}$ and an initial distribution of weights $\mathbf{W}$ over the training examples (the initial distribution is discussed in Section 3.2), AdaBoost.MH works by iteratively passing $\mathbf{W}$ along with $\boldsymbol{S}$ to a *base learner* and generating a set of weak hypotheses $h^{(r)} : \chi \times \mathcal{Y} \to \mathbb{R}$, $r = 1, \ldots, R$. In each round r, the goal is to

minimise the *base objective* $Z^{(r)}$, which is defined as

$$Z^{(r)} = \sum_{i=1}^{n}\sum_{l=1}^{m} w_{i,l}^{(r)}\ exp\left(-\alpha^{(r)}\ h^{(r)}(\mathrm{x}_i, l)\varphi(\mathrm{x}_i, l)\right), \tag{3}$$

where $\alpha^{(r)} \in \mathbb{R}^+$ is a positive real-valued *base coefficient*.

In the next round, $r+1$, the distribution $\mathbf{W}^{(r+1)}$ is updated based on the predicted values of the current hypothesis $h^{(r)}$ according to the rule

$$w_{i,l}^{(r+1)} = \frac{w_{i,l}^{(r)}\ \exp\left(-\alpha^{(r)}\ h^{(r)}(\mathrm{x}_i, l)\varphi(\mathrm{x}_i, l)\right)}{Z^{(r)}}, \tag{4}$$

so that the weight assigned by $h^{(r)}$ to misclassified examples is increased in order to be classified correctly in the following rounds.

After generating the weak hypotheses, AdaBoost.MH builds the final classifier as a composite of the weak hypotheses, defined as follows:

$$H(\mathrm{x}, l) = \sum_{r=1}^{R} h^{(r)}(\mathrm{x}, l). \tag{5}$$

Schapire and Singer [36] proved that $\hat{R}_{EXP}(H, \mathbf{W}) \leq \prod_{r=1}^{R} Z^{(r)}$.

In the case of *single-label* multi-class classification, the correct label assigned to a given example x is the label which has obtained the maximum estimated value:

$$l_H(\mathrm{x}) = \arg\max H(\mathrm{x}, l),\ \ l = 1, \ldots,\ m. \tag{6}$$

In contrast, for a *multi-label* corpus, the correct labels assigned to x are those which have obtained a positive value, while those with negative values are considered false labels:

$$l_H(\mathrm{x}) = sign H(\mathrm{x}, l),\ \ l\ = 1, \ldots,\ m. \tag{7}$$

However, if all estimated values in Eq. (7) are negative, then the given example would be treated as a misclassified example. To avoid this in the present study, when all of the labels have negative values, the example is assigned to the label which has the maximum value. This rule is used in all algorithms evaluated in this paper.

We now turn to AdaBoost.MH with real-valued predictions. Schapire and Singer [35] proposed an efficient base learner for AdaBoost.MH using binary features based on the idea of *domain partitioning*. Given a training set $\boldsymbol{S} = \{(\mathrm{x}_1,\ Y_1),\ \ldots,\ (\mathrm{x}_n,\ Y_n)\}$ in which $\mathrm{x}_i \in \mathcal{X}$ are textual documents, let $\mathcal{T} = \{t_1, \ldots,\ t_u\}$ be the set of all possible terms extracted from the training documents, and let each document x_i be represented as a vector $\langle x_1^i, \ldots,\ x_u^i\rangle$ of u binary weights such that $x_k^i = 1$ if t_k occurs in x_i and $x_k^i = 0$ otherwise.

When generating a new weak hypothesis, AdaBoost.MH analyses the absence or presence of each term t_k in a document x_i as follows to decide whether it belongs to a label l:

$$h^{(r)}(\mathrm{x}_i, l) = \begin{cases} c_{0l} & if\ x_k^i = 0 \\ c_{1l} & if\ x_k^i = 1 \end{cases}, \tag{8}$$

where c_{0l} and c_{1l} are real-valued constants chosen during iteration r according to the minimisation policy of the base objective $Z^{(r)}$. Using the idea of domain partitioning to calculate the decision stump c_{ul} (u ∈ {0, 1}) for term t_k, the training example set $\mathcal{X}$ is first divided into two subsets (X_0, X_1), as follows:

$$X_u = \{\mathrm{x} : x_k = u\ \},\ \ u = 0, 1, \tag{9}$$

so that X_1 is the set of all documents containing t_k and X_0 is the set of all documents which do not contain t_k. Then, for each label l, the weights of all documents in X_u which belong to l with respect to the distribution matrix $\mathbf{W}^{(r)}$ are summed:

$$W_{ul}^{+} = \sum_{i=1}^{n} w_{i,l}^{(r)}\ [\![x_k^i = u]\!][\![\varphi(\mathrm{x},\ \ l) = +1]\!] \tag{10}$$

The summation of the weights of documents which do not belong to l is also calculated:

$$W_{ul}^{-} = \sum_{i=1}^{n} w_{i,l}^{(r)}\ [\![x_k^i = u]\!][\![\varphi(\mathrm{x},\ \ l) = -1]\!]. \tag{11}$$

Thus, W_{0l}^{+} is the total weight of all documents which do not contain t_k and belong to l, W_{1l}^{+} is the total weight of all documents which do contain t_k and belong to l, W_{0l}^{-} is the weight of all documents which do not contain t_k and do not belong to l, and W_{1l}^{-} is the weight of all documents which do contain t_k but do not belong to l.

Then, for each term t_k, c_{ul} is calculated as follows for $u = 0,\ \ 1$:

$$c_{ul} = \begin{cases} \frac{1}{2}\ln\frac{W_{0l}^{+}}{W_{0l}^{-}}\ \ if\ x_k = 0 \\ \frac{1}{2}\ln\frac{W_{1l}^{+}}{W_{1l}^{-}}\ \ if\ x_k = 1. \end{cases} \tag{12}$$

By choosing $\alpha^{(r)} = 1$ [36], we have:

$$Z^{(r)} = 2\sum_{u\in\{0,1\}}\sum_{l=1}^{m}\sqrt{W_{ul}^{+} \times W_{ul}^{-}}. \tag{13}$$

The term which obtains the smallest value of $Z^{(r)}$, which is the *pivot term*, is then used to build the weak hypothesis $h^{(r)}$ according to Eq. (8), and the distribution matrix $\mathbf{W}^{(r+1)}$ is generated for the next iteration according to Eq. (4).

To avoid division by zero, the values in Eq. (12) are smoothed by a small value ϵ in both the denominator and the numerator. Therefore, Eq. (12) will now take the form

$$c_{ul} = \begin{cases} \frac{1}{2}\ln\frac{W_{0l}^{+} + \epsilon}{W_{0l}^{-} + \epsilon}\ \ if\ x_k = 0 \\ \frac{1}{2}\ln\frac{W_{1l}^{+} + \epsilon}{W_{1l}^{-} + \epsilon}\ \ if\ x_k = 1, \end{cases} \tag{14}$$

where $\epsilon = 1/mn$ [36].

Algorithm 2 presents AdaBoost.MH's weak learning with real-valued predictions using binary features.

Even this weak learning method is effective; however, a problem that arises is the computational complexity of the learning. This is because in each boosting round, the base learner uses all possible binary feature terms in a given feature term index $\mathcal{T}$ to select the single pivot term and build a new weak hypothesis; this is the main disadvantage of AdaBoost.MH learning [11].

3. Weighting policy

One of the most important factors which affect boosting algorithm performance is how the distribution **W** is weighted over the training examples and labels. Weighting policies involve initialising the distribution to serve as an input for the boosting algorithm and then normalising the distribution after updating the examples' weights in each boosting round during weak learning. In lieu of using the formal weight initialisation [35], we propose a novel initialisation method for the weight distribution which is based on the supervised topic model LLDA. Before presenting our proposed initialisation method, we first describe LLDA.

3.1. Labeled LDA

Latent Dirichlet Allocation (LDA) [5] is a generative probabilistic model for identifying the latent topics in a set of documents. In LDA, each document is represented as a combination of a fixed number of topics, while each topic is represented as a combination of words. The combination distributions are assumed to be

Dirichlet-distributed random variables and must be inferred from the data.

LDA is an unsupervised model, that is, it does not take the categorical structure of the documents into account when extracting latent topics. Several modifications of LDA to incorporate supervision have been proposed [27, 31, 46]. The first supervised LDA model, proposed by McAuliffe and Blei [27], was essentially designed for single-label corpora. It limits the assignment of a document to only one class label. Ramage et al. [31] proposed a supervised LDA model called *Labeled LDA* (LLDA) for multiple-labelled corpora. LLDA restricts the number of topics to the number of unique labels (categories). The major modification which differentiates LLDA from LDA is that in LLDA, the definitions of the topics assigned to a document are restricted to the multiple labels which the document belongs to. To estimate the latent topics in LLDA, Gibbs sampling [18,20] is employed, as follows.

