A Meta-Top-Down Method for Large-Scale Hierarchical Classification

Xiao-Lin Wang, Hai Zhao, and Bao-Liang Lu, *Senior Member*, *IEEE*

Abstract—Recent large-scale hierarchical classification tasks typically have tens of thousands of classes on which the most widely used approach to multiclass classification—one-versus-rest—becomes intractable due to computational complexity. The top-down methods are usually adopted instead, but they are less accurate because of the so-called error-propagation problem in their classifying phase. To address this problem, this paper proposes a meta-top-down method that employs metaclassification to enhance the normal top-down classifying procedure. The proposed method is first analyzed theoretically on complexity and accuracy, and then applied to five real-world large-scale data sets. The experimental results indicate that the classification accuracy is largely improved, while the increased time costs are smaller than most of the existing approaches.

Index Terms—Large-scale hierarchical classification, metalearning, ensemble learning, metaclassification, top-down method, text classification

1 Introduction

Multiclass classification—classifying samples into multiple predefined classes—is a fundamental task in both machine learning and data mining domains [1]. For example, each document in the Reuters-21578[1] corpus is assigned one or more labels from 120 predefined classes such as business, sports, and military.

The ensemble method of one-versus-rest is the most widely adopted solution for multiclass classification (see Fig. 1a) [2], [3]. First a binary-class classifier f_i $(i = 1, \ldots, n)$, named base classifier, is trained for each class c_i to predict whether an input sample x belongs to this class; then thresholding strategies are employed to decide the predicted labels according to the confidence scores of the base classifiers. Two commonly used thresholding strategies are score-cut (S-cut) that accepts the classes whose scores are larger than a predefined threshold, and rank-cut (R-cut) that accepts the classes whose scores are among the top-r (r is a predefined integer) [2].

In recent years, two types of classification strategies have been developed for multiclass classification problems. One is to reduce the computational complexity on the tasks that have large numbers of predefined classes, such as tens of thousands; usually hierarchies are used to organize the classes, so these tasks are called large-scale hierarchical classification. The examples include categorizing patent documents into the taxonomy of the International Patent Classification [4], [5], [6] and categorizing web pages into the directories of the Open Directory Project or Yahoo! [7], [8].

The top-down methods which organize base classifiers hierarchically are widely adopted for large-scale hierarchical classification (see Fig. 1b) [9], [10], [11], [12], [13], [14]. They classify a sample x by filtering it down a tree of base classifiers f_i $(i = 1, \ldots, p)$ from the root node f_1. For each parent node where this sample arrives, those child nodes whose confidence scores (produced by the base classifiers) pass the thresholding strategies will carry it on. The bottom leaf nodes c_j $(j = 1, \ldots, n)$ where the sample terminates are the predicted labels [8], [11], [12]. The top-down method employing the S-cut strategy, named S-cut top-down (ScutTD), is usually used in multilabeled classification, and the one employing the R-cut strategy $(r = 1)$, named R-cut top-down (RcutTD), is usually used in single-labeled classification.

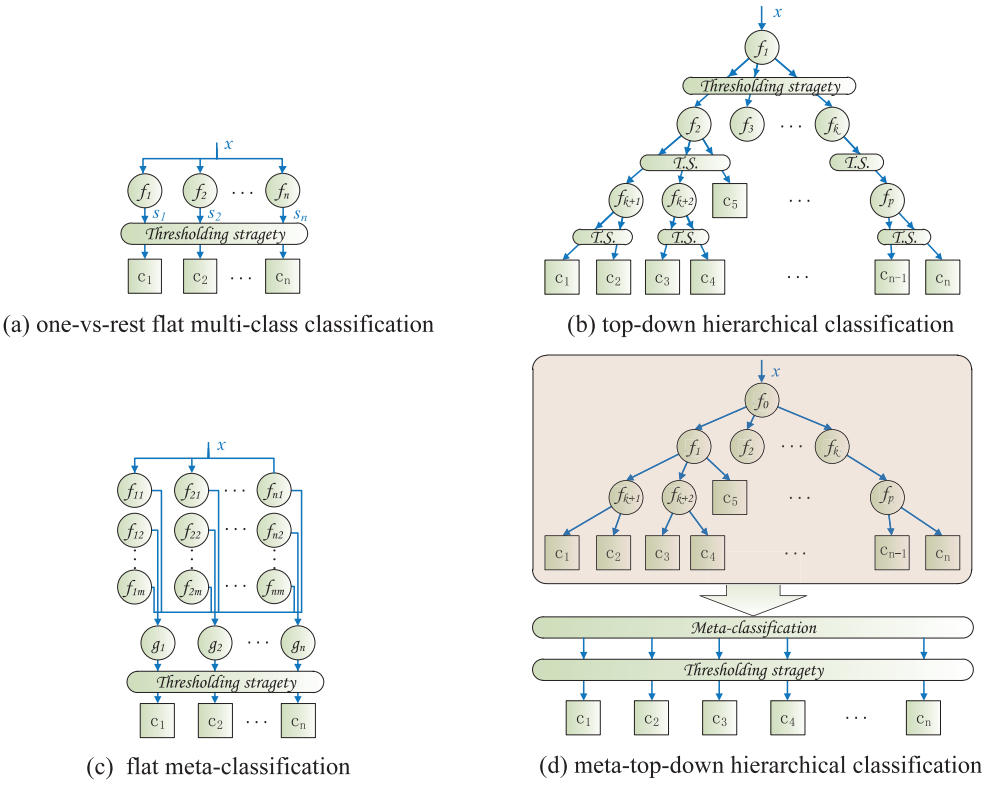
The computational complexity of the top-down methods is the logarithm of the number of classes [8], [15], while that of the one-versus-rest method is linear to the number of classes. As an evidence, in a classification experiment on 492,617 training documents, 275,364 test documents, and 132,199 categories of Yahoo!, the former costs only 2.1 hours for training and 0.12 hours for classifying, while the latter costs 310 hours for training and 54 hours for classifying [8].

The other type of classification strategy on multiclass classification is to raise the classification accuracy of one-versus-rest through metaclassification (see Fig. 1c) [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]. Metaclassification takes the outputs of base classifiers as inputs to better learn target signals. First, base classifiers f_{ij} $(i = 1, \ldots, n, j = 1, \ldots, m)$ are trained for each class c_i by employing different types of classifiers or manipulating features. Then, metaclassifier g_i is trained for each class c_i

1. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

• *X.L. Wang, H. Zhao, and B. L. Lu are with the Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligence Systems, Shanghai Jiao Tong University, Room 431, Dian Xin Building 3, 800 Dong Chuan Road, Min Hang District, Shanghai 200240, P.R. China.*
E-mail: arthur.xl.wang@gmail.com, {zhaohai, blu}@cs.sjtu.edu.cn.

Manuscript received 11 Nov. 2012; revised 29 Jan. 2013; accepted 29 Jan. 2013; published online 13 Feb. 2013.
Recommended for acceptance by B.C. Ooi.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-11-0763.
Digital Object Identifier no. 10.1109/TKDE.2013.30.



(a) one-vs-rest flat multi-class classification (b) top-down hierarchical classification

(c) flat meta-classification (d) meta-top-down hierarchical classification

Fig. 1. The meta-top-down method combines top-down hierarchial classification and metaclassification to achieve high classification accuracy on the tasks that have large numbers of predefined classes.

by learning the predictions of all base classifiers, i.e., an $m \times n$ matrix.

This paper joins these two types of classification strategy through exploring the usage of metaclassification in the top-down methods (see Fig. 1d). To the best of our knowledge, this topic has not been explored before. In this paper, a meta-top-down method (MetaTD) that is more accurate than the normal top-down methods[2] is proposed.

The motivation of MetaTD is that metaclassification can help to raise the classification accuracy of the normal top-down methods, which have a well-known deficiency of classification accuracy. Their accuracy is usually lower than the one-versus-rest method [13], [14], [15], [30]. As an evidence, in the PASCAL[3] challenge on large-scale hierarchical text classification[4] in 2009, flat methods rank highest, hybrid methods rank next, and the normal top-down methods rank the lowest [31].

The accuracy deficiency of the normal top-down methods is mainly caused by their nonrobust classifying procedure that consists of multiple cascaded decisions about which child nodes should be invoked from a parent node. Each of these decisions is made based only on the score of one base classifier, and is not changeable afterward. Thus, any wrong decision inevitably leads to wrong predictions. This problem is so-called error propagation [13], [30]. Sun et al. call a subset of this problem—mistakenly rejecting a child node at high layers—a blocking problem [32]. Liu et al. draw an analog between this procedure and a pachinko machine [8].

To relieve error propagation, the proposed MetaTD employs metaclassification to fuse the scores of all the base classifiers instead of making arbitrary cascaded decisions. The framework is to encode the scores along a root-to-leaf path into a feature vector, and to employ a metaclassifier to predict whether the corresponding leaf node is a correct label.

Note that there are two kinds of hierarchical classification tasks in real-world applications. One kind is mandatory leaf-node classification, where only the leaf nodes are valid labels [33], [34], [35]. In contrast, the other is nonmandatory leaf-node classification, where both the internal nodes and the leaf nodes are valid labels [8], [36]. In this paper, we consider the first kind—mandatory leaf-node classification.

The main contributions of this paper are as follows:

- Propose a meta-top-down method for large-scale hierarchical classification.
- Provide accuracy and complexity analysis on the proposed method.
- Achieve high classification accuracies on five real-world data sets.

