Enhance Top-down method with Meta-Classification for Very Large-scale Hierarchical Classification [*]

Xiao-Lin Wang[1,2], **Hai Zhao**[1,2], **Bao-Liang Lu**[1,2†]
[1]Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University
800 Dong Chuan Rd., Shanghai 200240, China
arthur.xl.wang@gmail.com, {zhaohai; blu}@cs.sjtu.edu.cn

Abstract

Recent large-scale hierarchical classification tasks typically have tens of thousands of classes as well as a large number of samples, for which the dominant solution is the top-down method due to computational complexity. However, the top-down method suffers from accuracy deficiency, that is, its accuracy is generally lower than that of the flat approach of 1-vs-Rest. In this paper, we employ meta-classification technique to enhance the classifying procedure of the top-down method. We analyze the proposed method on the aspect of accuracy, and then test it with two real-world large-scale data sets. Our method both maintains the efficiency of the conventional top-down method and provides competitive classification accuracies.

1 Introduction

Test categorization, as a key technology of data mining, has received intensive study for decades. Recently, real-world applications have raised some large-scale tasks that typically have tens of thousands of classes, where many established techniques such as the 1-vs-Rest multiclass classification fail due to computational complexity. Meanwhile, those large-scale tasks usually employ hierarchies to organize the huge number of classes that they have, which provides a clue to solve them. Such kind of tasks include categorizing patent documents into the taxonomy of International Patent Classification (IPC) (Fall et al., 2003; Fujii et al., 2007) and categorizing web pages into the directories of Open Directory Project (ODP) or Yahoo! (Labrou and Finin, 1999; Liu et al., 2005).

The existing approaches to hierarchical classification mainly fall into two categories. One category aims at raising classification accuracy, which generally takes hierarchies as additional clue for classifying a sample besides its content. Such researches include hierarchical support vector machines (SVM) (Cai and Hofmann, 2004; Tsochantaridis et al., 2005), hierarchical Rocchio-like classifiers (Labrou and Finin, 1999), min-max modular network (Lu and Ito, 2002; Lu and Wang, 2009) and ensemble classifications (Punera and Ghosh, 2008).

The other category aims at reducing computational complexity. The main approach in this category is an ensemble classification method called top-down method (Bennett and Nguyen, 2009; Ceci and Malerba, 2007a; Koller and Sahami, 1997; Liu et al., 2005; Montejo-Ráez and Ureña-López, 2006; Sun and Lim, 2001; Xue et al., 2008; Yang et al., 2003). Top-down method builds a tree of classifiers which is isomorphic with the hierarchy of classes.

Top-down method classifies a test sample as follows. The sample is filtered down the tree of classifiers from the root node. For each parent node that the sample reaches, those child nodes whose confidence values predicted by the base-classifiers exceed a predefined threshold are invoked to carry the sample on. When the sample reaches the bottom leaf nodes eventually, the predictions can be made (Liu et al., 2005; Montejo-Ráez and Ureña-López, 2006; Yang et al., 2003). As this classifying process employs the threshold strategy of comparing the scores with thresholds, which is named score-cut (S-cut) in the context of flat multiclass classification (Yang, 2001), we call this kind of

* This work was partially supported by the National Natural Science Foundation of China (Grant No. 60903119, Grant No. 61170114 and Grant No. 90820018), the National Basic Research Program of China (Grant No. 2009CB320901), the Science and Technology Commission of Shanghai Municipality (Grant No. 09511502400), and the European Union Seventh Framework Programme (Grant No. 247619).

† Corresponding author

conventional top-down method the S-CUT Top-Down method (ScutTD) so as to distinguish it from the later variant top-down methods in this paper.

ScutTD is far more efficient than the normal flat approach of 1-vs-Rest in handling the classification tasks that has a large number of classes. The computational complexity of 1-vs-Rest is linear to the number of classes, while that of ScutTD is approximately logarithmic (Ceci and Malerba, 2007a; Liu et al., 2005; Wang and Lu, 2010; Yang et al., 2003). As an practical example, in an classification experiment on 492 617 training documents, 275 364 test documents and 132 199 categories of Yahoo!, ScutTD costs only 2.1 hours on training and 0.12 hours on classifying, while 1-vs-Rest costs 310 hours on training and 54 hours on classifying (Liu et al., 2005).

However, ScutTD has a well-known deficiency of classification accuracy, that is, its performance is generally worse than the flat 1-vs-Rest approach (Bennett and Nguyen, 2009; Ceci and Malerba, 2007a; Wang and Lu, 2010; Xue et al., 2008). As a persuasive evidence, in the 2009 PASCAL challenge on large-scale hierarchical text [1], flat methods rank highest, hybrid methods rank next and top-down methods rank lowest.

