ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Pattern Recognition ■■■ (■■■■) ■■■–■■■

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/pr

Robust and efficient multiclass SVM models for phrase pattern recognition

Yu-Chieh Wu[a], Yue-Shi Lee[b], Jie-Chi Yang[c,*]

[a]*Department of Computer Science, National Central University, 300 Jhongda Rd., Jhongli City, Taoyuan 32001, Taiwan*
[b]*Department of Computer Science, Ming Chuan University, 5 De-Ming Rd., Gwei Shan District, Taoyuan 333, Taiwan*
[c]*Graduate Institute of Network Learning Technology, National Central University, 300 Jhongda Rd., Jhongli City, Taoyuan 32001, Taiwan*

Received 29 December 2006; received in revised form 22 November 2007; accepted 24 February 2008

Abstract

Phrase pattern recognition (phrase chunking) refers to automatic approaches for identifying predefined phrase structures in a stream of text. Support vector machines (SVMs)-based methods had shown excellent performance in many sequential text pattern recognition tasks such as protein name finding, and noun phrase (NP)-chunking. Even though they yield very accurate results, they are not efficient for online applications, which need to handle hundreds of thousand words in a limited time. In this paper, we firstly re-examine five typical multiclass SVM methods and the adaptation to phrase chunking. However, most of them were inefficient when the number of phrase types scales. We thus introduce the proposed two new multiclass SVM models that make the system substantially faster in terms of training and testing while keeps the SVM accurate. The two methods can also be applied to similar tasks such as named entity recognition and Chinese word segmentation. Experiments on CoNLL-2000 chunking and Chinese base-chunking tasks showed that our method can achieve very competitive accuracy and at least 100 times faster than the state-of-the-art SVM-based phrase chunking method. Besides, the computational time complexity and the time cost analysis of our methods were also given in this paper.
© 2008 Elsevier Ltd. All rights reserved.

Keywords: Machine learning; Multiclass classification; Natural language processing; Support vector machines

1. Introduction

Phrase pattern recognition (phrase chunking) is a task in which phrase structures in text are detected and classified into predefined types such as noun phrase (NP), verb phrase (VP), prepositional phrase (PP), etc. These phrases are non-recursive, non-overlap, i.e., they cannot be included in other chunks [1]. Phrase structures provide important and useful syntactic information for downstream applications. Examples include text mining [2], text categorization [3], named entity recognition [4], chunk-based machine translation [5], semantic role labeling (SRL) [6,7], and bottom-up chunk-based grammar parsing [8,9].

Over the past few years, a great amount of research studies stressed on developing high performance chunking models adopting various supervised machine learning algorithms such as SVM [10–14], voted-perceptrons [15], Winnow [16], and hidden Markov models (HMMs) [17]. Among them, the SVM-based method showed a great performance in terms of accuracy. However its inefficiency in actual use limits practical purpose, like information retrieval and extraction (IE and IR), and question answering (QA) which should take tens of thousand related articles into consideration in very short time. The major limitation of SVM is that it is essentially a binary classifier, which needs to decompose multiclass to several binary categories. However, adopting existing multiclass SVM models could not provide an efficient phrase chunking time performance. For example, Wu et al. [11–13] employed one-versus-all (OVA) decomposition method for SVM, and ran at a rate of 1200 words per second. Kudoh and Matsumoto [10] combined eight SVM classifiers with one-versus-one (OVO) SVMs, which could only handle 20–30 words in one second. Although these methods showed an excellent success in terms of accuracy, to handle large-scale classification tasks for online purpose, designing an efficient and robust multiclass SVM model is indispensable.

* Corresponding author. Tel.: +886 3 4227151x35414; fax: +88 6 3 4275 336.
E-mail address: yang@cl.ncu.edu.tw (J.-C. Yang).

0031-3203/$30.00 © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.02.010

On the other hand, some of the light-weight and efficient chunking models [17,18] can swift process tens of thousand words on single process. But these methods are not accurate. Even feeding the hand-annotated training data in several times larger, the performance is still worse than SVM-based methods. For example, the HMM-based chunking models [17] could recognize about 40 000 terms per second, while achieved 92.19 in $F_{(\beta)}$ rate (defined in Section 4.1) for the CoNLL-2000 chunking task. When the training size scales to four times larger (about 1 million words), the performance was still worse than the SVM-based methods (93.25 versus 94.12 reported by Ref. [12]).