The remainder of this paper is organized as follows: The proposed MetaTD and the normal top-down methods are presented and analyzed in Section 2. We then examine them with five real-world data sets in Section 3. After that, related work on top-down methods and metaclassification is discussed in Section 4. Finally, we conclude this paper with a description of future work in Section 5.

2. The normal top-down methods refer to ScutTD and RcutTD ($r = 1$) in this paper.

3. Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL) is an academic organization funded by the European Union. http://www.pascal-network.org.

4. http://lshtc.iit.demokritos.gr/.

2 METHODS

In this section, we first review the normal top-down methods and then present MetaTD in detail. After that an example is provided to illustrate MetaTD. Finally, the classification accuracy and computational complexity of MetaTD are analyzed.

2.1 Normal Top-Down Methods

Suppose H is a hierarchy of classes, which is a set of parent-child relations,

$$H = \{(p, c) \mid p \text{ is a parent node}, \\ c \text{ is one of its child node}\},$$

where (p, c) is called a parent-child relation.

ScutTD consists of the following three steps, while RcutTD omits the second step. Suppose T, D, and E are training, development, and test sets, respectively, where labels of the samples in development sets are available to classification systems for tuning parameters.

First, one base classifier per each parent-child relation (p, c), noted as f_c, is trained through the training set T_{pc} as follows,

$$T_{pc} = \{(x, y) \mid x \in T_p, y = +1 \text{ if } x \in T_c, \\ y = -1 \text{ otherwise}\}, \tag{1}$$

where T_p and T_c are the subsets of T which consist of the samples belonging to node p and c.

Second, the threshold values required by ScutTD are tuned on D. The methods for this step actually have alternatives. Early researchers such as [10], [33] employ a universal threshold value for all the base classifiers at a same layer. In contrast, recent researchers such as [8], [14] employ a different threshold value for each base classifier to achieve high classification accuracy. They take micro-F_1 as the optimization target to balance the precision and recall of a node's prediction, where micro-F_1 is a widely used performance measurement for multilabeled classification [1]. Here, we adopt the recent method, formulated as follows:

$$\begin{aligned} t_c &= \underset{t}{argmax}\, F_1(D_{pc}, f_c, t) \\ &= \underset{t}{argmax} \frac{2P(D_{pc}, f_c, t)R(D_{pc}, f_c, t)}{P(D_{pc}, f_c, t) + R(D_{pc}, f_c, t)}, \\ P(D_{pc}, f_c, t) &= \frac{n_r}{|\{x \mid (x, y) \in D_{pc}, f_c(x) \geq t\}|}, \\ R(D_{pc}, f_c, t) &= \frac{n_r}{|x \mid (x, y) \in D_{pc}, y = 1|}, \\ n_r &= |\{x \mid (x, y) \in D_{pc}, f_c(x) \geq t, y = 1\}|, \end{aligned} \tag{2}$$

where:

- F_1, P, and R are the micro-F_1, precision and recall, respectively;
- t_c is the optimized threshold of node c, and t is a threshold candidate;
- f_c is the base classifier of the node c, and $f_c(x)$ represents the confidence score given by f_c for the sample x;

Require: a test instance x
  a class hierarchy H={$(p, c)|(p, c)$ is a parent-child }
  base-classifiers $\{f_c|(p, c) \in H\}$
  thresholds $\{t_c|(p, c) \in H\}$ (for ScutTD only)
  an integer parameter r (for RcutTD only)
Ensure: a predicted label set y
  $q \leftarrow [Root]$, $y \leftarrow \{\}$
  while q is not empty **do**
    $p \leftarrow$ pop out the first item of q
    if p is a leaf node **then**
      $y \leftarrow y \cup \{p\}$
    else
      for all c, $(p, c) \in H$ **do**
        $s_c \leftarrow f_c(x)$
        ScutTD: add c into q if $s_c \geq t_c$
        RcutTD: add c into q if s_c ranks top-r
      end for
    end if
  end while
  return y

Fig. 2. S-cut and R-cut top-down algorithms.

- D_{pc} is a sample set consisting of the development samples belonging to the node p and those also belonging to the node c are taken as positive, similar to the T_{pc} defined by (1);
- n_r is the number of correctly predicted labels.

Third, the test samples in E are classified through the algorithm presented in Fig. 2. With the trained base classifiers f_c and the thresholds t_c (only required by ScutTD), the test set E can be classified.

2.2 Meta-Top-Down Method

The proposed meta-top-down method employs metaclassification to reclassify samples based on the output of the normal top-down methods. MetaTD takes the confidence scores of the base classifiers along a root-to-leaf path as the metalevel input, and takes whether the leaf node is a correct label as a metalevel target. This metaclassification task is formulated as follows:

$$\begin{aligned} \mathcal{M}(u, l) &= (\mathcal{M}_x(u_x, l), \mathcal{M}_y(u_y, l)), \\ \mathcal{M}_x(u_x, l) &= \{(n_i, f_{n_i}(u_x)) \mid n_i \in p_l\}, \\ \mathcal{M}_y(u_y, l) &= \begin{cases} +1, & l \in u_y \\ -1, & l \notin u_y \end{cases}, \end{aligned} \tag{3}$$

where:

- $\mathcal{M}$ is a metasample that consists of an input $\mathcal{M}_x$ and an output $\mathcal{M}_y$;
- $u = (u_x, u_y)$ is a base sample, here u_x is the input, and u_y is the output set, a set of correct labels;
- l is a leaf node, thus it is also a validate class label for base samples;
- $p_l = \{n_{i_1}, n_{i_2}, \ldots, n_{i_k}\}$ is a path from the root to l, where n_{i_1} is the root, $n_{i_k} = l$, and $(n_{i_a}, n_{i_{a+1}}) \in H$;
- $f_{n_{i_a}}$ is the base classifier of the node n_{i_a}.

However, (3) causes a problem of computational complexity on large-scale hierarchical classification tasks as it

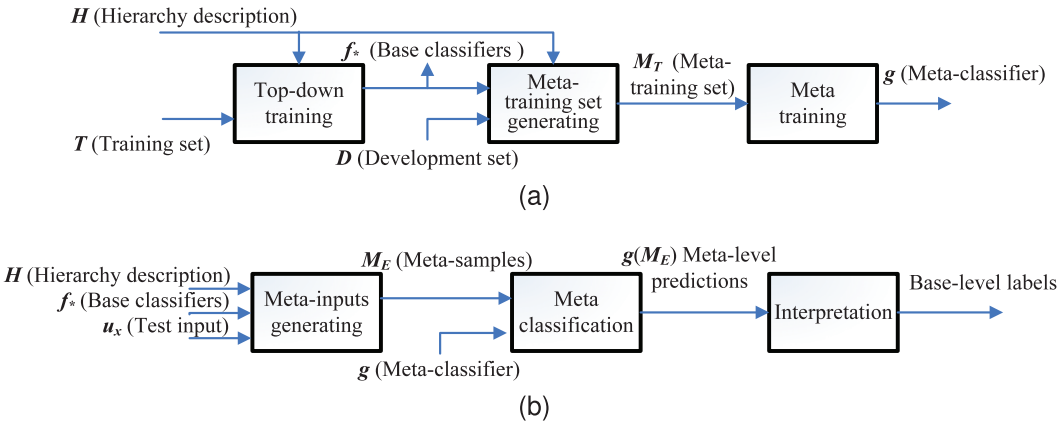


Fig. 3. Workflows of MetaTD: (a) training phase and (b) classifying phase.

produces a metasample for each class. For example, the International Patent Classification (IPC) hierarchy has over 49,187 labels, thus (3) generates over 49,187 metasamples from a single base sample. To address this problem, we propose a pruning method that selects a small number of label candidates for each base sample, noted as $L(u_x)$ (see Section 2.2.2 presents the pruning method).

MetaTD is based on the above settings, and its workflow is presented in Fig. 3. The training phase consists of three steps as follows:

1. Train base classifiers f_c on the training set T, which is the same as the normal top-down methods.
2. Construct a metatraining set with the base classifiers and the development set D through the pruning method $L(u_x)$,

$$M_T = \cup_{u \in D}\{\mathcal{M}(u, l) \mid l \in L(u_x)\}.$$

3. Train a metaclassifier g on M_T.

The whole training phase requires the base-level training set T, the development set D, and the description of the hierarchy H. It produces a base classifier f_c per child node c and a metaclassifier g.

The classifying phase also consists of three steps as follows:

1. Construct a group of metasamples from a test base sample u_x (its label set u_y is unknown),

$$M_E = \{\mathcal{M}_x(u_x, l) \mid l \in L(u_x)\}.$$

2. Apply the metaclassifier g to the metasamples,

$$g(M_E) = \{g(\mathcal{M}_x(u_x, l)) \mid l \in L(u_x)\} = \{g_{u_x,l} \mid l \in L(u_x)\}.$$

3. Interpret the predictions into base-level labels. Thresholding strategies are employed again. In single-labeled classification tasks, the label of the largest metascore is taken as the prediction (R-cut strategy); in multilabeled tasks, the labels whose metascores are larger than the threshold value are taken as the predictions (S-cut).

The remaining problems are how to implement the metasample representations $\mathcal{M}_x(u_x, l)$ and how to select label candidates $L(u_x)$. The solutions are presented in the following two sections.

2.2.1 Representations of Metasamples

This section describes how the metasamples are made into numerical vectors that are ready to be used by metaclassifiers. Sparse vectors are adopted to represent metasamples. The conversion procedure consists of the following steps:

First, all the nodes except the root are numbered with integers. These integers are taken as the dimensions to encode the confidence scores of base classifiers.