Given a training set $\boldsymbol{S}$ and the word index $\mathcal{T}$ extracted from $\boldsymbol{S}$, the probability of a label l_i being assigned to a word t_i in document x given all other label assignments to all other words is sampled from the following multinomial distribution:

$$p(l_i = k|L_{-i},\ \mathrm{x}) \propto \frac{n_{k,t}^{(-i)} + \beta_t}{\sum_{t'} n_{k,t'}^{(-i)} + \beta_{t'}} \frac{n_{\mathrm{x},k}^{(-i)} + \alpha_k}{\sum_{k'} n_{\mathrm{x},k'}^{(-i)} + \alpha_{k'}} \tag{15}$$

where L_{-i} denotes all label assignments except for the current label l_i, $n_{k,t}^{(-i)}$ is the number of times the word t is assigned to label l except for the current assignment, $\sum_{t'} n_{k,t'}^{(-i)}$ is the total number of words assigned to label l except for the current assignment, $n_{\mathrm{x},k}^{(-i)}$ is the total number of words in document x assigned to label l except for the word t_i, $\sum_{k'} n_{\mathrm{x},k'}^{(-i)}$ is the total number of words in document x assigned to all labels except for the current label k, and α and β are real-valued constants such that $0 < \alpha,\ \beta < 1$.

In our case, where the dataset is supervised, the set of labels L in Eq. (15) given with the document x is extracted from the pair (x, Y) in $\boldsymbol{S}$. After a few resampling iterations, the label index assignment for all of the words in the training corpus is obtained. Then this label index assignment is used to obtain the label–word distribution Φ and the document–label distribution Θ according to the following formulas:

$$\phi_{l,t} = p(t|l) = \frac{n_{l,t} + \beta_t}{\sum_{t'} (n_{l,t'} + \beta_{t'})} \tag{16}$$

$$\theta_{\mathrm{x},l} = p(\mathrm{x}|l) = \frac{n_{\mathrm{x},l} + \alpha_l}{\sum_{l'} (n_{\mathrm{x},l} + \alpha_{l'})} \tag{17}$$

where $n_{l,t}$ is the number of times label l is assigned to the word token t and $n_{\mathrm{x},l}$ is the number of times label l is assigned to some word token in document x. Therefore, $\Theta_{n\times m} = \{\theta_{\mathrm{x},l}\}$, where n is the total number of documents and m is the total number of single labels, and $\Phi_{m\times v} = \{\phi_{l,t}\}$, where $v = |\mathcal{T}|$ is the vocabulary size.

The reason for using LLDA in this study is to obtain the document–label distribution Θ and the label–word distribution Φ. The document–label distribution will be used to initialise the distribution matrix for weak learning (Section 3.2), and the label–word distribution will be used to obtain a weighted and ranked features index as an input for the RFBoost algorithm and as a feature reduction method (Section 5).

3.2. Distribution initialisation based on LLDA

Instead of random guessing, Schapire and Singer [36] gave an initial weight to each example x as follows:

$$w_{i,l} = \frac{1}{mn} \quad for\ i = 1, \ldots,\ n;\ \ l = 1, \ldots,\ m. \tag{18}$$

Thus, the initial weight given to all examples is the same, namely, $1/mn$. To distinguish between this initial distribution initialisation policy and the one we propose in this section, we will call the former the *formal* initialisation policy.

In place of the formal weighting policy defined in Eq. (18), we propose a new weighting method using LLDA. As mentioned in the previous section, our goal in using LLDA is to obtain the document–label distribution Θ and the label–word distribution Φ. For weighting the initial distribution matrix $\mathbf{W}$, we are interested in the document–label distribution Θ. However, Θ is a multinomial distribution, meaning that for each document x, the distribution of x over the set of labels L is a uniform distribution: $\sum_l \theta_{\mathrm{x},l} = 1$. But then assigning $w_{\mathrm{x},l}$ directly to $\theta_{\mathrm{x},l}$ will lead to a non-uniform distribution because the summation of $w_{\mathrm{x},l}$ for all documents and their label assignments in the training set will be

$$\sum_x \sum_l w_{\mathrm{x},l} = \sum_x \sum_l \theta_{\mathrm{x},l} = \sum_x 1 = n. \tag{19}$$

Therefore, to normalise $\boldsymbol{W}$ to be a uniform distribution, we simply divide the document–label weights by

$$w_{\mathrm{x},l} = \frac{\theta_{\mathrm{x},l}}{n},\ \ \forall\ \mathrm{x} \in \mathcal{X} \text{ and } l = 1, \ldots,\ m. \tag{20}$$

The purpose of this initialisation is to obtain a perfect distribution; the weight given to a document x assigned to predefined labels l is greater than the weight which is given if x does not belong to l, depending on the terms occurring in x.

3.3. Distribution normalisation

In each boosting round r, the distribution matrix $\mathbf{W}^{(r+1)}$ is updated according to Eq. (4) and normalised by $Z^{(r)}$ (defined in Eq. (3)). The normalisation factor $Z^{(r)}$ is simply the summation of the cumulative exponential-margin-based errors, $\sum_l w_{\mathrm{x},l} exp(-\alpha^{(r)}\ h^{(r)}(\mathrm{x}_i, l)\varphi(\mathrm{x}_i, l))$, for all documents in the training set. However, although using such a normalisation policy—which we will call *per-document* normalisation—is straightforward, it ignores the nature of the dataset, in which some categories may only have a small number of documents. Accordingly, we propose a new normalisation, which we call *per-category* normalisation. Algorithm 3 describes *per-category* normalisation for a given distribution matrix $\mathbf{W}$. First, for each label l, the total weight of all documents w_l is calculated. Then, to ensure that $\mathbf{W}$ is uniform, w_l is multiplied by the number of categories m. After that, each weight $w_{i,l}$ is normalised by w_l. Thus, the output is a normalised uniform distribution of weights. The purpose of Algorithm 3 is not merely to normalise the distribution by category; rather, it ensures that the exponential-margin-based error is minimised for each category.

4. Proposed boosting algorithm: RFBoost

Algorithm 2 shows how weak learning is performed using AdaBoost.MH based on domain partitioning. Given a feature index $\mathcal{T}$, in each boosting round r, AdaBoost.MH examines all features in $\mathcal{T}$ to select only one pivot feature for building a new weak hypothesis in which Z (Eq. (13)) is minimal. Here we are concerned with how this will work if $\mathcal{T}$ contains a very large number of features. Using the same weak learning mechanism may lead to *serious* computational complexity. We propose RFBoost to tackle this problem without losing the generality of boosting theory. The letters *RF* in *RFBoost* stand for *Rank and Filter*. RFBoost's weak learning is based on filtering a set of ranked features $\mathcal{T}'$ from $\mathcal{T}$, sorted by weights. Then, in each boosting round r, only the top $\boldsymbol{k}$ ranked features in $\mathcal{T}'$—where the pivot term is most likely located—are used to obtain