In this paper, we propose two new strategies to speed-up the SVM for phrase chunking problems. One is C-OVA (constraint one-versus-all), which is an extension of conventional one-versus-all decomposition scheme. The other is HC-OVA (hierarchical constraint one-versus-all) where the classification process can be viewed as visiting a constraint binary tree. We also re-examine and compare with five typical types of multiclass SVM models in terms of theoretical computational time complexity analysis and experimental results for chunking tasks. To the best of our knowledge, there is no work that had presented not only the efficient SVM-based phrase chunking methods yet the sizable and more complete comparative studies for the five typical multiclass SVM models. The experimental results on the CoNLL-2000 and Chinese base-chunking tasks show that our methods perform at least 100 times faster than the state-of-the-art SVM-based chunking models without a discernible change in accuracy. Besides, the two methods can also be applied to other similar tasks, such as named entity recognition [4,19], and Chinese word segmentation [20].

2. SVM-based phrase chunking models

Support vector machines (SVM) [21] was originally designed for binary classification problems. How to effectively cast the multiclass problems is still an on-going issue. Currently there are five typical types of multiclass SVM, namely OVA, OVO, directed acyclic graphs (DAG), error-correcting output codes (ECOC), and tree-based. In this section, we briefly introduce SVM at the beginning and review these typical multiclass SVM models. At the end of this section, we describe the use of SVM for phrase chunking.

2.1. Support vector machines

Suppose we have the training instance set for binary classification problem:

$$(x1, y1), (x2, y2), \ldots, (xN, yN) \quad xi \in \Re^D, \quad yi \in \{+1, -1\}$$

where x_i is a feature vector in D-dimension space of the ith example, and y_i is the label of x_i either positive or negative. To determine the class (+1 or −1) of an example x can be judged by computing the following equation:

$$y(x) = \text{sign}\left(\left(\sum_{x_i \in SVs} \alpha_i y_i K(x, x_i)\right) + b\right) \tag{1}$$

α_i is the weight of training example x_i ($\alpha_i > 0$), and b denotes as a threshold. Here the x_i should be the support vectors (SVs), and represent the separation between different classes, as they only lie along the separating hyperplane. The kernel function K is the kernel mapping function, which might map from R^D to $R^{D'}$ (usually $D \ll D'$). The natural linear kernel simply uses the dot-product as

$$K(x, x_i) = dot(x, x_i) \tag{2}$$

A polynomial kernel of degree d is given by

$$K(x, x_i) = (1 + dot(x, x_i))^d \tag{3}$$

One can design or employ off-the-shelf kernel types for particular applications. For example, the polynomial kernel-based SVM was shown to be the most successful kernels for many natural language processing (NLP) problems, such as part-of-speech (POS) tagging [22], phrase chunking [10,12], and propositional parsing [6,7].

It is known that the dot-product (linear form) represents the most efficient kernel computing which can produce the output value by linearly combining all SVs such as

$$y(x) = \text{sign}(dot(x, w) + b), \quad \text{where } w = \sum_{x_i \in SVs} \alpha_i y_i x_i \tag{4}$$

By combining Eqs. (1) and (3), the determination of an example of x using the polynomial kernel can be shown as follows:

$$y(x) = \text{sign}\left(\left(\sum_{xi \in SVs} \alpha iyi(dot(x, xi) + 1)^d\right) + b\right) \tag{5}$$

Usually, degree d is set more than 1 because when d is set as 1, the polynomial kernel backs-off to linear kernel. Although the effectiveness of polynomial kernel, it cannot be shown to linearly combine all SVs into one weight vector whereas it requires computing the kernel function (3) for each SV x_i. The situation is even worse when extending the binary classifier to multiclass problems. Besides, the training cost of polynomial kernel is far higher than linear kernel. Previous literatures [10,12,22] showed the effectiveness but high-computation cost when using the polynomial kernel. In Wu's [12] study, the use of polynomial kernel ($d = 2$) SVM showed a slightly improvement than linear kernel (94.12 versus 94.20 in $F_{(\beta)}$ rate for CoNLL-2000 chunking task), while the training time and testing speed scaled, respectively, from 3 hours to 4 days and 1200 terms/s to 5–10 terms/s.

2.2. Multiclass SVMs

As described above, to extend SVM to multiclass, it needs to decompose the multiclass to several binary categories. In this paper, we focus on comparing with the following five multiclass SVM types, namely, OVA [23], OVO [24], DAG [25], ECOC [26,27], and tree-based [28] which were successfully employed to sequential pattern recognition and NLP tasks. The OVA simply creates C binary SVMs for all categories, where

C is the number of class. For i-th SVM only the examples in the i-th class are viewed as positive instances while the remaining ones belong to negative. Both Refs. [12] and [22] showed very competitive results for POS tagging and phrase chunking by means of OVA-based multiclass SVM. Another multiclass SVM model is called OVO. It creates pairwise SVM for arbitrary class pairs. In other words, OVO constructs $C(C\text{-}1)/2$ SVMs to solve the multiclass problem. The class of a new item is mainly determined through majority voting by pairwise classifiers. The Kudoh's chunking system [10] was designed based on this model.