Second, sparse vectors are augmented with additional features that are the attributes of the root-to-leaf paths. The intuition is that these attributes are helpful to decide whether a path is correct, i.e., leading to a correct class label. The following four additional features are employed according to our pilot experiments:

1. the *average score* of the nodes along a path;
2. the *minimum score* of the nodes along a path;
3. the *average ranking* of the scores along a path

$$s_{ar} = \frac{1}{l}\sum_{i=1}^{l} r_i,$$

where l is the length of the path and r_i is the ranking of node is score (the first is counted as 0, the second as 1, and etc.); and
4. the fraction of the nodes whose scores exceed ScutTD's thresholds, named *pass-rate*.

The effects of *average ranking* and *pass-rate* are analyzed in Section 2.4.1.

In the end, the values of metafeatures are mapped into specific intervals to enhance the training of metaclassifiers [26], [27], [37]. Standard scaling function and Platt sigmoid fitting [38], [39] have been tried in our pilot experiments, and eventually the following mixed scaling method is employed. For the additional features, the standard scaling function is used as follows:

$$z_k = \frac{s_k - \mu_k}{\sigma_k},$$

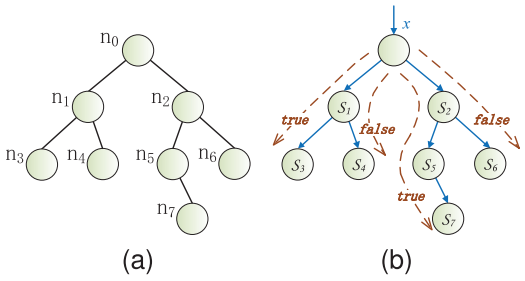


Fig. 4. Illustration of the proposed MetaTD: (a) class hierarchy and (b) paths as metasamples.

where s_k is the value of the kth feature in feature vectors and μ_k and σ_k are the corresponding mean and variance, respectively.

For the basic features, a simplified sigmoid function is used [39], [40],

$$z_n = \frac{1}{1 + e^{-(s_n - \mu_n)}},$$

where s_n is a score at a node n, and μ_n is the average score at node n.

2.2.2 Selection of Label Candidates

How should label candidates be selected? In fact, the method of selecting label candidates is a kind of a classification method as both of them take in samples and give out the labels that are most likely to be right. However, the method of selecting label candidates should output more labels than a normal classifying method, to provide a wider coverage of the truly correct ones. To find such a "loose" classifying method, we refer to RcutTD with a parameter $r \geq 2$ (see Fig. 2). RcutTD with $r = 1$ predicts one label per sample, thus it is normally used in single-labeled classification. RcutTD with $r \geq 2$ predicts multiple labels per sample which are taken as label candidates by MetaTD.

2.3 Illustration of Meta-Top-Down Method

This section illustrates MetaTD with a specific example. Suppose a hierarchical classification task has the hierarchy of classes shown by Fig. 4a. The nodes are numbered from n_0 to n_7 where n_0 is the root and the leaf nodes n_3, n_4, n_6, and n_7 are valid labels.

Suppose that a tree of base classifiers has been built through top-down training, and a sample takes n_3 and n_7 as its correct labels. Fig. 4b shows that each base classifier yields a relevant score s_1-s_7.

MetaTD considers classifying this sample as picking out the leaf nodes that are most likely to be correct. Each leaf node is converted to a metasample—the target is whether this leaf node is correct and the features are the scores of the base classifiers along the path (see Fig. 4b). For this example, the following four metasamples can be generated:

$$\begin{aligned}
&\text{true} \quad n_0 \rightarrow (n_1, s_1) \rightarrow (n_3, s_3),\\
&\text{false} \quad n_0 \rightarrow (n_1, s_1) \rightarrow (n_4, s_4),\\
&\text{true} \quad n_0 \rightarrow (n_2, s_2) \rightarrow (n_5, s_5) \rightarrow (n_7, s_7),\\
&\text{false} \quad n_0 \rightarrow (n_2, s_2) \rightarrow (n_6, s_6).
\end{aligned} \tag{4}$$

TABLE 1
Representing Metasamples with Sparse Vectors

Basic			Additional			
1:s_1 [a]	3:s_3		8:a_{13} [b]	9:mi_{13}	10:ar_{13}	11:pr_{13}
1:s_1	4:s_4		8:a_{14}	9:mi_{14}	10:ar_{14}	11:pr_{14}
2:s_2	5:s_5	7:s_7	8:a_{257}	9:mi_{257}	10:ar_{257}	11:pr_{257}
2:s_2	6:s_6		8:a_{26}	9:mi_{26}	10:ar_{26}	11:pr_{26}

[a] *dimension* : *value*, the dimension with zero value are not represented.

[b] $a_{i_1 i_2 \ldots i_k}$, $mi_{i_1 i_2 \ldots i_k}$, $ar_{i_1 i_2 \ldots i_k}$ and $pr_{i_1 i_2 \ldots i_k}$ denote the *average*, *minimum*, *average ranking* and *pass-rate* of s_{i_1}, s_{i_2} ,..., s_{i_k}, respectively.

These metasamples are then interpreted into sparse vectors. Suppose that n_1 to n_7 are numbered with integers $1, 2, \ldots, 7$, respectively, then sparse vectors can be generated as Table 1.

A metaclassifier is trained by a set of metasamples that are derived from multiple base-level samples as above. Afterward this classifier is applied to the metasamples derived from a base-level test sample to predict a score for each label candidate. In this way, MetaTD fulfills the original base-level classifying task.

2.4 Performance Analysis

In this section, we analyze the classification accuracy and computational complexity of the proposed MetaTD. During the accuracy analysis, we first formulate the accuracy of the normal top-down methods and the proposed MetaTD, and then qualitatively compare their predicted results to prove that MetaTD are more accurate.

2.4.1 Accuracy Analysis

The classifying procedure of the normal top-down methods is to recursively classify samples at a node into its children nodes. However, the practical accuracy of such a procedure is complicated and affected by many factors such as the number and quality of the training examples, the distribution of children nodes, and so on [8]. So we simplify the problem by assuming the average precision and recall of the flat atomic classification are $\mathcal{P}$ and $\mathcal{R}$, respectively. Since the top-down classification is equivalent to cascaded flat classifications, its accuracy can be expressed as follows:

$$\mathcal{P}_{TD} = \mathcal{P}^h,$$
$$\mathcal{R}_{TD} = \mathcal{R}^h,$$

where $\mathcal{P}_{TD}$ and $\mathcal{R}_{TD}$ are the precision and recall, respectively, and h is the height of the hierarchy.

The classifying procedure of MetaTD consists of two parts. The first part is to employ RcutTD ($r \geq 2$) to select label candidates. The ratio of recalling right labels is $\mathcal{R}_r^h$, where R_r is the recall of the atomic classification with the R-cut thresholding strategy. The second part is to employ metaclassification to decide the predicted labels. Assume that the precision and recall of the metaclassification are $\mathcal{P}_M$ and $\mathcal{R}_M$, respectively, then the classification accuracy of MetaTD is as follows:

$$\mathcal{P}_{MetaTD} = \mathcal{P}_M,$$
$$\mathcal{R}_{MetaTD} = \mathcal{R}_r^h \mathcal{R}_M.$$

In fact, $\mathcal{R}_r (r \geq 2)$ is fairly high in real-world hierarchical classification tasks as investigated in Section 3.6. Therefore, the performance of MetaTD is largely determined by that of the metaclassification.

To investigate the performance of the metaclassification, suppose there is a metasample represented by the following sparse vector:

$$(n_{i_1}{:}s_1,\ n_{i_2}{:}s_2,\ \ldots,\ n_{i_k}{:}s_k,\ n_{av}{:}s_{av},\ n_{mi}{:}s_{mi},\ n_{ar}{:}s_{ar}\ n_{pr}{:}s_{pr}),$$

where n_{k_i} and s_i are a node and its value; n_{av},n_{mi}, n_{ar}, and n_{pr} are the additional features of *average score*, *minimum score*, *average ranking*, and *pass-rate*, respectively.

ScutTD works the same way as the following metaclassifier [8], [35]:

$$output = \begin{cases} true, & \text{if } s_{pr} = 1 \text{ for all } i = 1\ \ldots\ k, \\ false, & \text{otherwise}, \end{cases}$$

where $s_{pr} = 1$ means that the scores along the root-to-leaf path sample all exceed the threshold values. Therefore, this metaclassifier is equivalent to ScutTD.

RcutTD ($r = 1$) works the same way as the following metaclassifier:

$$output = \begin{cases} true & \text{if } s_{ar} = 0, \\ false & \text{otherwise}, \end{cases}$$

where $s_{ar} = 0$ means that the scores along the path all rank at the first place. Therefore, this metaclassifier is equivalent to RcutTD ($r = 1$).

The metaclassification employs the machine learning methods to find the optimal metaclassifiers [22], [28]. Since the metaclassifiers equivalent to ScutTD and RcutTD ($r = 1$) are in its searching space, the metaclassification is able to achieve higher classification accuracy.

2.4.2 Computational Complexity Analysis

Before analyzing the complexity of MetaTD, let us first review the computational complexity of the normal top-down methods [8], [11]. The training process is mainly to train base classifiers whose complexity is as follows:

$$Q_{train}^{TD} = \sum_{i=1}^{h} \sum_{j=1}^{l_i} O\big(n_{ij}^{\alpha} m^{\beta}\big)$$

where h is the height of the hierarchy (the root node is at the 0-th level), l_i is the number of nodes in the ith level, n_{ij} is the number of training samples at the jth node of the ith level, m is the number of features, and α and β are constants determined by base classifiers.