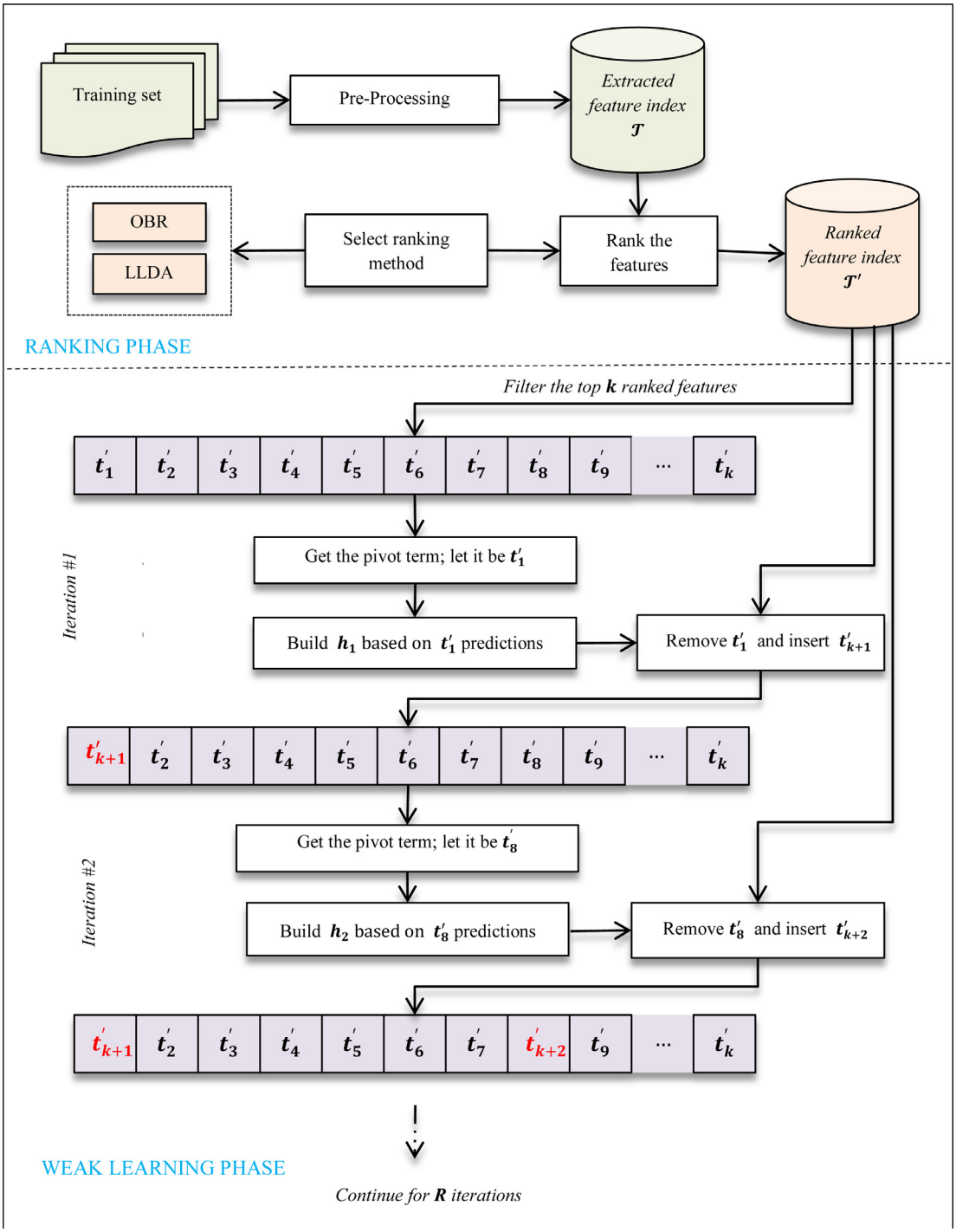

Fig. 1. RFBoost weak learning. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

the pivot term and build a new weak hypothesis. In this section, we describe RFBoost's weak learning on the assumption that the features have already been ranked. In the next sections, we will describe in detail our proposed methods for ranking the features as a pre-task for RFBoost.

In Algorithm 4, the ranked features index $\mathcal{T}'$ and the number of filtered features $\boldsymbol{k}$ are given as inputs. In line 2, a list called *FilterdList* is created to include the top $\boldsymbol{k}$ features in the ranked features index $\mathcal{T}'$. Then, for the first iteration, one feature from *FilterdList*, the *pivot term*, is obtained (lines 3–12). After obtaining the pivot term, generating the new weak hypotheses and updating the distribution for the current iteration r, the pivot term is eliminated from *FilterdList* and the next feature in the ranked features index $\mathcal{T}'$, $t'_{(k+r)}$, is added to *FilterdList*. Thus, the search domain for the pivot terms in each boosting round always has the same length, $\boldsymbol{k}$, and the term chosen as the pivot term is excluded during the following iterations.

Fig. 1 illustrates this process with a graphical example. After ranking the features, RFBoost filters the top $\boldsymbol{k}$ ranked features in $\mathcal{T}'$ for the weak learning. Then, for the first boosting iteration, the filtered $\boldsymbol{k}$ features are examined to obtain the pivot term. Assuming that the pivot term is t'_1, the first weak hypothesis is generated based on the real-valued predictions for t'_1 (Eq. (14)). Then, for the next iteration, t'_1 is removed from the top $\boldsymbol{k}$ features and replaced

Table 1
The pivot terms obtained during the first 14 iterations of weak learning on *Reuters-21578*.

Iteration	Ranked features based on OBR weights	Pivot	Pivot position
1	**102, 4879, 9009**, 92, 91, 15,182, **6972, 9004**,	102	1
2	**1761**, 9987, 2191, **320**, 15,179, 632, **8070, 24**,	9009	3
3	710, **417**, 15,199, 11,809, 157, 380, 12, **288**,	320	12
4	2114, 332, 30, 15,207, 7663, 11, **2478**, 46,	9004	8
5	12,684, 15183, 647, 14, 231, 4580, 5888, 36,	1761	9
6	872, 11,754, 283, 331, **5222**, 526, 5893, 16,	24	16
7	15,180, 9005, 2506, 1343, 1379, 2509, 13, 2562,	6972	7
8	90, 524, 5108, 4852, 1267, 3702, 11746, **8878**,	4879	2
9	866, 15,056, 6077, 2341, 1398, 757, 707, 594,	288	24
10	605, 1821, 1866, 1401, 10,584, 23, 2080, 1751,	8070	15
11	1246, 314, 2442, 570, 71, 341, 5892, 1022, 145,	2478	31
12	5228, 421, 3356, 162, 559, 704, 1469, 178, 708,	417	18
13	6333, 11,822, 15,313, 716, 5570, 610, 249,	8878	51
14	12,102, 321, 12,339, 2853, 12,508, 10,043, 718, 578, …	5222	45

Note: The second column contains the features ranked according to their OBR weights, and the features written in bold and underlined are the pivot terms which were selected during the first 14 boosting iterations of weak learning. The third column lists the pivot terms selected during each iteration. The fourth column lists the index of the selected pivot term in the features index.

by the next feature in $\mathcal{T}'$ which is $t'_{(k+1)}$, the feature in red. This process is repeated in the following iterations, and the size of the filtered set of ranked features in each boosting round is always $\boldsymbol{k}$.

The main advantage of RFBoost is that the domain search space for selecting the pivot term is reduced from the size $|\mathcal{T}|$, which may be hundreds of thousands, to a subset of $\mathcal{T}$ of size $\boldsymbol{k}$, where $\boldsymbol{k}$ is a small fixed number given as input. This mechanism dramatically decreases the computational time for weak learning. In addition, using weighted features rather than binary ones may yield a more accurate classification.

4.1. One-boosting-round feature ranking

Suppose that $\mathcal{T} = \{t_1, \ldots, t_V\}$ is the extracted feature index for a given training set $\boldsymbol{S}$ and let $\mathbf{W}$ be the initial distribution matrix drawn from the set of examples in $\boldsymbol{S}$. By performing only one boosting round on $\boldsymbol{S}$ using $\mathcal{T}$ and based on $\mathbf{W}$ (Eqs. (9)–(14)), we obtain a set of *normalisation factors* $\mathcal{Z} = (Z_1, \ldots, Z_V)$, one for each feature in $\mathcal{T}$. Hence, Z_i measures the Hamming loss error for feature term t_i. Given that a smaller value of Z_i means that t_i is more important, the values of Z_i can be used as a measure of each term t_i's importance. We propose calling the values of $\mathcal{Z}$ the *One-Boosting-Round* (OBR) *weights* of the features in $\mathcal{T}$; these weights will be used to directly rank the extracted features as an input for RFBoost without using an external ranking method. However, we need to prove that the OBR weights are capable of ranking the extracted features. For this purpose, we performed an experimental analysis on real data using the benchmark dataset *Reuters-21578*. The purpose of this analysis was to understand how the pivot terms were chosen and the relationship between the pivot terms and the features ranked by their OBR weights.

Table 1 shows the pivot terms used to build the weak hypotheses in the first 14 iterations of weak learning on *Reuters-21578*. The second column is the list of the top-ranked features sorted in descending order based on their OBR weights. The third column contains the pivot terms which were obtained in each round. The last column contains the index of each pivot term in the ranked feature list. For example, the pivot term 102, obtained in the first boosting round, is located in the first position of the ranked features list; the pivot term 9009, which prevailed in the second iteration, is located in the third position of the ranked features list; and so on. The 14 pivot terms for the first 14 iterations were all positioned among the top 51 features in the ranked features list, based on their boosting weights obtained in the first boosting round.

This analysis confirms that OBR is capable of ranking the features, and the pivot terms obtained in the next iterations are located at the top of the ranked feature index using OBR weights. This analysis emphasises that after performing one boosting round on a given training set, sorting the training features in $\mathcal{T}$ in descending order based on their OBR weights, and storing them in a new index$\mathcal{T}'$, the ranked features index $\mathcal{T}'$ can be used as an input for RFBoost. The procedure for OBR-based feature ranking is presented in Algorithm 5.

4.2. LLDA for feature ranking

As described in section 3.1, the goal in LLDA is to obtain the document–label distribution Θ and the label–word distribution Φ. Here we are concerned with the label–word distribution Φ, which contains the conditional probabilities of the words being assigned to each label. These probabilities are used as weights for ranking the features. To obtain the ranked features index $\mathcal{T}'$, all of the terms in $\mathcal{T}$ are sorted in ascending order based on their conditional probabilities $\phi_{l,t}$ as weights and then stored in $\mathcal{T}'$. In lines 4–8 of Algorithm 6, the maximum probabilities among all label–word conditional probabilities are obtained and are then stored along with the associated feature in $\mathcal{T}'$ in line 9. Next, the weighted feature index $\mathcal{T}'$ is sorted according to those weights and returned as output. The ranked features index $\mathcal{T}'$ using LLDA will be used as an input for RFBoost.