The third type is: DAG [25], which integrated the training schema of OVO and graph visiting strategy for testing. Its training phase is the same as OVO model. However, in testing phase it visits the DAG by incrementally removing an irrelevant category. This graph is a rooted binary acyclic graph which has $C(C-1)/2$ internal nodes and C leaves. Each internal node represents a binary SVM for two classes. To determine the class of the given example, DAG starts from the root, and moves left or right depending on the binary output of the SVM. When it moves left, the right class will no longer be compared in the following stage until the leaf node reaches.

Anther multiclass model is ECOC, which disambiguates the output Boolean codes from all binary classifiers. The main spirit of ECOC is to construct the code word matrix where row i represents the code vector of class i, and column j defines a split for binary classifier to learn. To determine the class, ECOC compares the generated bit vector with each row of the matrix. Usually, the row with minimum hamming distance is selected [26]. In basic, it requires at least $\log_2 C$ and at most $2^{C-1}-1$ bits to disambiguate C-classes. Usually, the longer the bit vector, the more correcting power (higher performance) it achieves [26,27]. But the scaled code word largely increases the training and testing time costs. On the contrary, the shorter the bit vector, the faster the classification time.

Unlike previous four methods, the tree-based multiclass SVM treats the multiclass classification process as the tree visiting. The leaf node of the tree represents the class, and the internal node can be viewed as a binary decision. It is clear to see that this method depends on the tree construction. Takahashi and Abe [28] presented both top-down and bottom-up clustering techniques with two different similarity measurements to build the tree. An extension of their work can be shown in Ref. [29] which made use of more elaborate similarity measurements and applied to all training examples. Vural and Dy [30] proposed divide-by-2 (DB2) algorithm that partitioned training set into almost equal two subsets. It is known that the "set partition" is an NP-hard problem [31], which exponentially scales. In addition, DB2 cannot partition the case like Chinese base-chunking in which the NP dominates 57% of the training set.

Additionally, we do not compare with some other multiclass SVM models, like multiclass SVM [32], and HMM-SVM [33] which linked the dependence among individual binary classifiers based on the OVA scheme. The multiclass SVM [32] has the same testing time complexity as the OVA model, while it scales the original dimensions with C times larger where C is the number of categories. Similarly, the HMM-SVM [33] has the similar training style as multiclass SVM, while it further takes the state(class)-transition estimations into account and also employed the Viterbi algorithm for training and testing. It is known that the Viterbi algorithm is a famous dynamic programming technique which is at least C times larger than conventional OVA in terms of testing time complexity. We do not compare the two methods here, since they were essentially the variant types of OVA, and required far training and testing time than the OVA, while achieved similar results. For example, the training time of the HMM-SVM[1] for the CoNLL-2000 chunking task (23 categories) was eight days. It willspend several weeks to train the Chinese base-chunking dataset, since 47 chunk classes should be learned.

Table 1
An Example for IOB1/2 and IOE1/2 chunk representation styles

Word	IOB1	IOB2	IOE1	IOE2
In	I-PP	B-PP	I-PP	E-PP
early	I-NP	B-NP	I-NP	I-NP
trading	I-NP	I-NP	I-NP	E-NP
in	I-PP	B-PP	I-PP	E-PP
Hong	I-NP	B-NP	I-NP	I-NP
Kong	I-NP	I-NP	E-NP	E-NP
Monday	B-NP	B-NP	I-NP	E-NP

2.3. Phrase chunking tasks

Ramshaw and Marcus [34] proposed the earliest inside/outside label style to represent NP chunks. This method involves in three tags, B, I, and O. I tag indicates the current word which is inside a chunk, B tag is used to represent the beginning of a chunk which immediately follows another chunk, O tag means the current word does not belong to a part of chunk. This method is also called IOB1. Tjong Kim Sang and Veenstra [35] derived the other three alternative versions, IOB2, IOE1, and IOE2.

IOB2: is different from IOB1, which uses the B tag to mark every beginning word of a chunk and the other inside terms are labeled as I tag.
IOE1: An E tag is denoted as the ending word of a chunk which is immediately before a chunk.
IOE2: The E tag is given for every word that is the end of a chunk.