The main reason for the accuracy deficiency of ScutTD is that its classifying procedure actually consists of cascaded decisions about which child nodes should be invoked from a parent node. Each of these decisions is made upon the score of local base-classifiers only, and not changeable after that. Thus a wrong decision inevitably leads to a group of wrong predictions. This problem is usually called error propagation (Wang and Lu, 2010; Xue et al., 2008). Sun et al. study a special case of this problem, the wrong decision of rejecting a child node at high layers, and call it the blocking problem (Sun et al., 2004). Liu et al. compare this classifying procedure to a Pachinko-machine (Liu et al., 2005). As a solution, Ceci and Malerba has proposed a bottom-up thresholding strategy (Ceci and Malerba, 2007a)

In this paper, we propose a 'global' classifying method for top-down method to reduce its error propagation. The idea is to treat combining the predictions of the base-classifiers as a meta-classification task, for which we name our method Meta-classification Top-down method (MetaTD).

[1] `http://lshtc.iit.demokritos.gr/`

There is one point that needs to be clarified. 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 the validate labels or classes (Dumais and Chen, 2000; Freitas and de Carvalho, 2007; Silla and Freitas, 2010). In contrast, the other is non-mandatory leaf-node classification correspondingly, where both the internal nodes and the leaf nodes are validate labels (Lewis et al., 2004; Liu et al., 2005). In this paper, we handle the first kind of hierarchical classification – mandatory leaf-node classification.

The rest of paper is organized as follows. The proposed MetaTD as well as the conventional ScutTD is formally presented in Sec. 2. We then provide some ideas on the classification accuracy of MetaTD in Sec. 3. After that we test MetaTD with two real-world data sets in Sec. 4. Finally we conclude this paper in Sec. 5.

2 Methods

In this section, we present the formal descriptions of the proposed MetaTD. We first review the conventional ScutTD. We then present MetaTD in detail. After that an example is given to illustrate MetaTD.

2.1 S-cut Top-down Method

Suppose H is a hierarchy of classes which records all the relations of parent nodes and their children,

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

where (p, c) is called a parent-child relation. Suppose T,D and E are the training, development and test sets respectively.

Applying ScutTD consists of the following three steps.

First, train base-classifiers. One classifier will be trained for each parent-child relation (p, c) of the hierarchy H, noted as f_c, through the following local training set,

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

where T_* is the subset of training samples that belong to the node $*$.

Second, find optimal thresholds for the base-classifiers. The approaches to this step actually have alternatives. Micro-F_1 is taken as the criterion optimization target which balances both precision and recall, as follows (Bennett and Nguyen, 2009; Liu et al., 2005).

$$t_c = \underset{t}{argmax}\, F_1(D_{pc}, f_c, t) = \underset{t}{argmax}\, \frac{2P(D_{pc}, f_c, t)R(D_{pc}, f_c, t)}{P(D_{pc}, f_c, t) + R(D_{pc}, f_c, t)}, \quad (2)$$

$$P(D_{pc}, f_c, t) = \frac{n_r}{|\{x|(x, y) \in D_{pc}, f_c(x) \geq t\}|},$$
$$R(D_{pc}, f_c, t) = \frac{n_r}{|D_{pc}|},$$
$$n_r = |\{x|(x, y) \in D_{pc}, f_c(x) \geq t, y = 1\}|,$$

where t_c and f_c are the local threshold and base-classifier, D_{pc} is the local development subset which is similar with the T_{pc} defined by Eq. 1), P and R are the precision and recall, and n_r is the number of correct predict labels.

Third, classify the test instances. The algorithm of this step is presented in Fig. 1. With the trained base-classifiers f_c and the thresholds t_c, the test set E can be classified.

```
Require: a test instance $x$
  a hierarchy $H$={$(p, c)|(p, c)$ is a parent-child }
  base-classifiers $\{f_c|(p, c) \in H\}$
  thresholds $\{t_c|(p, c) \in H\}$
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
        $s_c \leftarrow f_c(x)$
        if $s_c \geq t_c$ then
          append $c$ into $p$
        end if
      end for
    end if
  end while
  return $y$
```

Figure 1: ScutTD algorithm

2.2 Meta-classification Top-down method

To describe the proposed MetaTD, we first introduce the definition of meta-samples as follows,

$$\mathcal{M}(u, l, f_*) = (\mathcal{M}_x(u_x, l, f_*), \mathcal{M}_y(u_y, l, f_*)) \quad (3)$$

$$\mathcal{M}_x(u_x, l, f_*) = \{(n_i, f_{n_i}(u_x))|n_i \in p_l\}$$
$$\mathcal{M}_y(u_y, l, f_*) = \begin{cases} +1, & l \in u_y \\ -1, & l \notin u_y \end{cases}$$

where $\mathcal{M}$ is the meta-mapping that consists of meta-input $\mathcal{M}_x$ and meta-output $\mathcal{M}_y$, H is a hierarchy, $u = (u_x, u_y)$ is a base-sample where u_x is the input part and u_y is the label set, l is a leaf node (or a label), that is, a validate label for base-samples, $p_l = (n_0, n_1, \ldots, n_k)$ is a path from the root to l where $n_0 = root$, $n_k = l$, $(n_i, n_{i+1}) \in H$, and f_* are base-classifiers.