5. LLDA-based feature selection

The unsupervised LDA topic model has been employed for feature selection in TC. A study conducted by Zhang and colleagues [45] used unsupervised LDA with Gibbs sampling [18] as a feature selection method for TC. They used a label–word distribution matrix to select the words with lower entropy, and then they used a state-of-the-art classifier for classification. A similar study was conducted by Tasci and Gungor [39]; however, they used variational expectation maximisation [5] for the estimation instead of Gibbs sampling.

In this study, we use the label–word distribution Φ, obtained using the supervised topic model LLDA. In our case, the topics were the predefined labels. After obtaining the topic–word distribution Φ, the top words were simply selected based on the maximal conditional probabilities of the words across the predefined labels. Algorithm 7 presents the procedure.

Table 2
The contingency table.

Category l		Predefined assignment	
		True	False
Estimated assignment	True	TP_l	FP_l
	False	FN_l	TN_l

6. Evaluation measures

The output of learning for both AdaBoost.MH and RFBoost is a strong classifier of the form $H : \chi \times \mathcal{Y} \rightarrow \mathbb{R}$. In the case of a single-label dataset, the *true* estimated category of a given document is the one that has the maximum predicted value of H, whether it is positive or negative (Eq. (6)). If the dataset is multi-labelled, then the *true* estimated categories are all the categories with positive values (Eq. (7)). Those categories that have negative values are considered *false* labels. To obtain the contingency table, the categories of each document in the test set are estimated and compared to the predefined categories, and the value of *TP, FP, FN*, or *TN* is incremented by 1 depending on the comparison, as shown in Table 2. *T* and *F* represent "True" and "False", respectively.The most common performance measures which are widely used to evaluate TC systems are *precision* (p) and *recall* (r). For a category l, the p and r measures are calculated as

$$r_l = \frac{TP_l}{TP_l + FN_l}, \tag{21}$$

$$p_l = \frac{TP_l}{TP_l + FP_l}. \tag{22}$$

There is a trade-off between precision and recall: if the prediction of each category is *true* for each test example, then the classifier will obtain perfect precision and low recall. Therefore, we employ another measure, called the *F1-score* [33], which measures the performance as a harmonic mean of r and p. $F1$ for a category l is therefore calculated as

$$F1_l = 2\frac{r_l \times p_l}{r_l + p_l} = 2\frac{TP_l}{2TP_l + FN_l + FP_l}. \tag{23}$$

For the global classifier performance overall, two measures—Macro-averaged F1 and Micro-averaged F1—are the most widely used, particularly for measuring multi-label classification performance. For simplicity, we call these measures *MacroF1* and *MicroF1*, respectively. MacroF1 is the mean of F1 over all categories:

$$\text{MacroF1} = \frac{1}{M}\sum_l F1_l, \tag{24}$$

while MicroF1 is calculated as the global F1 for the total *TP, FP*, and *FN* values, as follows:

$$\text{MicroF1} = 2\frac{\sum_l TP_l}{2\sum_l TP_l + \sum_l FN_l + \sum_l FP_l}. \tag{25}$$

In this study, we use MacroF1 and MicroF1 to evaluate classification performance.

7. Experiments and results

7.1. Datasets

Reuters-21578: The *Reuters*-21578 benchmark is widely used for TC evaluation. For this study, we used *ModApte split,*[1] which includes 12,902 documents divided into 3299 for testing and 9603 for training. Of the 135 potential categories, we used only the 90 categories for which there was at least one training document and one testing document. This set was prepared by [28].[2] For simplicity, we refer to the *Reuters*-21578-*ModApte*-90cat as "*R90*". In addition, we adopted a subset of *R90, R10,* which includes the ten most frequent categories: *wheat, acq, trade, ship, interest, grain, crude, earn, money-fx*, and *corn*.

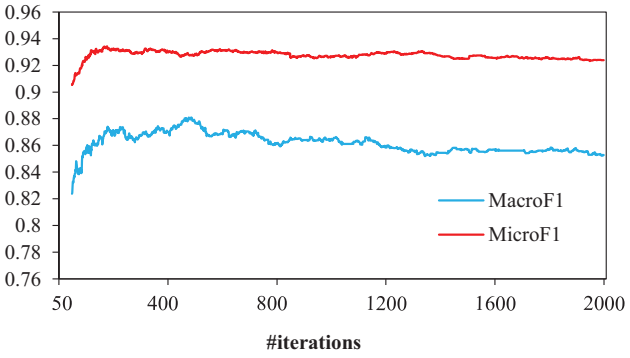

Fig. 2. AdaBoost.MH performance on the *R10* dataset for all boosting rounds.

WebKB: This is a multi-label benchmark used for evaluating TC systems. *WebKB* is a collection of Web pages collected from university computer science departments, including the departments at Cornell University, University of Texas, University of Washington, and University of Wisconsin. The Web pages have been manually classified into the following categories: student, faculty, staff, course, project, department, and other. In this paper, we use a *WebKB-Top4* subset[3] (W4) which contains 4199 pages of the four most populous entity-representing categories: *student, faculty, course*, and *project*.

20-Newsgroups (20NG): This is a collection of approximately 20,000 documents which are evenly partitioned across 20 different newsgroups. *20NG* is a popular benchmark for experiments in text applications of machine learning techniques, such as TC and text clustering. In this paper, we use the "by date" version,[4] which contains 18,825[5] documents sorted by date into training (60%) and test (40%) sets.

OHSUMED: This set includes medical abstracts from the MeSH9 categories for the year 1991. Joachims [21] used the first 20,000 documents, which were divided into 10,000 for training and 10,000 for testing The particular task was to categorise the 23 cardiovascular disease categories. After selecting this category subset, the number of documents reduces to 13,929 (6286 for training and 7643 for testing). In this study, we used the version prepared by [28].

Medical: The *Medical* dataset [30] is composed of documents with free-text summaries of patient symptom histories and prognoses which are used to predict insurance codes. We used the preprocessed version of this dataset by [40] in this experiment.

TMC2007: This is a text dataset for the SIAM Text Mining Competition 2007. It contains 28,596 text samples, each of which belongs to one or more of 22 categories.

RCV1-v2: This is a popular benchmark for TC which consists of 804,414 news stories produced by Reuters between 20 August 1996 and 19 August 1997 [24]. In this experiment, we used the RCV1-Subset1[6] prepared by [40], which consists of 6000 examples be-

[1] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

[2] http://disi.unitn.it/moschitti/corpora/Reuters21578-Apte-90Cat.tar.gz.

[3] http://web.ist.utl.pt/acardoso/datasets/.

[4] http://qwone.com/~jason/20Newsgroups/.

[5] The dataset size reported in http://qwone.com/~jason/20Newsgroups/ is 18,846 documents, but the actual size is 18,825 documents.

[6] http://mulan.sourceforge.net/datasets-mlc.html.