Suppose the base classifiers are linear classifiers whose computational complexity is linear in the number of features. Then the classifying complexity of the normal top-down methods can be expressed as follows:

$$Q_{classify}^{TD} = \sum_{i=1}^{h} \sum_{j=1}^{l_i} O(\delta_{ij} m),$$

where δ_{ij} is a 0/1 function indicating whether the father node of the ith node at the jth level is invoked, which is equivalent to whether this node is to be checked.

The classifying complexity of RcutTD has an upper bound as follows:

$$Q_{classify}^{RcutTD} \leq \sum_{i=1}^{h} O(r^i m) \leq O(r^{h+1} m) \tag{5}$$

From (5), we can see that the complexity is decided by the hierarchy. The upper bound is reached when the hierarchy is a complete tree, that is, every node above the bottom level is a parent node with no less than r child nodes.

Now, let us turn to the complexity analysis of the proposed MetaTD. Its training process mainly consists of training base classifiers like the normal top-down methods, and an additional metatraining whose complexity is determined by the size of metatraining set. Thus, the training complexity of the proposed MetaTD can be expressed as follows:

$$\begin{aligned} Q_{train}^{MetaTD} &= Q_{train}^{TD} + O\big(N_s^{\alpha} N_f^{\beta}\big) \\ &\leq Q_{train}^{TD} + O\big(r^{h\alpha} (N_D)^{\alpha} (N_{node} + 4)^{\beta}\big), \end{aligned}$$

where r is the parameter of RcutTD, N_s is the number of metasamples, N_f is the number of metafeatures, N_D is the size of the development set, and N_{node} is the total number of nodes in the hierarchy. Note that each metasample actually has at most $h + 4$ nonzero features, so the metatraining must be quite fast if the proper training algorithms are employed (see Section 3.7).

The classifying part of MetaTD is composed of an RcutTD classification and a metaclassification. Suppose the metaclassifier is also linear. The complexity of MetaTD classifying can be expressed as follows:

$$\begin{aligned} Q_{classify}^{MetaTD} &= Q_{classify}^{RcutTD} + O(n_r n_f') \\ &\leq O\big(r^{h+1} m\big) + O(r^h (h + 4)) \\ &= r^h (rm + h + 4), \end{aligned} \tag{6}$$

where n_r is the number of label candidates per base sample selected by RcutTD, and n_f' is the number of nonzero features.

3 Experiments

In this section, after describing the data sets and experimental settings, we present the performance comparison between MetaTD and the baseline methods as well as the existing records. We then report several auxiliary experiments including more performance comparison and tuning the settings of MetaTD.

3.1 Experimental Data Sets

Five real-world data sets of large-scale hierarchical classification are used in our experiments. Three of them are online document classification tasks from the PASCAL large-scale hierarchical text classification challenges (LSHTC), and two of them are International Patent Classification (IPC) tasks from the World Intellectual

TABLE 2
Scales of Data Sets

	# categories	# stems	# train docs	# test docs	avg. cat. per doc	longest path
LSHTC1 DMOZ	12,294	381,581	128,710	34,880	1.000	5
LSHTC2 DMOZ	27,875	594,158	394,756	104,263	1.028	5
LSHTC3 Wikipedia	36,504	346,299	456,886	81,262	1.845	11
WIPO-alpha IPC	3,715[a]	52,437	46,234	28,926	1.832	4[a]
NTCIR IPC	12,294[b]	694,021	3,137,057	359,080	2.651	5[b]

[a] Classification tasks of WIPO-alpha are conducted on the fourth level of IPC hierarchy since WIPO-alpha has insufficient samples for training models for low-level nodes.
[b] Classification tasks of NTCIR are conducted on full IPC hierarchy.

Property Organization (WIPO)[5] and NII Test Collection for IR Systems Project (NTCIR).[6] Table 2 presents the scales of these data sets. We briefly describe them below.

3.1.1 LSHTC Data Sets

LSHTC challenges aim at promoting the study of classification methods for large hierarchies. The first challenge was launched in 2009 [31], [41], the second in 2011 [42], and the third in 2012. The descriptions of participant systems are published through technical reports at the challenge's homepage and published papers at workshops of ECIR 2010 and ECML/PKDD 2011. The challenge data sets are publicly available, but the labels of the test sets remain undisclosed. The online result evaluation service remains open to facilitate related research. This service is employed to evaluate the experimental results on LSHTC data sets in this paper.

The LSHTC1 DMOZ data set is built by crawling the web pages annotated by the ODP project. The crawled documents are preprocessed through removing stop words, stemming by the Libstemmer toolkit,[7] and replacing the tokens by numeric IDs. The LSHTC2 DMOZ data set is built similarly, but the scale is much larger. The number of the documents increases threefold, and the number of the categories increases more than twofold. The LSHTC3 Wikipedia (median-size) data set is based on the Wikipedia categorization.[8] This data set uses the data from the open resource of DBpedia [43].

3.1.2 IPC Data Sets

Accurate classification of patent documents is crucial both to patent issuing and patent retrieval. The IPC taxonomy is a hierarchical patent classification system created under the Strasbourg Agreement in 1971 and administered by WIPO at present. IPC divides all the technical fields hierarchically into 8 sections, about 120 classes, 630 subclasses, about 7,000 groups and 69,000 subgroups [4], [5], [6], [44], [45].

The WIPO-alpha data set consists of English patent applications submitted to WIPO between 1998 and 2002. The NTCIR IPC data set consists of 3,496,137 Japanese patent applications submitted to Japan Patent Office from 1993 to 2002. This data set does not have a split of training and test data sets, so we partition the set by a time point—applications submitted from 1993 to 2001 as training set and those submitted at 2002 as test set.

3.2 Experimental Settings

3.2.1 Base-Level Sample Representation

The most widely used text representation in text classification, the bag-of-words model with the term weight of term frequency—inverse document frequency (TF-IDF), is adopted as the base-level sample representation [1], [36]. The following TFIDF formula suggested by [46] is adopted, as our pilot experiments show that it yields slightly higher classification accuracies than other variants.

$$\mathrm{TFIDF}(t,d) = n(t,d)\log\frac{|T|}{n_T(t)},$$

where t denotes a term, d denotes a document, T denotes the training corpus, $n(t,d)$ denotes the number of times t that occurs in d, namely term frequency, and $n_T(t)$ denotes the number of documents where t occurs, named document frequency. The length of representation vectors is normalized to 1.

In LSHTC data sets, samples are presented in the form of numeric stem IDs and their term frequencies. Thus, they can be directly converted to the desired TFIDF bag-of-words representations.

In IPC data sets, samples are presented in plain text, thus they need to be tokenized, stemmed, and filtered to remove stop words. English texts in WIPO-alpha are tokenized by CoreNLP,[9] stemmed by Porter stemmer [47],[10] and filtered by Snowball stop words [48]. Japanese texts in NTCIR are tokenized and stemmed by Chasen [49],[11] and then function words are removed.

3.2.2 Performance Measurement and Baseline Method

The performance measurements and baseline methods are different between single-labeled data sets and multilabeled data sets. The LSHTC1 and DMOZ data sets are single-labeled, thus the accuracy is taken as the performance measurement. RcutTD with parameter $r = 1$ is taken as the baseline method [9].

5. http://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/wipo-alpha-readme.html.
6. http://research.nii.ac.jp/ntcir/index-en.html.
7. http://snowball.tartarus.org.
8. http://en.wikipedia.org/wiki/Wikipedia:FAQ/Categorization.
9. http://nlp.stanford.edu/software/lex-parser.shtml.
10. http://tartarus.org/~martin/PorterStemmer/.
11. http://chasen.naist.jp/hiki/ChaSen/.