However, the above definition yields one meta-sample for each class, which may cause a problem of computational complexity on large-scale tasks. Hence a method of selecting label candidates for each base-sample is employed so that only a small fraction of labels need to be delivered into meta-classification. We note this selection method as $L(u_x, f_*, H)$.

MetaTD is based on the above two settings, and its workflow is described in Fig. 2.

The training phase consists of three steps as follows,

1. Train base-classifiers f_* on a training data set T, which is the same with ScutTD.
2. Construct a meta-training set with the base-classifiers and a development set D,
   $M_T = \cup_{u \in D}\{\mathcal{M}(u, l, f_*, H)|l \in L(u_x, f_*, H)\}$.
3. Train a meta-classifier g on M_T.

The whole training phase requires the base-level training set T and development set D, the description of the hierarchy H, and produces a set of base-classifiers f_* and a meta-classifier g.

The classifying phase also consists of three steps as follows,

1. Construct a group of meta-samples from a test base-sample u_x (its label u_y is unknown),
   $M_E = \{\mathcal{M}_x(u_x, l, f_*)|l \in L(u_x, f_*, H)\}$.
2. Present these meta-samples to the meta-classifier g,
   $$g(M_E) = \{g(\mathcal{M}_x(u_x, l, f_*))|l \in L(u_x, f_*, H)\} = \{g_{u_x,l}|l \in L(u_x, f_*, H)\}.$$
3. Interpret the predictions into base-level labels. The interpretation is generally simple and

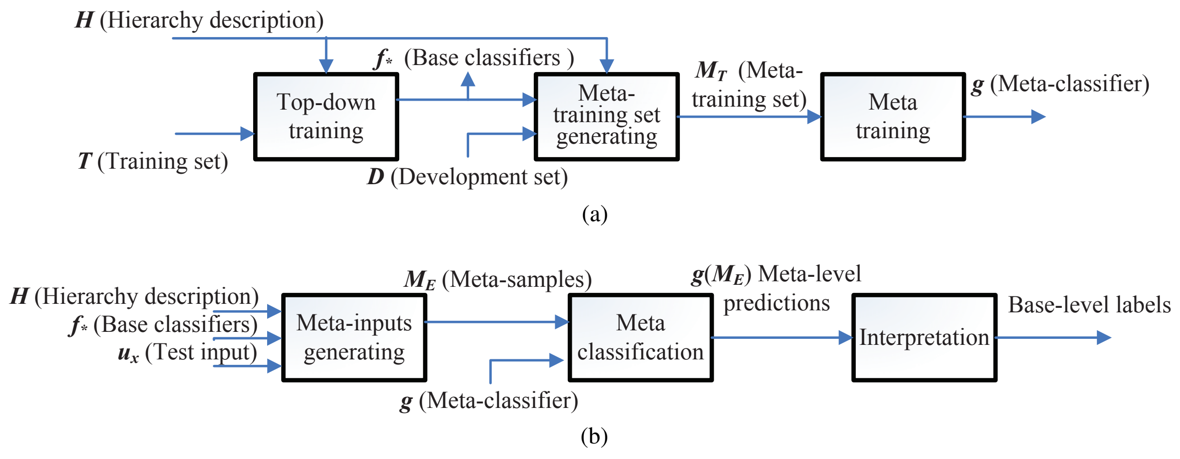

Figure 2: Workflows of meta-classification top-down method: (a) training phase; (b) classifying phase.

straightforward, and just outputs the labels with large scores. The practical interpretation depends on the data sets, and will be described in the section of experiments.

The remained problems now are how to implement meta-sample representations $\mathcal{M}_x(u_x, l, f_*)$ and selection of label candidates $L(u_x, f_*, H)$, which are solved in the next two subsections.

2.2.1 Representations of Meta-samples

In this subsection, the meta-samples will be made into real numerical vectors that are ready to be used by meta-classifiers. We use sparse vector to represent meta-samples through the following steps:

First, encode the scores of the related base-classifiers into a sparse vector. All the nodes except the root are numbered with integers, which serve as the dimensions of the sparse vector.