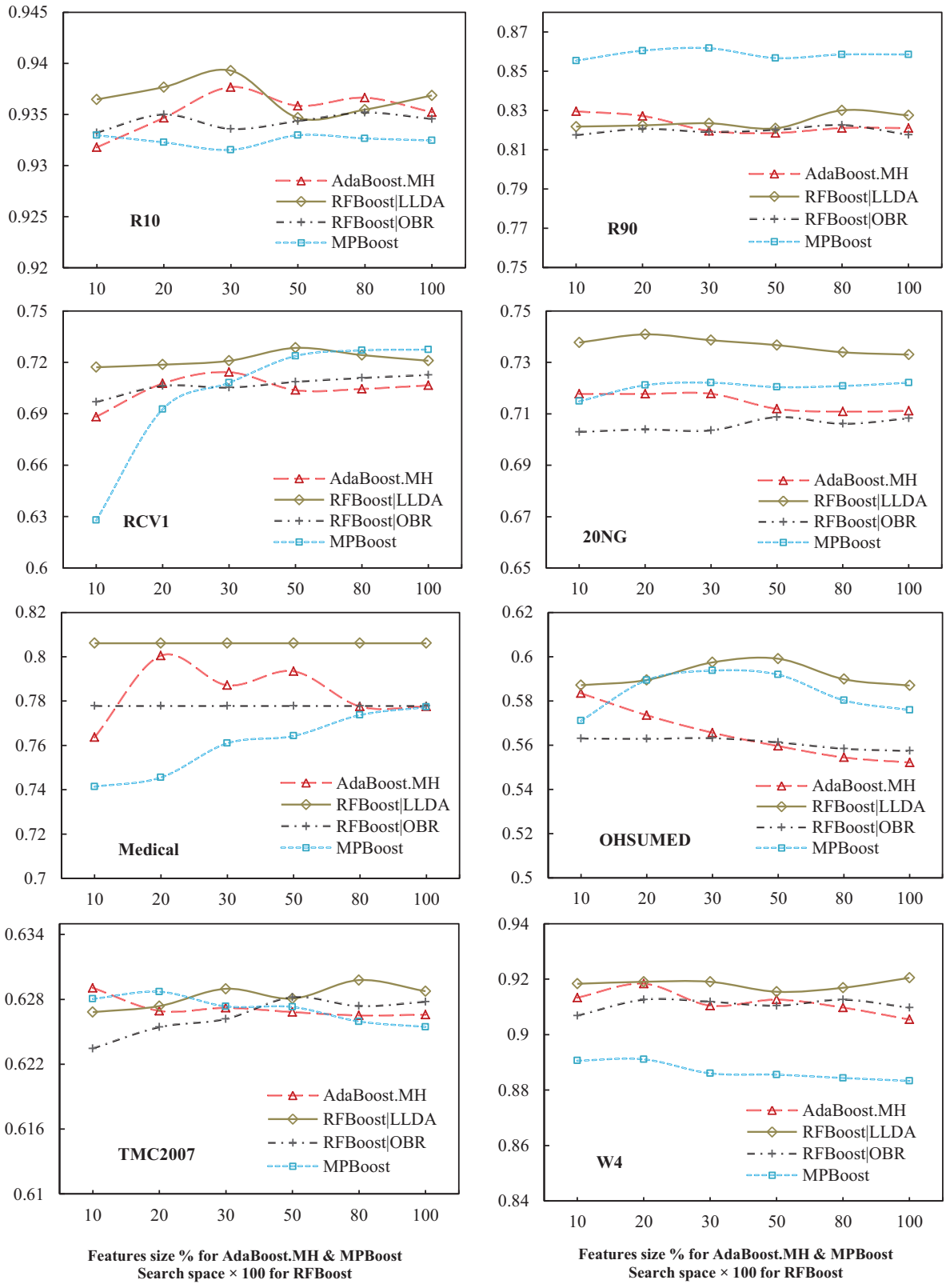

Fig. 3. The MicroF1 results for all datasets.

Table 3
Dataset statistics.

Size/number of	R10	R90	W4	20NG	OHUSMED	Medical	TMC2007	RCV1
Training set	7,194	9,603	2,803	9,840	6,286	333	21,519	3,000
Test set	2,787	3,299	1,396	6,871	7,643	645	7,077	3,000
Labels	10	90	4	20	23	45	22	101
Features	23,578	25,913	7,154	117,227	29,422	1,449	49,060	47,236

Table 4
The best MicroF1 results (%) with and without using LLDA-based feature selection.

Dataset	AdaBoost.MH			MPBoost		
	With FS	BOW%	Without FS	With FS	BOW%	Without FS
R10	93.77	30	93.52	93.3	10	93.25
R90	82.95	10	82.1	86.16	30	85.85
RCV1	71.42	30	70.66	72.7	80	72.74
20NG	71.78	30	71.11	72.21	30	72.21
MEDICAL	80.05	20	77.76	77.37	80	77.73
OHSUMED	58.34	10	55.21	59.37	30	57.59
TMC2007	62.91	10	62.66	62.87	20	62.55
W4	91.83	20	90.54	89.11	20	88.33

longing to 101 categories; the examples are divided into 3000 for training and 3000 for testing.

The datasets are summarised in Table 3.

7.2. Experimental settings

Text preprocessing is a key component in a typical TC framework. The typical text preprocessing tasks which we performed in this study include tokenisation, normalisation, stemming, and stopword removal. Then the weak learning was performed for each boosting algorithm, up to a maximum of 2000 iterations.

The performance of RFBoost with the proposed methods for feature ranking were compared with the original AdaBoost.MH and MPBoost [11], an improved version of AdaBoost.MH which returns multi-pivot terms in each boosting round. A recent comparative study [2] showed that MPBoost is the most effective and efficient algorithm among eight boosting algorithms when BOW is used for text representation and, as mentioned earlier, we used BOW for the text representation in this study. We developed a boosting-based TC system in Java which includes AdaBoost.MH, RFBoost, text preprocessing, and all algorithms presented in this paper. We evaluated MPBoost using MP-Boost++, C++ software developed by Esuli et al. [11] which is available online.[7] We used our proposed weighting policies for RFBoost and the default weighting policies for both AdaBoost.MH and MPBoost.

Our evaluation studied the impact of using LLDA-based feature selection on the classification performance and learning times of AdaBoost.MH and MPBoost, with different percentages used to determine the number of top-ranked features: 10%, 20%, 30%, 50%, 80%, and 100% of the total number of features for each dataset. In the case where 100% of the total features were used, no feature reduction was performed. We evaluated RFBoost with different sizes for the weak learning search space: 100, 200, 300, 500, 800, and 1000 ranked features, corresponding to each subset of features used for the other two algorithms.

We measured the classification performance using both MicroF1 and MacroF1. For the comparative evaluation, we used the best result obtained by any boosting round during the 2000 iterations. For example, AdaBoost.MH's MacroF1 and MicroF1 results from 50 to 2000 iterations for the *R10* dataset are illustrated in Fig. 2. The result at each iteration r is obtained using a subset of weak hypotheses of size r, starting with the first hypothesis at the first boosting round. It is clear that the best results were not obtained at exactly 2000 iterations. Therefore, for all algorithms, we sorted the results for all boosting rounds and only used the best results for the evaluation.

7.3. Results and discussion

Figs. 3 and 4 illustrate the MicroF1 and MacroF1 results, respectively, of the boosting algorithms on all datasets, using varying numbers of selected features for AdaBoost.MH and MPBoost and different sizes of the search space for RFBoost weak learning. Regarding the MicroF1 results, as Fig. 3 shows, on average RFBoost with LLDA-based feature ranking outperformed the other boosting algorithms on all datasets except for *R9*, where MPBoost achieved the best results. RFBoost with OBR-based feature ranking exhibited the worst performance in general, except for the *R10, Medical*, and *W4* datasets, where it slightly outperformed MPBoost, while AdaBoost.MH exceeded the performance of RFBoost with OBR-based feature ranking on all datasets. AdaBoost.MH performed well on average, while MPBoost outperformed it on the *R90, 20NG, RCV1*, and *OHSUMED* datasets. The MacroF1 results in Fig. 4 support these findings. RFBoost with LLDA-based feature ranking obtained the best MacroF1 results on all datasets except for *R90*, where MPBoost outperformed it. RFBoost with OBR-based feature ranking performed well with respect to MacroF1. It outperformed MPBoost on the *R10, RCV1*, and *W4* datasets, and it outperformed AdaBoost.MH on the *R10* and *RCV1* datasets.

Regarding the use of LLDA-based feature selection for improving the classification performance of both AdaBoost.MH and MPBoost, Table 4 summarises the best MicroF1 results for both algorithms on all datasets with and without LLDA-based feature selection. It is clear that using feature selection yielded AdaBoost.MH's best performance on all datasets. For MPBoost, the use of feature selection led to the best performance in general, while for the *RCV1* and *Medical* datasets, MPBoost achieved the best performance without using LLDA-based feature selection. Although the use of feature selection slightly improves the classification performance, the significant impact of using it is amply demonstrated in the learning time, as will be discussed at the end of this section.

Tables 5 and 6 summarise the best MicroF1 and MacroF1 results, respectively, for all of the evaluated boosting algorithms on all datasets. RFBoost with LLDA-based feature ranking achieved

[7] http://www.esuli.it/software/mpboost/.