Let us illustrate the four representation styles with an example, considering a clause, "In early trading in Hong Kong Monday", the four representation styles of the clause are listed in Table 1. This example also encodes the phrase chunk type with labeling the specific type behind the B/I/E tags. For example, the B-PP is used to represent the beginning of a PP in the IOB2 style.

[1] We use the SVM-struct toolkits which can be found at: http://svmlight.joachims.org/svm_struct.html.

By encoding with the IOB tags, we can reformulate the chunking problem as sequence of word classification [34]. The sequential word classification had been applied to many similar tasks such as full parsing [8,9], IE [2], SRL [6,7], etc. In general, the contextual information is often used as the basic feature type; the other features can then be derived based on the surrounding words, such as words and their POS tags. The chunk class of the current word is mainly determined by the context information. By following [10–13], we use the following context feature types to represent a training example. Examples of the representations for the features can be found in Refs. [10–13].

- Lexical information (unigram/bigram).
- POS tag: unigram_POS, bigram_POS, and trigram_POS information were encoded.
- Affix ($2 \sim 4$ suffix and prefix letters).
- Previous chunk information: unigram_chunk, and bigram_chunk were considered.
- Orthographic feature type [12,13].
- Possible chunk classes: the possible chunk class of current word.
- Word+POS bigram (current token+next token's POS tag): it represents the combination of the current word and its next word's POS tag as a bigram feature.

In addition, the chunking directions can be reversed from left to right into right to left. The original left to right chunking process classifies word with the original directions, i.e., the class of the current word is determined after chunking all preceding terms of it. In the reverse version, the chunking process begins from the last word of the sentence to the first term. We name the original chunking process as *forward chunking*, while the reverse process as *backward chunking*.

3. Efficient multiclass SVM models

In terms of testing time complexity, most multiclass models explicitly perform classification for each category other than tree-based methods. That is to say when the number of class scales to high, the testing speed will be reasonably reduced, in particular to OVO. To remedy this, we propose two new strategies that make the SVM-based chunking substantially faster. The first two sections, we present the proposed two methods. The comparison and computational time complexity analysis of our methods are discussed in Section 3.3.

3.1. Hierarchical constraint-one-versus-all

When employing the IOB-like encoding styles, the scaled number of phrases directly increases twice number of chunk class. In the CoNLL-2000 chunking task, the 11 phrase types produce $11*2+1=23$ chunk classes, and 47 chunk classes are derived from 23 phrase types for the Chinese base-chunking. The situation is even more aggravated when using more specific labeling styles, like BIESO that explicitly indicates the begin, interior, end, single, and outside of a chunk type since the scaled chunk types is four times larger. However about half of the chunk classes are unnecessarily classified. For example, if the previous word was classified as B-NP, then its next term is impossible to be I-VP, I-PP, etc.

Table 2
Consistent matrix

Class Pair	B-NP	B-VP	I-NP	I-VP	O
B-NP	1	1	1	0	1
B-VP	1	1	0	1	1
I-NP	1	1	1	0	1
I-VP	1	1	0	1	1
O	1	1	0	0	1

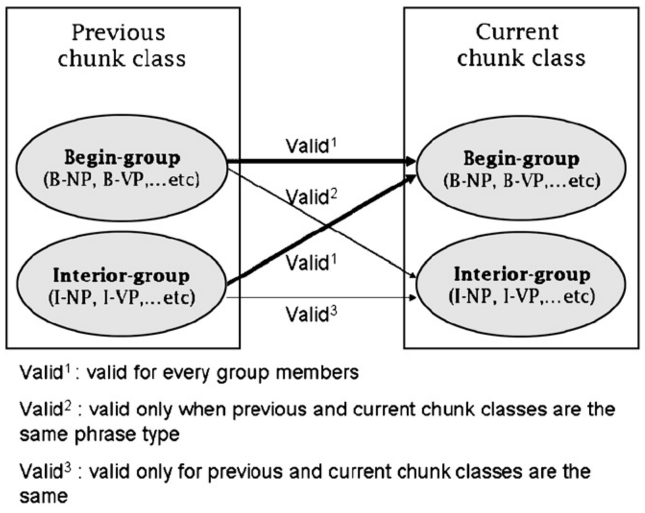
Previous chunk class
Current chunk class
Begin-group (B-NP, B-VP,...etc)
Interior-group (I-NP, I-VP,...etc)
Begin-group (B-NP, B-VP,...etc)
Interior-group (I-NP, I-VP,...etc)
Valid¹
Valid²
Valid¹
Valid³
Valid¹ : valid for every group members
Valid² : valid only when previous and current chunk classes are the same phrase type
Valid³ : valid only for previous and current chunk classes are the same

Fig. 1. The validity relationship between Begin- and Interior-groups.