Second, augment the representations with the features about the global attributes of the root-to-leaf paths in the hierarchy. The purpose of this step is to raise classification accuracy, as these global attributes may be helpful to decide whether a path is true. The following three additional features are used according to our pilot experiments,

1. the *average* score of nodes along a path;
2. the *minimum* score of nodes along a path;
3. the fraction of nodes whose scores exceed the thresholds employed in ScutTD, named *pass-rate*.

In the end, the values of meta features are transferred into a sensible interval in order to fit the training of meta-classifiers (Liu, 2005; Liu et al., 2004). Two types of transformation functions are used according to our pilot experiments. For the additional features, the following standard scaling function is used,

$$z_s = \frac{s - \mu_s}{\sigma_s}$$

where s is the value of an additional feature, μ_s and σ_s are the corresponding mean and variance.

For the basic features, the following sigmoid function is used,

$$z_s = \frac{1}{1 + e^{-(s-\mu_s)}}$$

where s is a score at a node n, and μ_s is the average score at node n. This function is a simplification of the Platt' sigmoid fitting (Platt, 1999; Cesa-Bianchi et al., 2006), and it is more robust than the original one in the context of hierarchical classification according to our pilot experiements.

2.2.2 Selection of Label Candidates

How should label candidates be selected? In fact, the method of selecting label candidates is kind of like a classification method as both of them take in samples and give out the labels most likely to be right. However, the method of selecting label candidates should output more labels than a normal classifying method, in order to provide a wider coverage on truly correct ones. To find such a 'loose' classifying method, we refer to flat multi-class classification where another threshold strategy of Rank-cut (R-cut), besides the S-cut introduced above, is also widely used (Montejo-Ráez and Ureña-López, 2006; Yang, 2001). R-cut is to accept the top r labels with the highest confident scores, where r is a predefined integer.

Applying R-cut to the context of the top-down method is straightforward. The top-r children are

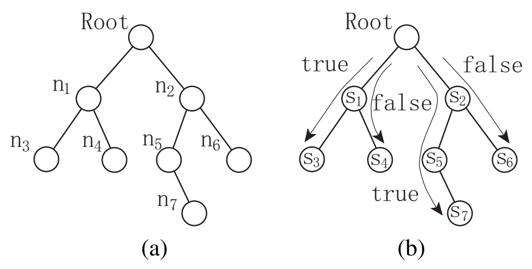

Figure 3: An illustration of solving hierarchical classification with MetaTD: (a) the class hierarchy; (b) the paths as meta-samples.

No.	Basic			Extension		
1	1:s_1[a]	3:s_3		8:a_{13}[b]	9:m_{13}	10:p_{13}
2	1:s_1	4:s_4		8:a_{14}	9:m_{14}	10:p_{14}
3	2:s_2	5:s_5	7:s_7	8:a_{257}	9:m_{257}	10:p_{257}
4	2:s_2	6:s_6		8:a_{26}	9:m_{26}	10:p_{26}

[a] dimension:value

[b] $a_{i_1 i_2 \ldots i_k}$, $m_{i_1 i_2 \ldots i_k}$, $p_{i_1 i_2 \ldots i_k}$ denote the average, minimum, and pass-rate of s_{i_1}, s_{i_2} $\ldots s_{i_k}$ respectively.

Table 1: Representing meta-samples with sparse vectors

invoked from their parent node regardless of their scores, and the rest procedure is the same with S-cutTD (see Fig. 1). We name this method RcutTD, and employ it to select label candidates in the proposed MetaTD. Note that RcutTD has been discussed before and is considered improper for the classifying procedure of the top-down method (Liu et al., 2005).

2.3 Illustration of Meta-classification Top-down Method

In this subsection we illustrate MetaTD with an example. Suppose a hierarchical classification task has the hierarchy of classes shown by Fig. 3a, where n_0 is the root, and the leaf nodes n_3, n_4, n_6 and n_7 are validate labels.

Further suppose that a tree of base-classifiers have been built through the top-down training. Here comes a sample with n_3 and n_7 as its correct labels. Fig. 3b shows that each base-classifier yields a relevant score s_i.

MetaTD converts each possible label (or leaf node) into a meta-sample – the target is whether this leaf node is a correct label and the features are the scores of the base-classifiers along the path (Fig. 3b). For this example, the following four meta-samples can be generated,

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

These meta-samples are then interpreted into numerical sparse vectors. Suppose that n_1 to n_7 are numbered with integers 1–7, then the numerical sparse vectors can be generated (see Tab. 1).