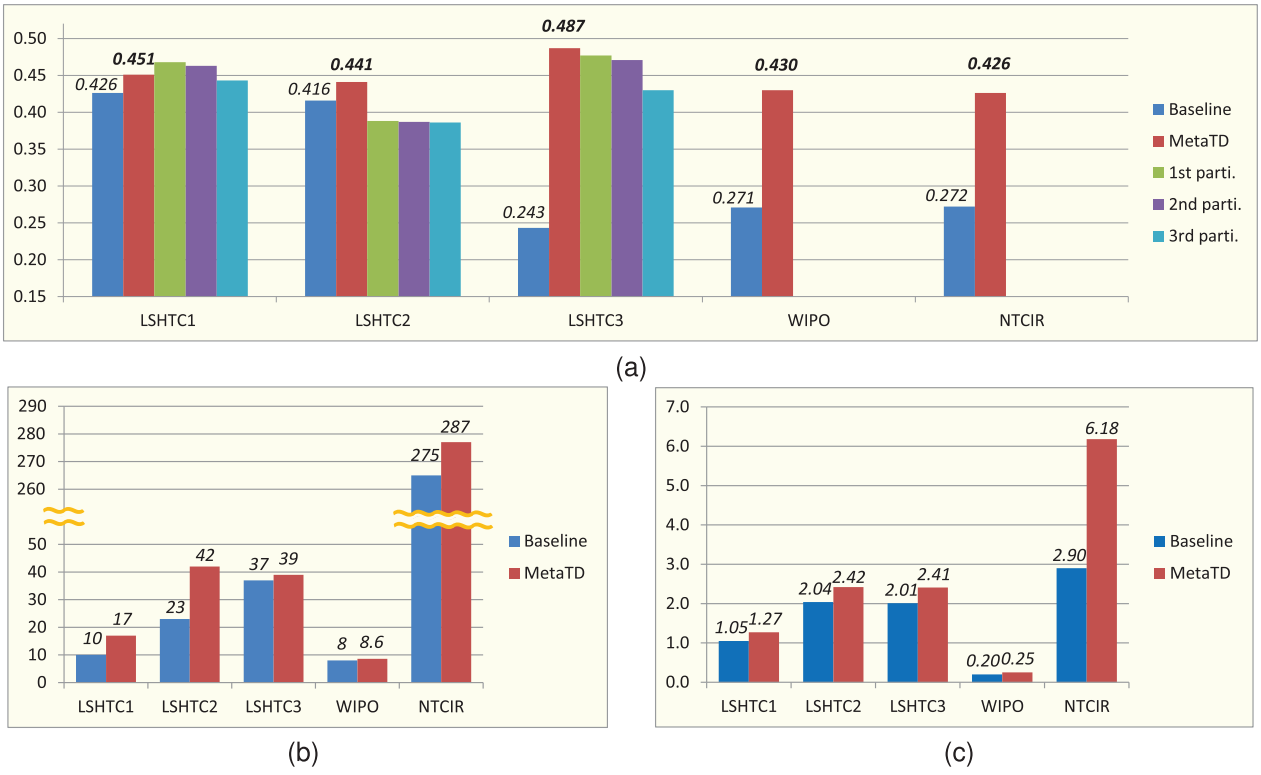


Fig. 5. Performance on whole data sets: (a) classification accuracies (accuracy for LSHTC1-2; micro-F_1 for LSHTC3, WIPO, and NTCIR), (b) training time cost (in hours), and (c) classifying time cost (in hours).

The LSHTC3, WIPO, and NTCIR data sets are multi-labeled, thus the micro-F_1 is taken as the performance measurement [1]. ScutTD is taken as the baseline method.

3.2.3 Settings of MetaTD

MetaTD selects label candidates through RcutTD as described in Section 2.2.2. The parameter r of RcutTD is set to 2 due to a tradeoff between the classification accuracy and time cost. The experiments testing different rs are present at Section 3.6.

The SVM implementation of Liblinear is adopted as the metaclassifier [50].[12] An investigation on various metaclassifiers can be found in Section 3.7.

Meta-to-base interpreters are needed to transfer metalevel predictions into base-level labels. For single-labeled tasks, it is natural to predict the labels with the largest metalevel scores. For multilabeled tasks, the strategy of S-cut in flat multiclass classification is employed. An optimal threshold is first decided on development sets, aiming to maximize micro-F_1. Then, when classifying an instance, any label with a metalevel score no less than that threshold is predicted.

3.2.4 Other Settings

The base-level classifier is SVM^{light} with a linear kernel. This classifier is widely used in text categorization because of its efficiency and high accuracy. The cost factors are tuned by development sets on LSHTC1 to make our experimental results competitive with the challenge records, while the default cost factors are adopted in the rest data sets.

Development sets are necessary for ScutTD and MetaTD. LSHTC1 provides a development set. On LSHTC2-3 and WIPO, we randomly pick out one-third of the training samples from each class as development samples. On NTCIR, we take the patent applications submitted at 2001 as development samples.

The experiments are run on four 64-bit computers with multicore 1.9-GHz AMD CPUs and 8G-64G memory. All the experiments actually require up to 8G memory according to our observations, except for the experiments with larger parameter rs (see Section 3.6). During the experiments, the independent computations such as training the base classifiers of the top-down methods are conducted in parallel, and the equivalent serial running times are calculated and taken as the time costs.

3.3 Performance on Entire Data Sets

3.3.1 Accuracy

Fig. 5a presents the classification accuracies of the normal top-down methods (as baseline), MetaTD, and LSHTC1-3 participants.[13] MetaTD achieves higher accuracies than the baseline methods in all five data sets. The improvements on the multilabeled data sets of LSHTC3, WIPO, and NTCIR are significant, where the micro-F_1s are raised by 36.2-57.3 percent. The improvements on single-labeled data sets of LSHTC1 and LSHTC2 are less significant but still obvious, where the accuracies are raised by 5.9 percent.

MetaTD are also competitive among LSHTC participants' methods. MetaTD ranks the first in both LSHTC2-3, and ranks the third in LSHTC1. In LSHTC1, one of the two

12. http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

13. Retrieved from http://lshtc.iit.demokritos.gr/ at June 29th, 2012.

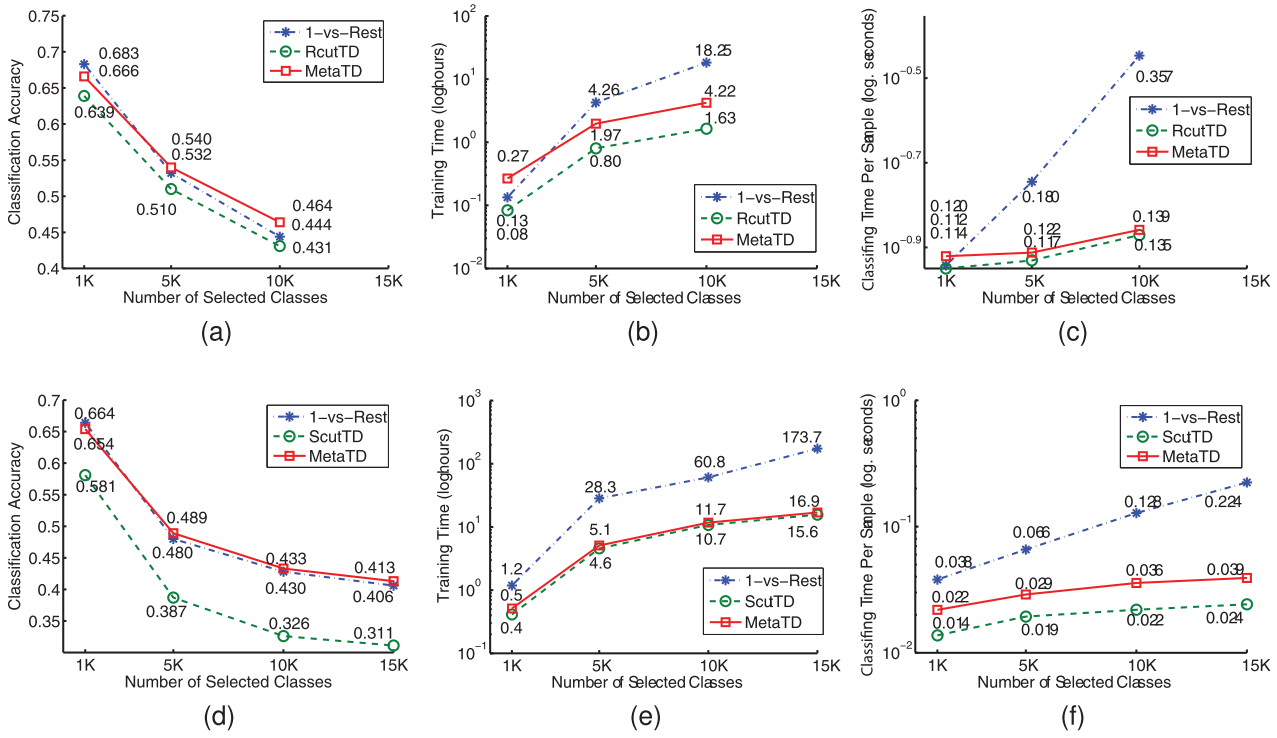


Fig. 6. Performance comparison of the one-versus-rest method, the normal top-down methods and MetaTD on subsets of various sizes: (a) through, (c) for LSHTC1; (d) through, (f) for NTCIR.

methods that beat MetaTD is a combination of two flat approaches—variants of the OOZ algorithm [51], [52] and the passive-aggressive algorithm [53]. The other method is not published by the participants.

Note that accuracy gaps among LSHTC1-3 participants' methods are rather small, which implies accuracy improvements are quite difficult. In LSHTC1, the baseline method RcutTD would ranks eighth place though its accuracies are only 0.025 lower than MetaTD. In LSHTC2, MetaTD outperforms the best method of the participants in accuracy by 0.053. In LHSTC3, MetaTD outperforms the best method in micro-F_1 by 0.010, while the best outperforms the second by 0.006. Published papers show that LSHTC2 participants employ methods such as simplified plat approaches [54], [55], k-nearest neighbors [56], [57], but they have not tried top-down methods.

3.3.2 Efficiency

Figs. 5b and 5c present the training and test time costs, respectively. Note that for top-down methods, training is usually much more time-consuming than test. Therefore, the training time costs are more concerned by us.

The training time costs are mainly spent on training base classifiers for both the normal top-down methods and MetaTD, while the cost spent by MetaTD on metaclassification is small by contrast. On the multilabeled data sets of LSHTC3, WIPO, and NTCIR, MetaTD's time costs are 4.3-7.5 percent larger than those of the baseline method ScutTD. It is because that both MetaTD and ScutTD need to train base classifiers twice, once on the monotraining sets and the other on the union sets of the development and training samples. On single-labeled data sets of LSHTC1-2, MetaTD's time costs are 70.0-82.6 percent larger than those of baseline method RcutTD. It is because RcutTD saves the time costs of training base classifiers on the monotraining sets as it has no parameters to tune.

3.4 Performance for Increasing Number of Classes

This section compares the performance of the one-versus-rest method, the normal top-down methods and MetaTD. Given the high computational complexity of the one-versus-rest method, several subsets are made from data sets LSHTC1 and NTCIR.