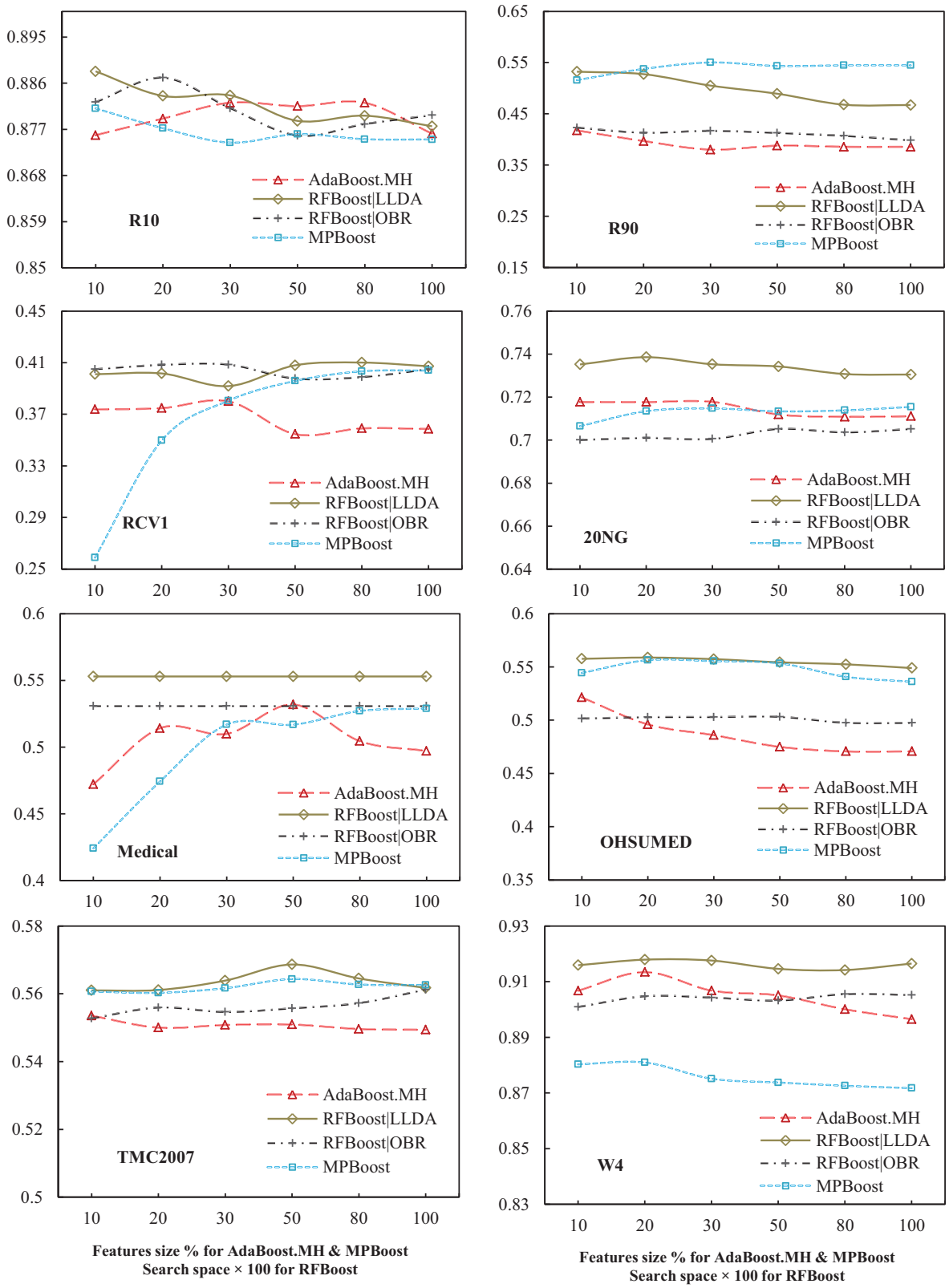

Fig. 4. The MacroF1 results for all datasets.

the best classification performance on all datasets except for *R90*, where MPBoost outperformed it as measured by both the MicroF1 and MacroF1 results, and *W4*, where AdaBoost.MH obtained the best results among all the algorithms. AdaBoost.MH ranked second in terms of MicroF1, behind MPBoost. RFBoost with OBR ranked the worst overall.

In order to statistically validate the significant improvement of our proposed approach as shown in Table 5 and 6, we used the

Algorithm 1 AdaBoost.MH.

INPUT: *A training set* **S**, *probability distribution matrix* **W**, *and number of iterations R*

OUTPUT: *The final hypothesis* $H(x, l) = \sum_{r=1}^{R} \alpha^{(r)} h^{(r)}(x, l)$

1: **set** $\mathbf{W}^{(1)} \leftarrow \mathbf{W}$
2: **for** $r \leftarrow 1$ *to* **R**
3: **pass** $(\mathbf{S}, \mathbf{W}^{(r)})$ *to a base learner*
4: **get** *weak hypothesis* $h^{(r)} : \chi \times \mathcal{Y} \rightarrow \mathbb{R}$
5: **choose** $\alpha^{(r)} \in \mathbb{R}^{+}$
6: **update** $\mathbf{W}^{(r+1)}$ (Eq. (4))

Table 5
The best MicroF1 results (%) overall.

Dataset	AdaBoost.MH	RFBoost|LLDA	RFBoost|OBR	MPBoost
R10	93.77 (2)	93.93 (1)	93.52 (3)	93.30 (4)
R90	82.95 (3)	83.00 (2)	82.26 (4)	86.16 (1)
RCV1	71.42 (3)	72.84 (1)	71.27 (4)	72.74 (2)
20NG	71.78 (3)	74.09 (1)	70.87 (4)	72.21 (2)
MEDICAL	80.05 (2)	80.61 (1)	77.78 (3)	77.73 (4)
OHSUMED	58.34 (3)	59.91 (1)	56.32 (4)	59.37 (2)
TMC2007	62.91 (2)	62.98 (1)	62.82 (4)	62.87 (3)
W4	91.83 (2)	92.05 (1)	91.26 (3)	89.11 (4)
Mean of rank	2.5	1.125	3.625	2.75

Friedman test. The Friedman rank test is defined as follows:

$$\mathcal{X}_F^2 = \frac{12N_d}{k(k-1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{26}$$

where N_d is the number of datasets, k is the number of classification algorithms, and R_j is the average rank of each algorithm.

After performing the Friedman test on the given ranks in Tables 5 and 6 and obtaining the distribution according to Eq. (26) with $k-1$ degrees of freedom, the p-values using MicroF1 and MacroF1 are 0.001 and 0.006, respectively, at the 5% significance level. This means that there are significant differences between the methods, and thus the *null hypothesis* that the methods have the same performance can be rejected.

Having rejected the null hypothesis through the Friedman test, we proceeded with multiple pairwise comparisons of the boosting algorithms. We performed a two-tailed Bonferronni–Dunn test to compare pairs of methods. Table 7 shows the p-values of the compared pairs using the Bonferronni–Dunn test, with boldface indicating strong significant differences. The results demonstrate that RFBoost with LLDA-based feature ranking differs significantly from the other algorithms in terms of the MacroF1 results. The results showed similar findings for MicroF1, with the exception of AdaBoost.MH. For the rest of the pairs, there are no significant differences in terms of MacroF1 and MicroF1 for AdaBoost.MH, RFBoost with OBR-based feature ranking, and MPBoost.

Finally, we turn to computational cost. Let n be the number of training examples, m the number of categories, and v the training vocabulary size. For AdaBoost.MH, the training time for performing one boosting round and building a new weak hypothesis is $O(nmv)$. For RFBoost, the number of features examined at each round is k,

Table 6
The best MacroF1 results (%) overall.

Dataset	AdaBoost.MH	RFBoost|LLDA	RFBoost|OBR	MPBoost
R10	88.22 (3)	88.83 (1)	88.71 (2)	88.11 (4)
R90	41.76 (4)	53.22 (2)	42.31 (3)	55.02 (1)
RCV1	38.00 (4)	41.01 (1)	40.85 (2)	40.39 (3)
20NG	71.78 (2)	73.86 (1)	70.52 (4)	71.54 (3)
MEDICAL	53.19 (2)	54.35 (1)	51.74 (4)	52.89 (3)
OHSUMED	52.14 (3)	55.88 (1)	50.32 (4)	55.64 (2)
TMC2007	55.35 (4)	56.86 (1)	56.12 (3)	56.44 (2)
W4	91.34 (2)	91.79 (1)	90.56 (3)	88.10 (4)
Mean of rank	3	1.125	3.125	2.75

Table 7
Pairwise comparisons using the two-tailed Bonferronni–Dunn test after the Friedman test with $\alpha = 0.05$ and *critical value* $= 7.815$.

	MicroF1	MacroF1
Friedman test p-value	0.001	0.006
RFBoost|LLDA–AdaBoost.MH	0.015	0.001
RFBoost|LLDA–MPBoost	0.004	0.004
RFBoost|LLDA–RFBoost|OBR	<0.0001	0.0004
AdaBoost.MH–MP-Boost	0.660	0.980
AdaBoost.MH–RFBoost|OBR	0.048	0.660
MPBoost–RFBoost|OBR	0.123	0.509
Bonferroni-corrected significance level	0.0083	0.0083

Algorithm 3 *Per-category* normalisation procedure.

INPUT: *A non-normalised distribution matrix* **W**
OUTPUT: *A normalised distribution matrix* **W**

1: **for** $l \leftarrow 1$ *to* m **do**
2: **set** $w_l \leftarrow 0$
3: **for** $i \leftarrow 1$ *to* n do
4: $w_l \leftarrow w_l + w_{i,l}$
5: **end for**
6: **end for**
7: $w_l \leftarrow w_l * m$
8: **for** $i \leftarrow 1$ *to* n **do**
9: $w_{i,l} \leftarrow w_{i,l} / w_l$
10: **end for**

Algorithm 2 AdaBoost.MH with real-valued predictions.