With more meta-samples like above, a meta-classifier can be trained. Later this meta-classifier can be applied to the meta-samples made from a base-level test sample to pick out the right labels. In this way, MetaTD fulfills the original base-level classifying task.

3 Accuracy Analysis

The classification accuracy of top-down methods is actually not very clear or predictable. To our best knowledge, no strict accuracy analyses on the conventional ScutTD have been reported yet. Here we just provide some general ideas about the comparison of accuracy between MetaTD and ScutTD.

First, whether pruning possible labels with R-cutTD or not has minor impact on the overall classification result. The labels rejected by RcutTD all have quite low scores on some parent-child relations and are very likely to be filtered out by the successive meta-classifier.

Second, ignoring the impact of selecting label candidates, the conventional top-down method of ScutTD can be actually seen as a weak meta-classifier in the framework of MetaTD. Suppose here is a meta-sample (a sparse vector),

$$(n_1{:}z_1,\ n_2{:}z_2,\ \ldots,\ n_k{:}p_k,\ n_a{:}z_a,\ n_m{:}z_m,\ n_p{:}z_p)$$

where n_i and p_i are a node number and its value, and n_a,n_m,n_p are the additional features. Then ScutTD works like,

$$\text{Output} = \begin{cases} \text{True} & \text{if } p_i > t_i \text{ for all } i = 1 \ldots k \\ \text{False} & \text{otherwise} \end{cases}$$

where t_i is the threshold of node n_i. Clearly this formula is a cascaded of binary decisions, which is weaker than some common classifiers such as weighted voting.

4 Experiments

In this section, after describing the experimental settings, we present the performance comparisons between MetaTD and baseline methods as well as historical records on the entire data sets. We then

Data	No. Sample			Feature		Class	
	Train.	Dev.	Test	No.	Avg.[a]	No.	Avg.[b]
LSHTC	93k	34k	34k	381k	173	12k	1.0
NTCIR	2 762k	374k	359k	694k	108	49k	2.7

[a] average features per sample, that is, average unique terms per document.
[b] average labels per sample.

Table 2: Statistical information of data sets

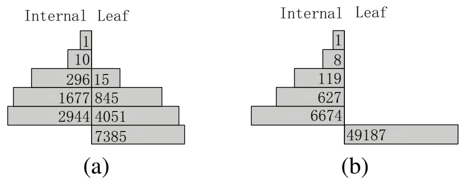

Figure 4: Number of internal and leaf nodes at each level of the hierarchy: (a) LSHTC; (b) NTCIR.

report the comparison with flat 1-vs-Rest approach on several subsets.

4.1 Experimental Settings

4.1.1 Data Sets

Two real-world data sets, the data set of web pages in the PASCAL2 Large-scale Hierarchical Text Classification challenge (LSHTC)[2] and the data set of patent documents from NII Test Collection for IR Systems Project (NTCIR)[3], are used in our experiments.

The PASCAL2 Large-scale Hierarchical Text Classification (LSHTC) challenge is held at 2009, aimed at promoting the study of classification methods for large hierarchies. The challenge attracts 19 participants with a variety of approaches (Kosmopoulos et al., 2010).

International Patent Classification (IPC) is a real-world taxonomy maintained by World Intellectual Property Organization (WIPO) [4]. The data set that we use is provided by NTCIR which is freely available for research purpose (Fall et al., 2003; Fujii et al., 2007). This data set consists of 3 496 137 Japanese patent documents submitted to Japan Patent Office from 1993 to 2002.

The statistics of two data sets and their hierarchies are presented in Tab. 2 and Fig. 4. Note that LSHTC's is a single-labeled task while NTCIR's is a multi-labeled ones.

4.1.2 Performance Measurement and Baseline Methods

Different performance measurements and baseline methods are adopted for the two data sets due to their difference of single-label and multi-label. NTCIR is multi-labeled, so the most commonly used criterion for general multi-labeled classifications, micro-F_1, is taken as the performance measurement. ScutTD is taken as the baseline method.

LSHTC is single-labeled, so accuracy is taken as the performance measurement. However, there is a problem about baseline method as ScutTD is not proper for single-labeled task. As a matter of fact, single-labeled hierarchical classifications are easier than multi-labeled ones, and it is natural to activate the child node with the largest score during top-down classification, like (Koller and Sahami, 1997). This method happens to be RcutTD with the parameter r=1. In addition to this baseline method, the evaluation records of LSHTC are also used for comparison.

4.1.3 Settings of MetaTD

The representation of meta-samples follows the description in Sec. 2.2.1. RcutTD is employed to select label candidates as described in Sec. 2.2.2. We set the parameter r=2 due to a trade-off between classification accuracy and time cost according to several pilot experiments.