The subsets of LSHTC1 are made as follows: First, all the 10,159 classes that have at least three training samples are collected. Second, a desired number of classes are randomly picked out. Finally, the samples of these classes are divided into training, development, and test sets with the ratio of 3:1:1, while each set has at least one sample per class. The subsets of NTCIR are made similarly with the training, development, and test samples, respectively, picked from the data set of the years 1998, 1999, 2000, 2001, and 2002.

The experimental settings are consistent with the previous experiments. The experimental results are presented in Fig. 6. In classification accuracy, MetaTD is close to one-versus-rest. In particular, MetaTD slightly outperforms one-versus-rest on both data sets when the number of classes exceeds 5,000. As for the normal top-down methods, their accuracies are 6-30 percent lower than those of one-versus-rest as expected.

In computational complexity, MetaTD is close to ScutTD and RcutTD, and all these top-down methods show a great superiority over the one-versus-rest method on both the training and classifying time costs as expected.

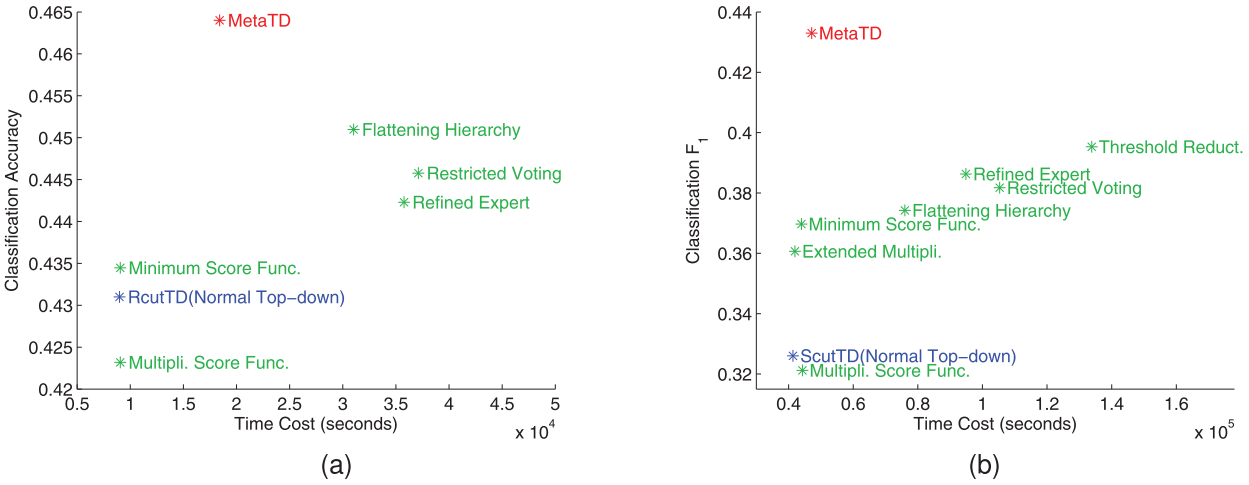


Fig. 7. Time cost versus classification accuracy (red: MetaTD, green: related works, and blue: normal top-down methods): (a) results on the subset of LSHTC1 (10k classes) and (b) results on the subset of NTCIR (10k classes).

TABLE 3
Experiments of MetaTD with Different Feature Combinations and rs

	Accuracy/Micro-F_1					Time cost				
	LSHTC1	LSHTC2	LSHTC3	WIPO	NTCIR	LSHTC1	LSHTC2	LSHTC3	WIPO	NTCIR
Selecting Label Candidates with Different r's[a]										
$r = 1$	0.451	0.416	0.351	0.327	0.360	**1.27**	**2.42**	**1.05**	**0.13**	**4.82**
$r = 2$	0.451	0.441	**0.487**	**0.343**	**0.426**	1.52	2.98	2.41	0.25	6.18
$r = 3$	**0.452**	**0.447**	0.461	0.292	0.405	2.12	4.67	4.19	0.32	7.58
Choice of Meta-classifiers[b]										
Liblinear(SVM,primal)	**0.451**	**0.441**	0.487	**0.430**	**0.426**	**18**	170	**350**	7	**466**
Stanford(ME, primal)	0.434	0.435	0.485	0.418	0.413	23	**137**	403	**6**	502
SVM^{light}(SVM, dual)	0.446	0.437	**0.494**	0.406	0.422	6,377	21,594	104,955	436	2.07×10^6

[a] The presented time costs (in hours) are for classifying test samples, and they are mainly determined by r's.
[b] The presented time costs (in seconds) are for training meta classifiers, and they are affected by practical implements.

3.5 Performance Comparison with Related Work

This section compares the performance of MetaTD with those of the related work. Seven enhanced top-down methods are tested, including minimum score function,[14] multiplicative score function [33], restricted voting, threshold reduction, extended multiplicative method [32], flattening hierarchy [30], [58], and refined expert [14] (see Section 4 for details). Due to the high computational complexity of some methods, the experiments are conducted in the subsets made in the last section. The threshold reduction and extended multiplicative method proposed by Sun et al. [32] are based on the thresholding strategy of ScutTD, which are proper for multilabeled classification. Therefore, they are tested only on the subset of NTCIR.

The results are presented in Fig. 7. On the aspect of effectiveness, MetaTD achieves the highest accuracy in both data sets. The second best methods are flattening hierarchy on the LSHTC1 subset and threshold reduction on the NTCIR subset, respectively. On the aspect of efficiency, the time consumed by MetaTD is close to that of the normal top-down methods, and much less than that of threshold reduction, refined expert, restricted voting, and flattening hierarchy.

14. The originally proposed function is the Boolean function, and this paper replaces it by the continuous counterpart—the minimum function—to achieve higher accuracy.

3.6 Select Label Candidates with Different rs

MetaTD employs RcutTD to select label candidates. Parameter r of RcutTD is the maximum number of children that can be invoked from a parent node.

Larger r will recall more correct labels for the metaclassification phrase, but the time cost will also be increased.

This paper sets the parameter r to 2 mainly based on the following heuristics. First, on the multilabeled data sets LSHTC3, WIPO, and NTCIR, the average numbers of correct children per invoked parent node are 1.340, 1.171, and 1.258, respectively. Second, two children per parent node can cover 92.0 percent, 98.0 percent and 89.6 percent of parent-child relations on these data sets, respectively.

To examine such setting, the experimental results with different rs are presented in Table 3. The time costs are increased by using larger rs, but the classification accuracies do not rise significantly. Therefore, the setting of $r = 2$ is proper for all five data sets.

3.7 Choice of Metaclassifiers

This section explores which classifier can provide high accuracy as well as short training time on the metaclassification tasks of MetaTD. Three classifier implements, including Liblinear [50], the Stanford classifier [59],[15] and SVM^{light} [60], are evaluated. Liblinear and SVM^{light} are both

15. http://nlp.stanford.edu/software/classifier.shtml.

TABLE 4
T-Test on LSHTC1 and LSHTC2

	LSHTC1 subsets	LSHTC2 subsets
Baseline	0.3988±0.0015	0.3858±0.0007
MetaTD	0.4165±0.0016	0.4024±0.0007
t_{paired}	33.9774	64.9683

support vector machine (SVM), and the Stanford classifier is a maximum entropy classifier. Besides, the solver of SVM^{light} works on the dual quadratic optimization problem of SVM training, while the solvers of the rest two implements work on the primal optimization problem.[16]

The experimental results are presented in Table 3. On the aspects of classification accuracy, three implements provide close accuracies, while the SVM classifiers of Liblinear and SVM^{light} are slightly more accurate than the Stanford maximum entropy classifier.

On the aspects of training time costs, SVM^{ight} spends far more time than the other two implements. It is because that the metatraining sets of MetaTD have far more samples than features. As the solver of SVM^{light} works on the dual problem of SVM training, whose complexity is greatly determined by the numbers of samples, it is slow in training. Note that, the normal text classification tasks usually have far more features than samples, which is the reason why SVM^{light} chooses to solve the dual problem. This experimental result confirms the information at LibLinear's homepage.[17]

3.8 T-Test Evaluation

The accuracy improvements over the baseline methods on LSHTC1 and LSHTC2 are small according to Fig. 5a. Therefore, the t-test evaluation is employed to find out whether such improvements are significant. The labels of the test samples of LSHTC1 and LSHTC2 are not revealed, thus we randomly pick up two-third labeled samples as training samples, and take the rest as test samples. The experiments are repeated for five times, the results of which are presented in Table 4. Paired-sample (or dependent sample) t-test reveals statistically significant differences between the performances of the baseline method and the proposed MetaTD, $t(4) = 4.604$ and $p = 0.005$.

4 Comparison with Related Work

In this section, the related work on the top-down methods and metaclassification is reviewed and compared with the proposed MetaTD. Several enhanced top-down methods are described and discussed here.

4.1 Top-Down Methods

Many researchers have proposed the variants of the top-down methods to overcome the deficiency of classification accuracy. Dumais and Chen investigate the method of classifying test samples by a universal scoring function of paths [33]. They test Boolean and multiplicative functions on a two-layered hierarchical web page classification task where the Boolean function turns out to perform a bit better. MetaTD is inspired by their work, while the universal scoring function is replaced with the metaclassification to improve the classification accuracy.