INPUT: *A training set* **S**, *distribution matrix* **W**, *number of iterations R, and feature term index* $\mathcal{T}$
OUTPUT: *The final hypothesis* $H(x, l) \leftarrow \sum_{r=1}^{R} h^{(r)}(x, l)$

1: **set** $\mathbf{W}^{(1)} \leftarrow \mathbf{W}$ ▷ start with an initial distribution of weights
2: **for** $r \leftarrow 1$ to **R do** ▷ for each boosting round r
3: **set** $pt \leftarrow t_1$ ▷ initialise the pivot term
4: **set** $Z_{min} \leftarrow 99,999$ ▷ initialise Z to a large positive number
5: **for** $i \leftarrow 1$ to $|\mathcal{T}|$ **do** ▷ for each feature in $\mathcal{T}$
6: **calculate** $Z_i^{(r)}$ ▷ Eqs. (9)–(14)
7: **if** $Z_i^{(r)} < Z_{min}$ **then**
8: $Z_{min} \leftarrow Z_i^{(r)}$
9: $pt \leftarrow t_i$
10: **end if**
11: **end for**
12: **return** $h^{(r)}$ ▷ build the new weak hypothesis based on pt
13: predictions (Eq. (8))
14: **update** $\mathbf{W}^{(r+1)}$ ▷ update the distribution matrix (Eq. (4))
15: **end for**

Algorithm 4 RFBoost.

INPUT: *A training set* $\mathbf{S}$, *distribution matrix* $\mathbf{W}$, *number of iterations* R, *ranked feature term index* $\mathcal{T}'$, *and number of filtered features* $\boldsymbol{k}$
OUTPUT: *The final hypothesis* $H(\mathrm{x}, l) \leftarrow \sum_{r=1}^{R} h^{(r)}(\mathrm{x}, l)$

set $\mathbf{W}^{(1)} \leftarrow \mathbf{W}$ ▷ start with an initial distribution of weights
set *FilterdList* $\leftarrow (t'_1, \ldots, t'_k)$ ▷ filter the top $\boldsymbol{k}$ ranked features
for $r \leftarrow 1$ to $\boldsymbol{R}$ **do** ▷ for each boosting round r
  set $pt \leftarrow t'_1$ ▷ initialise the pivot term
  set $Z_{min} \leftarrow 99,999$ ▷ initialise Z to a large positive number
  for $k \leftarrow 1$ to $\boldsymbol{k}$ **do** ▷ for each feature in *FilterdList*
    calculate $Z_k^{(r)}$ ▷ Eq. (9) to (14)
    if $Z_k^{(r)} < Z_{min}$ **then**
      $Z_{min} \leftarrow Z_k^{(r)}$ ▷ select the pivot term, the one with the smallest Z
      $pt \leftarrow t'_k$
    end if
  end for
  return $h^{(r)}$ ▷ build the new weak hypothesis based on pt predictions
  update $\mathbf{W}^{(r+1)}$ ▷ update the distribution matrix
  FilterdList $\leftarrow$ *FilterdList* $-$ pt ▷ remove the current pivot term
  FilterdList $\leftarrow$ *FilterdList* $+ t'_{k+r}$ ▷ add t'_{k+r}, the next feature in $\mathcal{T}'$
end for

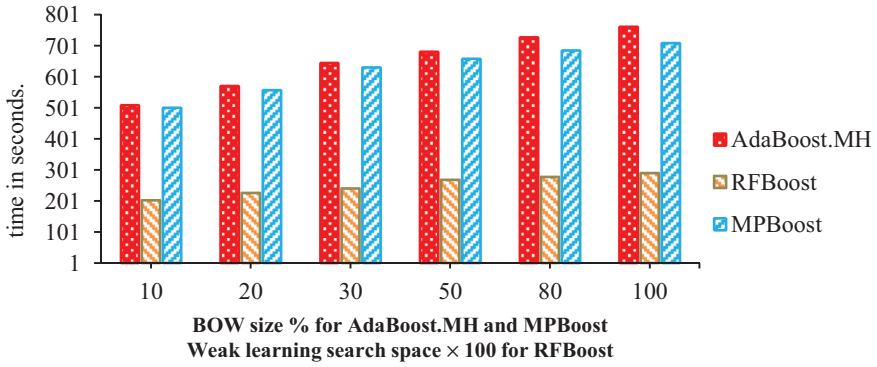

Fig. 5. The training time averages for all datasets.

Algorithm 5 The feature-ranking procedure based on one boosting round.

INPUT: *A training set* $\mathbf{S}$, *distribution matrix* $\mathbf{W}$, *and feature term index* $\mathcal{T}$
OUTPUT: *The ranked feature term index* $\mathcal{T}'$

set $\epsilon \leftarrow 1 \backslash mn$
define *an empty index* $\mathcal{T}'$
for $k \leftarrow 1$ to $|\mathcal{T}|$ **do**
  for $l \leftarrow 1$ to m **do**
    for $u \leftarrow 0$ to 1 **do**
      divide $\mathcal{X}$ *into* X_0, X_1 (Eq. (9))
      for $b \leftarrow -1, +1$ **do**
        calculate W_{ul}^{b} (Eqs. (10) and (11))
        calculate Z_k (Eq. (13))
        add *the pair* (t_k, Z_k) *to* $\mathcal{T}'$
      end for
    end for
  end for
end for
sort $\mathcal{T}'$*based on the values of* Z

Algorithm 6 The feature-ranking procedure based on LLDA.

INPUT: *A training set* $\mathbf{S}$, *feature terms index* $\mathcal{T}$, *and label–word distribution* Φ
OUTPUT: *The ranked feature terms index* T^{*}

define *an empty index* T^{*}
for $k \leftarrow 1$ *to* $|\mathcal{T}|$ **do**
  set $\phi_{max.t_k} \leftarrow \phi_{1.t_k}$
  for $l \leftarrow 2$ *to* m **do**
    if $\phi_{l.t_k} > \phi_{max.t_k}$
      $\phi_{max.t_k} \leftarrow \phi_{l.t_k}$
    end if
  end for
  add the pair$(t_k, \phi_{max.t_k})$ to $\mathcal{T}'$
end for
sort $\mathcal{T}'$based on $\phi_{max,\,t}$ $\forall t \in \mathcal{T}$

the number of top-ranked features, which is a very small number compared to the training vocabulary size v. Therefore, RFBoost's training time for performing one boosting round and building a new weak hypothesis is $O(nmk)$. Fig. 5 illustrates the learning time average for all algorithms on all datasets. It is clear that RFBoost is dramatically faster than both AdaBoost.MH and MPBoost. Using LLDA-based feature selection dramatically accelerates the learning of both AdaBoost.MH and MPBoost, as shown by the last columns in Fig. 5, where the feature size of 100% indicates that all training features have been used, without any reduction. MPBoost is slightly faster than AdaBoost.MH. However, theoretically, MPBoost is slower than AdaBoost.MH. That is, in MPBoost the weak learner returns multi-pivots, one for each label, which makes it slower than AdaBoost.MH. The reason behind the slight differences in efficiency is due to the different programming languages that are used, Java and C++, as mentioned in Section 7.2. It is also worth mentioning that RFBoost was evaluated without using any feature reduction, as it is expressly designed to handle the feature dimensionality by itself.

8. Conclusions

In this study, we have proposed RFBoost, an improved version of AdaBoost.MH, the well-known boosting algorithm for multi-label classification. The idea behind RFBoost is to rank the features extracted from the training set as a pre-task before weak

Algorithm 7 LLDA-based feature selection.

INPUT: *A training set* $\boldsymbol{S}$, *feature term index* $\mathcal{T}$, *label–word distribution* Φ, *and number of top words u*
OUTPUT: *A sorted and weighted feature term index* $\mathcal{T}''$

define *a temporary index* $\mathcal{T}''_{tmp}$
define *an empty index* $\mathcal{T}''$
for $k \leftarrow 1$ *to* $|\mathcal{T}|$ **do**
  set $\phi_{max.t_k} \leftarrow \phi_{1.t_k}$
  for $l \leftarrow 2$ *to* m **do**
    if $\phi_{l.t_k} > \phi_{max.t_k}$ **then**
      set $\phi_{max.t_k} \leftarrow \phi_{l.t_k}$
    end if
  end for
  add *the pair*$(t_k, \phi_{max.t_k})$ to $\mathcal{T}''_{tmp}$
end for
sort $\mathcal{T}''_{tmp}$*based on* $\phi_{max.\ t}\ \forall t \in \mathcal{T}$
for $i \leftarrow 1$ *to* u **do**
add $(t_i, \phi_{max.t_i})$ to $\mathcal{T}''$

learning. Then a small fixed number of top-ranked features are filtered from all the ranked features, and only these are examined during each boosting round—rather than examining all of the features, as AdaBoost.MH does. We have proposed two methods for ranking the features: *one boosting round* (OBR) and *Labeled Latent Dirichlet Allocation* (LLDA). The former is based on performing one boosting iteration on a given initial distribution and then weighting and sorting the features according to their *boosting weights*. The latter is based on performing LLDA based on Gibbs sampling to obtain two indexes, the *label–word* assignment index and the *document–label* assignment index. Then the words are ranked according to their conditional probabilities in the label–word assignment index. In addition, we have proposed new weighting policies for initialising a distribution of weights over the training examples and for normalising the distribution at each boosting round. To initialise the distribution, we used the conditional probabilities of the documents in a document–label assignment index obtained by LLDA as the initial weights. For normalising the distribution, we used per-category normalisation rather than the usual per-document normalisation. We also investigated the use of LLDA as a feature selection method by simply using the top-weighted words based on the maximal conditional probabilities of the words across the labels in the label–word assignment index.