The recent implement of SVM, Liblinear, is adopted as the meta-classifier (Fan et al., 2008).

Meta-to-base interpreters are needed to transfer the meta-level predictions into base-level labels. LSHTC is single-labeled, so it's natural to take the label with the largest meta-level scores. NTCIR is multi-labeled, and the strategy of S-cut in flat multi-class classification is employed.

4.1.4 Other Settings

The bag-of-word model with the term weight of TFIDF is adopted as the base-level sample representation in this paper (Sebastiani, 2002). To handle the Japanese text in the NTCIR's data set, we use the segment tool of Chasen [5] (Jin et al., 2010), and remove the function words from the result.

The base-level classifier is SVM^{light} with linear kernel. The default cost factor of SVM^{light} is used on NTCIR, while the cost factors are tuned by the development sets on LSHTC.

[2] `http://lshtc.iit.demokritos.gr/`
[3] `http://research.nii.ac.jp/ntcir/index-en.html`
[4] `http://www.wipo.int/classifications/ipc/en/`
[5] `http://chasen.naist.jp/hiki/ChaSen/`

LSHTC, 12 294 classes			
Rank	Method	Acc.	Group
1	Not reported	0.4676	alpaca
2	Committees of flat approaches	0.4632	jhuang
	MetaTD	**0.4513**	
3	Flattened Top-down method	0.4433	arthur.
4	Centroid-based classifier	0.4431	XipengQiu
5	Deep Classification	0.4317	Turing
6	Not reported	0.4270	Dyakonov
	RcutTD	**0.4262**	
7	Flattened Top-down method	0.4152	logicators
11	k-NN	0.4023	NakaCristo

Table 3: Classification accuracies on single-labeled LSHTC as well as its challenge records

NTCIR, 49 187 classes	
Method	Micro-F$_1$
ScutTD	0.272
MetaTD	**0.426**

Table 4: Classification accuracies on the multi-labeled data set of NTCIR

The experiments are run on four 64-bit computers with multi-core $1.9GHz$ AMD CPUs. All the experiments require actually up to $8G$ memory according to our observation.

4.2 Performance on Entire Data Sets

In this subsection we compare MetaTD with baseline methods on the entire data sets of LSHTC and NTCIR from the aspects of accuracy and efficiency.

4.2.1 Accuracy Comparisons

The experimental results on the single-labeled LSHTC as well as the challenge records are presented in Tab. 3. MetaTD turns out to be between the second and third place, while the baseline method of RcutTD ranks between the sixth and seventh place. The method at the second place is a committee of two flat approaches – variants of the OOZ algorithm (Madani and Huang, 2008) and the passive-aggressive algorithm (Crammer et al., 2006). The methods at both the third and seventh places are both top-down methods enhanced by flattening the original hierarchy. Deep classification ranks at the fifth place (Xue et al., 2008). In short, MetaTD outperforms the conventional and several variant top-down methods.

The results on the multi-labeled NTCIR are presented in Tab. 4. MetaTD achieves a much higher micro-F$_1$ than the baseline method of ScutTD.

Method	Training		Classify.[a]
LSHTC, 12 294 classes			
RcutTD	Train base-classifiers	$10h$	0.108
MetaTD	Train base-classifiers	$17h$	0.131
	Prepare meta-train. set	$1h$	
	Meta-training	$18s$	
NTCIR, 49 187 classes			
Scut/MetaTD	Train base-classifiers	$261h$	
ScutTD	Find optimal thresholds	$4h$	0.029
Meta-learning	Prepare training set	$12h$	0.062
	Meta-training	$466s$	

[a] seconds per sample

Table 5: Time costs of training and classifying with conventional top-down methods and MetaTD.

No. Class.	LSHTC			NTCIR		
	Train.	Dev.	Test	Train.	Dev.	Test
$1k$	$7k$	$2k$	$2k$	$38k$	$16k$	$17k$
$5k$	$39k$	$12k$	$12k$	$155k$	$64k$	$68k$
$10k$	$76k$	$23k$	$23k$	$253k$	$102k$	$104k$
$15k$	–[a]	–	–	$320k$	$129k$	$129k$

[a] there is not enough classes in the original data set.

Table 6: Numbers of classes and samples at the subsets of LSHTC and NTCIR

4.2.2 Efficiency Comparisons

The training and classifying time costs of MetaTD and baseline methods are presented in Tab. 5. In the training phrase, training base-classifiers causes most time cost. The additional cost of meta-classification MetaTD is only 5%–10% of that cost. Meta-training unexpectedly costs very little time, while preparing meta-training sets costs most additional time cost.

On the aspect of classifying, the time cost of MetaTD is about twice as much as the conventional top-down methods. According to our observation, considerable time is spent on reading samples and loading classifiers.