Sun et al. improve the top-down method by addressing the so-called blocking problem [32]. The blocking problem refers to the phenomenon that samples are wrongly rejected by the high-layered classifiers and cannot be passed down; it is part of the error-propagation problem. They propose three solutions. The first one, named the restricted voting method, is to modify the hierarchy through linking nodes with their grandchild nodes (see Fig. 8a). The modified hierarchy makes samples easier to be passed down, while the computational complexity is largely increased at the same time. The second solution, named the threshold reduction method, is to exhaustively search for the optimal combination of thresholds. The intuition is that the thresholds at higher layers should be smaller than those at lower layers to let more samples pass. To minimize the number of threshold combinations, the classifiers at the same layer are required to use the same threshold value. The third solution, named the extended multiplicative method, is to associate a local classifier with the parent's classifier. When the product of the two classifier's probabilistic scores is accepted by the thresholding strategy, the sample is passed down.

Wang and Lu, and Malik flatten the hierarchy to raise the classification accuracy of the top-down methods [30], [58]. This method is a compromise between the flat and hierarchical classifications. The flattened hierarchy is closer to a flat structure and has less layers, thus the problem of error propagation is relived (see Fig. 8b). However, the computation complexity is also increased and becomes closer to that of the flat classification. Wang and Lu investigate the strategy of flattening, and conclude that removing the nodes from the higher layers brings more accuracy improvement [30].

Bennett and Nguyen also propose a metaclassification enhanced top-down method named refined expert [14]. They first build a tree of classifiers through the standard top-down training, and then build another tree of metaclassifiers, which are trained by the combination vectors of samples' normal representations and their predictions from the lower nodes in the first tree (see Fig. 8c). The intuition is that the scores of the lower nodes should be taken into consideration while making decisions at high layers. Refined expert retrains the base classifiers through metaclassification, so its computation complexity times that of the normal top-down methods, much higher than that of the proposed MetaTD as shown by the experimental results.

4.2 Metaclassification

Metaclassification is a classification method that takes the outputs of the existing classifiers as inputs to better learn target signals. Metaclassification has been widely used to improve flat multiclass classification [21], [23], [24], [25], [26], [27], [28], [29]. Suppose there is a classification problem of n_c classes. First n_{bc} base classifiers are trained for each class by employing different classifiers or manipulating the input features. Then a metaclassifier is trained for each class by learning the predictions of all base classifiers, i.e., an $n_{bc} \times n_c$ matrix of predictions. In addition, fixed combination rules

16. Liblinear also provides solvers on the dual problems.
17. http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html.

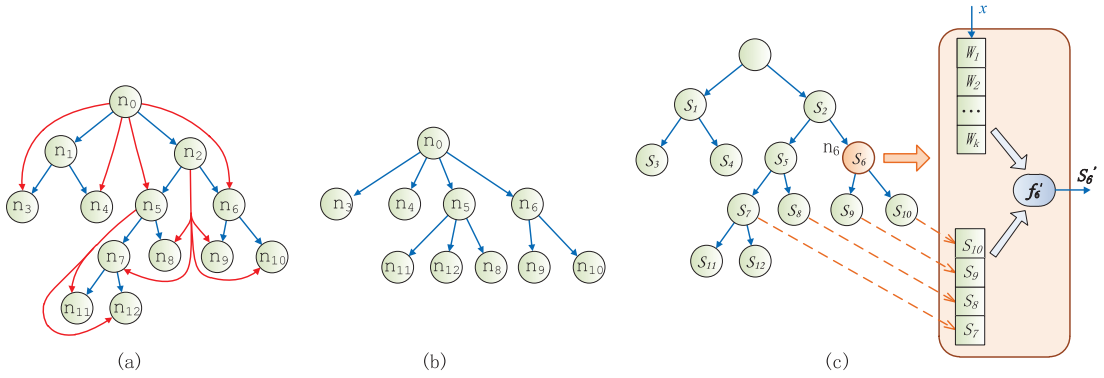


Fig. 8. Related work on the top-down methods: (a) restricted voting method (red paths are added), (b) flattening the hierarchy into two layers, and (c) refined expert (The metaclassifier f'_6 takes sample xs normal representation and scores of the child and nephew nodes as input.).

such as summation and mean have also been used to combine multiple classifiers instead of training metaclassifiers [16], [18], [19], [20], [22].

The proposed MetaTD apply metaclassification to top-down methods in a different way from the general usage. As far as we know, metaclassification has not been used in top-down methods before. The main differences are as follows:

- Base classifiers are attached to internal nodes instead of classes. Besides, the number of base classifiers is huge. As a solution, a sparse vector is adopted as the representation of metasamples.
- Numbers of classes are so large that training one metaclassifier per class is infeasible. Instead, a global metaclassifier for all classes is trained.
- To raise classification accuracy, besides the scores of base classifiers, the average and minimum values of these scores that are derived from fixed combination rules, are also taken as metafeatures.

5 Conclusions and Future Work

This paper proposes a meta-top-down method (MetaTD) to relieve the error-propagation problem of the normal top-down methods while retaining their capability for large-scale hierarchical classification. In the accuracy analysis, MetaTD is proved to subsume the normal top-down methods, ensuring that it can provide higher classification accuracy.

The experimental results show that, on the aspect of classification accuracy, MetaTD outperforms ScutTD on multilabeled data sets by 36.2-57.3 percent, and outperforms RcutTD on single-labeled data sets by 5.9 percent. The comparison with the results from LSHTC1-3 challenges indicates that MetaTD is among the state-of-the-art methods. On the aspect of computational complexity, MetaTD raises the training time costs of ScutTD and RcutTD by 4.3-7.5 percent and 70.0-82.6 percent, respectively. Such performance is competitive among the related work.

In the future, we will apply MetaTD to more large-scale hierarchical classification tasks, particularly the nonmandatory leaf classification tasks such as Yahoo! categories. We expect that developing a flexible method of selecting label candidates for MetaTD will be a promising solution.

Acknowledgments

Bao-Liang Lu is supported by the National Natural Science Foundation of China (grant no. 61272248 and grant no. 90820018), the National Basic Research Program of China (grant no. 2009CB320901 and grant no. 2013CB329401), and the European Union Seventh Frame-work Programme (grant no. 247619). Hai Zhao is supported by the National Natural Science Foundation of China (grant no. 60903119 and grant no. 61170114).

References

[1] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.

[2] Y. Yang, "A Study of Thresholding Strategies for Text Categorization," *Proc. 24th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '01)*, pp. 137-145, 2001.

[3] Z.H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall Press, 2012.

[4] C.J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka, "Automated Categorization in the International Patent Classification," *ACM SIGIR Forum*, vol. 37, no. 1, pp. 10-25, 2003.

[5] A. Fujii, M. Iwayama, and N. Kando, "Introduction to the Special Issue on Patent Processing," *Information Processing and Management*, vol. 43, no. 5, pp. 1149-1153, 2007.

[6] C. Ma, B.L. Lu, and M. Utiyama, "Incorporating Prior Knowledge into Task Decomposition for Large-Scale Patent Classification," *Proc. Sixth Int'l Symp. Neural Networks: Advances in Neural Networks (ISNN '09)*, pp. 784-793, 2009.

[7] Y. Labrou and T. Finin, "Yahoo! as an Ontology: Using Yahoo! Categories to Describe Documents," *Proc. Eighth Int'l Conf. Information and Knowledge Management*, pp. 180-187, 1999.

[8] T.Y. Liu, Y. Yang, H. Wan, H.J. Zeng, Z. Chen, and W. Ma, "Support Vector Machines Classification with a Very Large-Scale Taxonomy," *ACM SIGKDD Explorations*, vol. 7, no. 1, pp. 36-43, 2005.

[9] D. Koller and M. Sahami, "Hierarchically Classifying Documents Using Very Few Words," *Proc. Int'l Conf. Machine Learning (ICML '97)*, pp. 170-178, 1997.

[10] A. Sun and E.P. Lim, "Hierarchical Text Classification and Evaluation," *Proc. IEEE Int'l Conf. Data Mining (ICDM '01)*, pp. 521-528, 2001.

[11] Y. Yang, J. Zhang, and B. Kisiel, "A Scalability Analysis of Classifiers in Text Categorization," *Proc. 26th Ann. Int'l ACM SIGIR Conf. Research and Development in Informaion Retrieval (SIGIR '03)*, pp. 96-103, 2003.

[12] A. Montejo-Ráez and L. Ureña-López, "Selection Strategies for Multi-Label Text Categorization," *Proc. Advances in Natural Language Processing*, pp. 585-592, 2006.

[13] G.R. Xue, D. Xing, Q. Yang, and Y. Yu, "Deep Classification in Large-Scale Text Hierarchies," *Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '08)*, pp. 619-626, 2008.

[14] P.N. Bennett and N. Nguyen, "Refined Experts: Improving Classification in Large Taxonomies," *Proc. 32nd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '09)*, pp. 11-18, 2009.

[15] M. Ceci and D. Malerba, "Classifying Web Documents in a Hierarchy of Categories: A Comprehensive Study," *J. Intelligent Information Systems*, vol. 28, no. 1, pp. 37-78, 2007.

[16] L. Xu, A. Krzyzak, and C.Y. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418-435, May/June 1992.

[17] P. Brazdil, J. Gama, and B. Henery, "Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning," *Proc. European Conf. Machine Learning*, pp. 83-102, 1994.

[18] T.K. Ho, J.J. Hull, and S.N. Srihari, "Decision Combination in Multiple Classifier Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66-75, Jan. 1994.

[19] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, "On Combining Classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226-239, Mar. 1998.

[20] B. Lu and M. Ito, "Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1244-1256, Sept. 1999.