The experimental results for eight benchmarks prove that RFBoost is more efficient than AdaBoost.MH and MPBoost. In terms of the classification performance, the results demonstrate that RFBoost with the new weighting policies and the LLDA-based feature ranking significantly outperformed all other evaluated algorithms, while OBR-based feature ranking yielded the worst performance overall. However, the advantage of OBR-based feature ranking is that the features are ranked within the weak learner and there is no need to use external ranking methods. In the future, we will investigate the use of RFBoost to solve other multi-label classification problems. In addition, we will investigate the use of other feature ranking methods, as feature ranking is the core idea for improving RFBoost's effectiveness.

Acknowledgement

This research was partially supported by the Malaysia Ministry of Education Grant FRGS/1/2014/ICT02/UKM/01/1 awarded to the Center for Artificial Intelligence Technology at the Universiti Kebangsaan Malaysia.

References

[1] B. Al-Salemi, M.J. Ab Aziz, Statistical Bayesian learning for automatic Arabic text categorization, J. Comput. Sci. 7 (2010) 39.
[2] B. Al-Salemi, M.J. Ab Aziz, S.A. Noah, Boosting algorithms with topic modeling for multi-label text categorization: a comparative empirical study, J. Inf. Sci. 41 (2015) 732–746.
[3] B. Al-Salemi, M.J. Ab Aziz, S.A. Noah, LDA-AdaBoost.MH: accelerated AdaBoost.MH based on latent Dirichlet allocation for text categorization, J. Inf. Sci. 41 (2015) 27–40.
[4] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2002) 235–256.
[5] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[6] S. Bloehdorn, A. Hotho, Boosting for Text Classification with Semantic Features, in: B. Mobasher, O. Nasraoui, B. Liu, B. Masand (Eds.), Advances in Web Mining and Web Usage Analysis, Springer, 2006, pp. 149–166.
[7] R. Busa-Fekete, B. Kégl, Accelerating AdaBoost using UCB, J. Mach. Learn. Res.: Proc. Track 7 (2009) 111–122.
[8] F. Colace, M. De Santo, L. Greco, P. Napoletano, Text classification using a few labeled examples, Comput. Hum. Behav. 30 (2014) 689–697.
[9] F. De Comité, R. Gilleron, M. Tommasi, Learning multi-label alternating decision trees from texts and data, in: Proceedings of Machine Learning and Data Mining in Pattern Recognition, 2003, pp. 35–49.
[10] G. Escudero, L. Mrquez, G. Rigau, Boosting applied to word sense disambiguation, in: Proceedings of the 11th European Conference on Machine Learning, 2000, pp. 129–141.
[11] A. Esuli, T. Fagni, F. Sebastiani, MP-Boost: A multiple-pivot boosting algorithm and its application to text categorization, in: Proceedings of International Symposium on String Processing and Information Retrieval, 2006, pp. 1–12.
[12] D. Feng, F. Chen, W. Xu, Supervised feature subset selection with ordinal optimization, Knowledge-Based Syst. 56 (2014) 123–140.
[13] A.J. Ferreira, M.A.T. Figueiredo, Boosting algorithms: a review of methods, theory, and applications, Ensemble Machine Learning: Methods and Applications, 35, Springer US, 2012.
[14] Y. Freund, L. Mason, in: Proceedings of ICML, 1999, pp. 124–133.
[15] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: P. Vitányi (Ed.), Computational Learning Theory: Second European Conference, EuroCOLT '95 Barcelona, Spain, March 13–15, 1995 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 23–37.
[16] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[17] N. García-Pedrajas, A. de Haro-García, Boosting instance selection algorithms, Knowledge-Based Syst. 67 (2014) 342–360.
[18] T.L. Griffiths, M. Steyvers, in: Finding scientific topics, 101, 2004, pp. 5228–5235.
[19] T. Hastie, R. Tibshirani, J. Friedman, Boosting and additive trees, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2009, pp. 337–387.
[20] G. Heinrich, Parameter estimation for text analysis, University of Leipzig, Tech. Rep., 2005.
[21] T. Joachims, Text Categorization With Support Vector Machines: Learning With Many Relevant Features, Springer, 1998.
[22] Kégl, B. The return of AdaBoost. MH: multi-class Hamming trees. In: International Conference on Learning Representations (ICLR) (2014). URL http://arxiv.org/abs/1312.6086.
[23] B. Kégl, R. Busa-Fekete, Boosting products of base classifiers, in: Proceedings of the 26th Annual International Conference on Machine Learning, Publishing, 2009, pp. 497–504.
[24] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, RCV1: a new benchmark collection for text categorization research, J. Mach. Learn. Res 5 (2004) 361–397.
[25] P. Li, H. Li, M. Wu, Multi-label ensemble based on variable pairwise constraint projection, Inf. Sci. 222 (2013) 269–281.
[26] H. Liu, S. Zhang, X. Wu, MLSLR: multilabel learning via sparse logistic regression, Inf. Sci. (2014).
[27] J.D. Mcauliffe, D.M. Blei, Supervised topic models, in: Proceedings of Advances in Neural Information Processing Systems, 2008, pp. 121–128.

[28] A. Moschitti, R. Basili, Complex linguistic features for text classification: a comprehensive study, in: Proceedings of Advances in Information Retrieval, 2004, pp. 181–196.
[29] I. Mukherjee, R.E. Schapire, A theory of multiclass boosting, J. Mach. Learn. Res. 14 (2013) 437–497.
[30] J.P. Pestian, C. Brew, P. Matykiewicz, D. Hovermale, N. Johnson, K.B. Cohen, W. Duch, A shared task involving multi-label classification of clinical free text, in: Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing, 2007, pp. 97–104.
[31] D. Ramage, D. Hall, R. Nallapati, C.D. Manning, Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1, 2009, pp. 248–256.
[32] E. Rashedi, A. Mirzaei, A hierarchical clusterer ensemble method based on boosting theory, Knowledge-Based Syst. 45 (2013) 83–93.
[33] C.J.V. Rijsbergen, Information Retrieval, Butterworth-Heinemann, 1979.
[34] R.E. Schapire, Y. Freund, Boosting: Foundations and Algorithms, MIT Press, MA, 2012.
[35] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (1999) 297–336.
[36] R.E. Schapire, Y. Singer, BoosTexter: A boosting-based system for text categorization, Mach. Learn. 39 (2000) 135–168.
[37] F. Sebastiani, A. Sperduti, N. Valdambrini, An improved boosting algorithm and its application to text categorization, in: Proceedings of the Ninth International Conference on Information and Knowledge Management, 2000, pp. 78–85.
[38] A.F. Smith, G.O. Roberts, Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods, J. R. Stat. Soc. Ser. B: Methodol. (1993) 3–23.
[39] S. Tasci, T. Gungor, LDA-based keyword selection in text categorization, in: Proceedings of the 24th International Symposium on Computer and Information Sciences, ISCIS, 2009, pp. 230–235.
[40] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas, Mulan: a Java library for multi-label learning, J. Mach. Learn. Res. 12 (2011) 2411–2414.
[41] A.K. Uysal, S. Gunal, A novel probabilistic feature selection method for text classification, Knowledge-Based Syst. 36 (2012) 226–235.
[42] Q. Wu, M.K. Ng, Y. Ye, X. Li, R. Shi, Y. Li, Multi-label collective classification via Markov chain based learning method, Knowledge-Based Syst. 63 (2014) 1–14.
[43] M.-L. Zhang, J.M. Peña, V. Robles, Feature selection for multi-label naive Bayes classification, Inf. Sci. 179 (2009) 3218–3229.
[44] M.-L. Zhang, Z.-H. Zhou, ML-KNN: A lazy learning approach to multi-label learning, Pattern Recognit. 40 (2007) 2038–2048.
[45] Z. Zhang, X.-H. Phan, S. Horiguchi, An efficient feature selection using hidden topic in text categorization, in: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications Workshops, AINAW 2008, 2008, pp. 1223–1228.
[46] D. Zhao, J. He, J. Liu, An improved LDA algorithm for text classification, in: Proceedings of the International Conference on Information Science, Electronics and Electrical Engineering (ISEEE), 2014, pp. 217–221.
[47] J. Zhu, S. Rosset, H. Zou, T. Hastie, Multi-class adaboost, Ann Arbor 1001 (2006) , 1612.
[48] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class adaboost, Stat. Interface 2 (3) (2009) 349–360.
[49] H. Zou, J. Zhu, T. Hastie, New multicategory boosting algorithms based on multicategory fisher-consistent losses, Ann. Appl. Stat. 2 (2008) 1290–1306.