4.3 Comparison with Flat Approach of 1-vs-Rest on Subsets

In this subsection we compare the performance of top-down methods with the flat approach of 1-vs-Rest multiclass classification. Given the great computational complexity of the flat approach, several subsets are made from the entire data sets of LSHTC and NTCIR through randomly picking up classes and samples (see Tab. 6).

All the experimental settings here are consistent with previous experiments on the entire data sets. For the flat approach of 1-vs-Rest, the SVM^{light} is

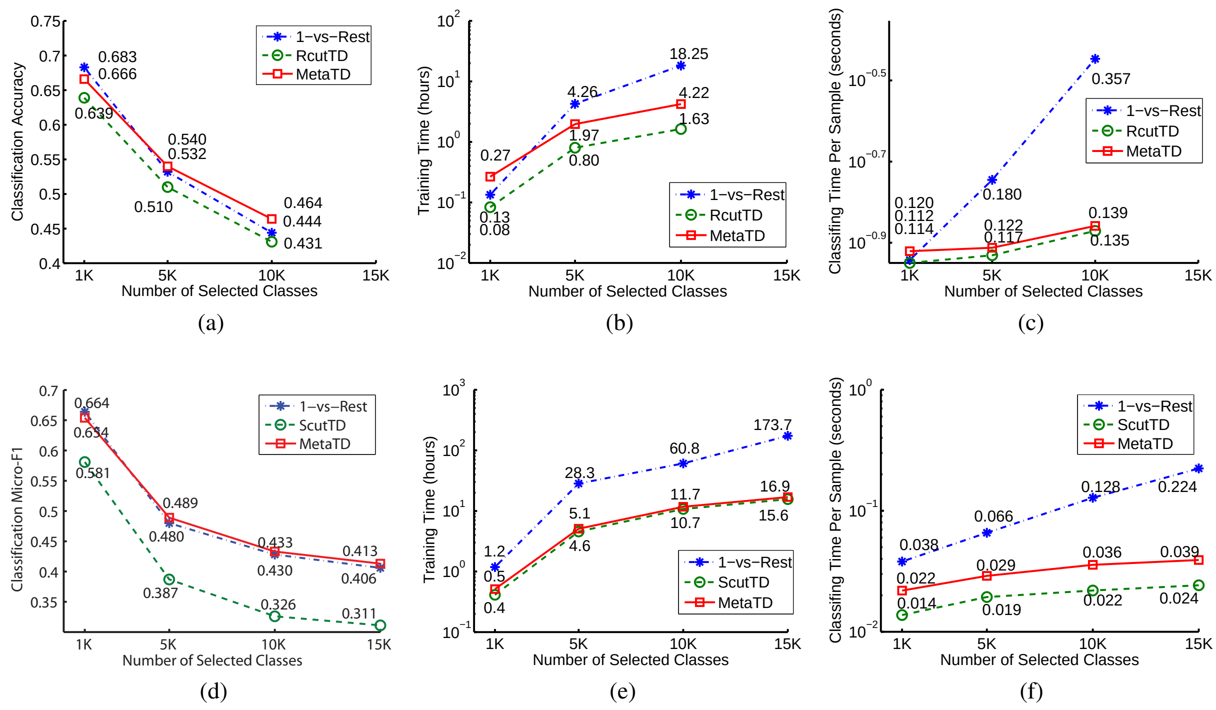

Figure 5: Performance comparison of flat 1-vs-Rest approach, conventional top-down methods and MetaTD on subsets of various sizes: (a) through (c) for LSHTC; (d) through (f) for NTCIR.

taken as the base-classifier.

The experiment results are presented in Fig. 5. On the aspect of classification accuracy, MetaTD catches up with the 1-vs-Rest approach. In particular, MetaTD slightly outperforms 1-vs-Rest approach on both data sets when the number of classes exceeds 5 thousands.

On the aspect of computational complexity, MetaTD is close to the conventional top-down methods, and they all show a great superiority over the 1-vs-Rest approach on both training and classifying as expected.

5 Conclusions

In this paper, we propose a meta-learning top-down method (MetaTD) in order to reduce the error propagation of the conventional ScutTD while remain its capability for large-scale hierarchical classification. In the experiments, MetaTD outperforms ScutTD and catches up with the flat 1-vs-Rest approach on classification accuracy. On the aspect of computational complexity, MetaTD only costs 5%-10% extra time in training and classifying, so it is suitable for most applications where ScutTD are being used.

References

P.N. Bennett and N. Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *Proc. of SIGIR'09*, pages 11–18. ACM.