[21] K.M. Ting and I.H. Witten, "Issues in Stacked Generalization," *J. Artificial Intelligence Research*, vol. 10, pp. 271-289, 1999.

[22] R.P.W. Duin, "The Combining Classifier: To Train or Not to Train?" *Proc. 16th Int'l Conf. Pattern Recognition*, vol. 2, pp. 765-770, 2002.

[23] W.H. Lin, R. Jin, and A. Hauptmann, "Meta-Classification of Multimedia Classifiers," *Proc. Int'l Workshop Knowledge Discovery in Multimedia and Complex Data*, 2002.

[24] L. Todorovski and S. Džeroski, "Combining Classifiers with Meta Decision Trees," *Machine Learning*, vol. 50, no. 3, pp. 223-249, 2003.

[25] S. Džeroski and B. Ženko, "Is Combining Classifiers with Stacking Better than Selecting the Best One?" *Machine Learning*, vol. 54, no. 3, pp. 255-273, 2004.

[26] C.L. Liu, H. Hao, and H. Sako, "Confidence Transformation for Combining Classifiers," *Pattern Analysis and Applications*, vol. 7, no. 1, pp. 2-17, 2004.

[27] C.L. Liu, "Classifier Combination Based on Confidence Transformation," *Pattern Recognition*, vol. 38, no. 1, pp. 11-28, 2005.

[28] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, "Review of Classifier Combination Methods," *Machine Learning in Document Analysis and Recognition*, pp. 361-386, Springer, 2008.

[29] Q. Kong, H. Zhao, and B.L. Lu, "Adaptive Ensemble Learning Strategy Using an Assistant Classifier for Large-Scale Imbalanced Patent Categorization," *Proc. 17th Int'l Conf. Neural Information Processing: Theory and Algorithms*, pp. 601-608, 2010.

[30] X.L. Wang and B.L. Lu, "Flatten Hierarchies for Large-Scale Hierarchical Text Categorization," *Proc. Fifth Int'l Conf. Digital Information Management*, pp. 139-144, 2010.

[31] A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham, "The ECIR 2010 Large Scale Hierarchical Classification Workshop," *ACM SIGIR Forum*, vol. 44, no. 1, pp. 23-32, 2010.

[32] A. Sun, E.P. Lim, W.K. Ng, and J. Srivastava, "Blocking Reduction Strategies in Hierarchical Text Classification," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1305-1308, Oct. 2004.

[33] S. Dumais and H. Chen, "Hierarchical Classification of Web Content," *Proc. 23rd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '00)*, pp. 256-263, 2000.

[34] A.A. Freitas and A.C. de Carvalho, "A Tutorial on Hierarchical Classification with Applications in Bioinformatics," *Research and Trends in Data Mining Technologies and Applications*, D. Taniar, ed., pp. 175-208, Idea Group, 2007.

[35] C.N. Silla and A.A. Freitas, "A Survey of Hierarchical Classification across Different Application Domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 1-42, 2010.

[36] D.D. Lewis, Y. Yang, T.G. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *J. Machine Learning Research*, vol. 5, pp. 361-397, 2004.

[37] C.C. Chang and C.J. Lin, "LIBSVM: A Library for Support Vector Machines," http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.

[38] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Proc. Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61-74, 1999.

[39] H.T. Lin, C.J. Lin, and R.C. Weng, "A Note on Platt's Probabilistic Outputs for Support Vector Machines," *Machine Learning*, vol. 68, no. 3, pp. 267-276, 2007.

[40] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical Classification: Combining Bayes with SVM," *Proc. Int'l Conf. Machine Learning (ICML '06)*, pp. 177-184, 2006.

[41] A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham, "The ECIR 2010 Large Scale Hierarchical Classification Workshop," *ACM SIGIR Forum*, vol. 44, no. 1, pp. 23-32, 2010.

[42] G. Paliouras, E. Gaussier, A. Kosmopoulos, I. Androutsopoulos, T. Artieres, and P. Gallinari, *Proc. Joint ECML/PKDD PASCAL Workshop Large-Scale Hierarchical Classification*, 2011.

[43] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," *Proc. Sixth Int'l The Semantic Web and Second Asian Conf. Asian Semantic Web Conf.*, pp. 722-735, 2007.

[44] K. Wu, B.L. Lu, M. Utiyama, and H. Isahara, "An Empirical Comparison of Min-Max-Modular K-NN with Different Voting Methods to Large-Scale Text Categorization," *Soft Computing*, vol. 12, no. 7, pp. 647-655, 2008.

[45] X. Chu, C. Ma, J. Li, B. Lu, M. Utiyama, and H. Isahara, "Large-Scale Patent Classification with Min-Max Modular Support Vector Machines," *IEEE Int'l Joint Conf. Neural Networks (IJCNN '08)*, pp. 3973-3980, 2008.

[46] S. Fujita, "Technology Survey and Invalidity Search: A Comparative Study of Different Tasks for Japanese Patent Document Retrieval," *Information Processing Management*, vol. 43, no. 5, pp. 1154-1172, 2007.

[47] P. Willett, "The Porter Stemming Algorithm: Then and Now," *Program: Electronic Library and Information Systems*, vol. 40, no. 3, pp. 219-223, 2006.

[48] M. Porter, "Snowball: A Language for Stemming Algorithms," 2001.

[49] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, and M. Asahara, "Morphological Analysis System Chasen Version 2.2. 1 Manual," *Nara Inst. of Science and Technology*, 2000.

[50] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin, "LIBLINEAR: A Library for Large Linear Classification," *J. Machine Learning Research*, vol. 9, pp. 1871-1874, 2008.

[51] O. Madani and J. Huang, "On Updates That Constrain the Features' Connections during Learning," *Proc. ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD '08)*, pp. 515-523, 2008.

[52] O. Madani and J. Huang, "Large-Scale Many-Class Prediction via Flat Techniques," *Proc. ECIR Large-Scale Hierarchical Classification Workshop*, 2010.

[53] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online Passive-Aggressive Algorithms," *J. Machine Learning Research*, vol. 7, pp. 551-585, 2006.

[54] C. Brouard, "ECHO at the LSHTC Pascal Challenge 2," *Proc. Joint ECML/PKDD PASCAL Workshop Large-Scale Hierarchical Classification*, pp. 49-57, 2011.

[55] V.N. Rao and R. Uppuluri, "A Two Stage Classification Approach for Large Scale Heirarchical Classification," http://lshtc.iit.demokritos.gr/lshtc2_shortpapers, 2011.

[56] X.L. Wang, H. Zhao, and B. Lu, "Enhance K-Nearest Neighbour Algorithm for Large-Scale Multi-Labeled Hierarchical Classification," *Proc. Joint ECML/PKDD PASCAL Workshop Large-Scale Hierarchical Classification*, 2011.

[57] X. Han, J. Liu, Z. Shen, and C. Miao, "An Optimized K-Nearest Neighbor Algorithm for Large Scale Hierarchical Text Classification," *Proc. Joint ECML/PKDD PASCAL Workshop Large-Scale Hierarchical Classification*, pp. 2-12, 2011.

[58] H. Malik, "Improving Hierarchical SVMS by Hierarchy Flattening and Lazy Classification," *Proc. ECIR Large-Scale Hierarchical Classification Workshop*, 2010.

[59] C. Manning and D. Klein, "Optimization, Maxent Models, and Conditional Estimation Without Magic," *Proc. Conf. North Am. Chapter of the Assoc. for Computational Linguistics on Human Language Technology: Tutorials*, pp. 8-8, 2003.

[60] T. Joachims, *Making Large-Scale Support Vector Machine Learning Practical*, pp. 169-184, MIT Press, 1999.

Xiao-Lin Wang received the BS degree in computer science from Shanghai Jiao Tong University, China, in 2004. Since then, he has been working toward the PhD degree at the Center of Brain-like Computing and Machine Intelligence in the same university. He is the key member of the lab participating at recent evaluations PASCAL LSHTC1-3 challenge and NTCIR8-9 Patent Mining task. His research interests include machine learning, natural language processing, and information retrieval.

Hai Zhao received the BEng degree in sensor and instrument and the MPhil degree in control theory and engineering from Yanshan University, and the PhD degree in computer science from Shanghai Jiao Tong University, China. He is currently an associate professor at Shanghai Jiao Tong University. He was a postdoctoral research fellow at the City University of Hong Kong from 2006 to 2009. His research interests include machine learning, natural language processing, and data mining and artificial intelligence.

Bao-Liang Lu received the BS degree in instrument and control engineering from the Qingdao University of Science and Technology, Shandong, China, in 1982, the MS degree in computer science and technology from the Northwestern Polytechnical University, Xi'an, China, in 1989, and the Dr Eng degree in electrical engineering from Kyoto University, Japan, in 1994. From 1982 to 1986, he was with the Qingdao University of Science and Technology. From April 1994 to March 1999, he was a frontier researcher at the Bio-Mimetic Control Research Center, the Institute of Physical and Chemical Research (RIKEN), Japan. From April 1999 to August 2002, he was a research scientist at the RIKEN Brain Science Institute. Since August 2002, he has been a full professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He has been an adjunct professor of the Laboratory for Computational Biology, Shanghai Center for Systems Biomedicine since 2005. His research interests include brain-like computing, neural network, machine learning, pattern recognition, computer vision, and neural engineering. He was a past president of the Asia Pacific Neural Network Assembly (APNNA) and the general chair of the 18th International Conference on Neural Information Processing (ICONIP2011). He is an associate editor of *Neural Networks*. He is a senior member of the IEEE.