L. Cai and T. Hofmann. 2004. Hierarchical document categorization with support vector machines. In *Proc. of ACM international conference on information and knowledge management*, pages 78–87. ACM.

M. Ceci and D. Malerba. 2007a. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78.

M. Ceci and D. Malerba. 2007b. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. 2006. Hierarchical classification: combining Bayes with SVM. In *Proc. of ICML'06*, pages 177–184. ACM.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

S. Dumais and H. Chen. 2000. Hierarchical classification of Web content. In *Proc. of SIGIR'00*, pages 256–263. ACM.

C.J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. 2003. Automated categorization in the international patent classification. In *ACM SIGIR Forum*, volume 37, pages 10–25. ACM.

R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

AA Freitas and A.C. de Carvalho, 2007. *A Tutorial on Hierarchical Classification with Applications in Bioinformatics.*, pages 175–208. IGI Publishing.

A. Fujii, M. Iwayama, and N. Kando. 2007. Introduction to the special issue on patent processing. *Information Processing & Management*, 43(5):1149–1153.

Gang Jin, Qi Kong, Jian Zhang, Xiaolin Wang, Cong Hui, Hai Zhao, and Bao-Liang Lu. 2010. Multiple strategies for NTCIR-08 patent mining at BCMI. In *Proc. of the 8th NTCIR workshop meeting on evaluation of information access technologies*, pages 303–308.

D. Koller and M. Sahami. 1997. Hierarchically classifying documents using very few words. In *Proc. of ICML'97*, pages 170–178.

A. Kosmopoulos, E. Gaussier, G. Paliouras, and S. Aseervatham. 2010. The ECIR 2010 large scale hierarchical classification workshop. In *ACM SIGIR Forum*, volume 44, pages 23–32. ACM.

Y. Labrou and T. Finin. 1999. Yahoo! as an ontology: using Yahoo! categories to describe documents. In *Proc. of the eighth international conference on Information and knowledge management*, pages 180–187. ACM.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

C. L. Liu, H. Hao, and H. Sako. 2004. Confidence transformation for combining classifiers. *Pattern Analysis & Applications*, 7(1):2–17.

T. Y. Liu, Y. Yang, H. Wan, H. J. Zeng, Z. Chen, and W.Y. Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations*, 7(1):36–43.

C. L. Liu. 2005. Classifier combination based on confidence transformation. *Pattern Recognition*, 38(1):11–28.

B.L. Lu and M. Ito. 2002. Task decomposition and module combination based on class relations: A modular neural network for pattern classification. *IEEE Tran. on Neural Networks,*, 10(5):1244–1256.

B.L. Lu and X.L. Wang. 2009. A Parallel and Modular Pattern Classification Framework for Large-Scale Problems. *Chen C. H. editor, Handbook of Pattern Recognition and Computer Vision (4th Edition)*, pages 725–746.

O. Madani and J. Huang. 2008. On updates that constrain the features' connections during learning. In *Proceeding of SIGKDD'08*, pages 515–523. ACM.

A. Montejo-Ráez and L. Ureña-López. 2006. Selection strategies for multi-label text categorization. *Advances in Natural Language Processing*, pages 585–592.

J. Platt. 1999. Probabilistic outputs for support vector machines. *Bartlett P., Schoelkopf B., Schurmans D., Smola, A. J. editor, Advances in Large Margin Classifiers*, pages 61–74.

K. Punera and J. Ghosh. 2008. Enhanced hierarchical classification via isotonic smoothing. In *Proceeding of the 17th international conference on World Wide Web*, pages 151–160. ACM.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

C. N. Silla and A. A. Freitas. 2010. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, pages 1–42.

A. Sun and E. P. Lim. 2001. Hierarchical text classification and evaluation. In *Proc. of the ICDM'01*, pages 521–528. IEEE.

A. Sun, E. P. Lim, W. K. Ng, and J. Srivastava. 2004. Blocking reduction strategies in hierarchical text classification. *IEEE Tran. on Knowledge and Data Engineering*, pages 1305–1308.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453.

X. L. Wang and B. L. Lu. 2010. Flatten hierarchies for large-scale hierarchical text categorization. In *Proc. of fifth international conference on digital information management*, pages 139–144.

G. R. Xue, D. Xing, Q. Yang, and Y. Yu. 2008. Deep classification in large-scale text hierarchies. In *Proc. of SIGIR'08*, pages 619–626. ACM.

Y. Yang, J. Zhang, and B. Kisiel. 2003. A scalability analysis of classifiers in text categorization. In *Proc. of SIGIR'03*, pages 96–103. ACM.

Y. Yang. 2001. A study of thresholding strategies for text categorization. In *Proc. of SIGIR'01*, pages 137–145.