To solve this, a consistent matrix is therefore used, which marks the validity of previous and current chunk classes. For example, Table 2 lists an example of the consistent matrix of NP and VP types using IOB2. The first column indicates the chunk class of previous word while the first row gives legal chunk classes of current term. Intuitively, if previous word was classified as B-NP, then the current chunk class should belong to one of the four chunk classes, B-NP/B-VP/I-NP/O, whereas I-VP is invalid. Furthermore, we can generalize this relation to the other three IOB styles, IOB1, IOE1, and IOE2. Here, we define that all chunk classes should belong to one of the following two groups:

Begin-group: start of a chunk, for example, the B-tag for IOB2, I-tag for IOE2.
Interior-group: inside of a chunk, for example, I-tag for IOB2, and E-tag for IOE1.

Fig. 1 illustrates the relationships between the two groups. Based on the relationship, we can manually build the consistent matrix by connecting the validity between the two groups. That is, whether previous chunk class is Begin-group or interior-group, the chunk class of current word potentially belongs to

```
      Algorithm 1: Auto-construct consistent matrix
      Notation: IOB-tag = {I, B, E} (Obviously, O tag should be the Begin-group)
                Phrase type = {NP, VP, PP,…}
                Chunk class is the combinations of each chunk type and IOB-tag

1:)   Initialize the consistent matrix (CM) by verifying the valid chunk class pairs of previous and current tokens in the training
      data
2:)   Get IOB-tag and set them as Interior-group initially. (for example, B-NP => B, I-NP => I)
3:)   For each IOB-tag {
4:)   For each chunk class pair ($Ch_j$ and $Ch_k$) with the same IOB-tag where $j \neq k$ {
5:)   Check whether there exists a valid item between $Ch_j$ and $Ch_k$ in CM
6:)   If the validity exists, then this IOB-tag is assigned to Begin-group
          }//End for
      }//End for
7:)   Re-organizes CM with the relations between Begin-group and Interior-group.
```

Fig. 2. A consistent matrix construction algorithm.

any member of the Begin-group or "a specific" chunk class of the interior-group in which the phrase type is equivalent to previous one. In other words, from Begin/Interior-groups to Begin-group is valid, while from Begin/Interior-groups to Interior-group is valid only when they share the same chunk.

Obviously, for IOB2 representation style, it is not difficult to annotate the consistent matrix by hand. However, for the other three representation styles together with forward and backward chunking directions, it requires lots of human efforts especially to a large amount of phrase types, like Chinese (47 chunk classes). For different chunking tasks, manual-development of consistent matrix is needed. To generalize the matrix for different purpose, an automatic approach is required.

Hence we propose an algorithm to generate the consistent matrix automatically without considering the used representation method is IOB or IOE labeling styles. Fig. 2 shows the proposed algorithm.

As outlined in Algorithm 1, the first two steps aim to initialize the consistent matrix by scanning the previous-current chunk class pairs and extracting the IOB tag from training data. At third step, we check the consistency of arbitrary chunk class pairs of the same IOB-tag in the initial consistent matrix. If the two chunk classes are valid, then the IOB tag belongs to Begin-group. For Interior-group, the validity only exists when previous and current chunk share the same phrase type. Once the Begin and Interior-groups are determined, then the final consistent matrix could be generated by connecting the relationships between the two groups. The final line of the algorithm finally fills the matrix according the classified group. It is worth to note that we ignore the O-tag here since it should be the Begin-group clearly.

Let us illustrate with a simple example, Table 3 lists the initial consistent matrix that is gathered from the training data. In this example, the IOB tags are B and I which are set to Interior-group initially. For B tag, we find that the validity also occurs in the chunk class pair, (B-NP, B-VP) in which they share the same IOB tag (i.e., B tag) in the initial consistent matrix. Thus, B tag is assigned to Begin-group. For I tag, after scanning all pairs, we conclude that the validity only exists when previous and current chunk classes belong to the same phrase type such as (I-NP, I-NP), (B-NP, I-NP), (B-VP, I-VP). Therefore I tag is still

Table 3
Initial observed consistent matrix

Class Pair	B-NP	B-VP	I-NP	I-VP	O
B-NP	1	1	1	0	1
B-VP	0	0	0	1	0
I-NP	1	0	1	0	1
I-VP	0	0	0	0	0
O	1	0	0	0	1

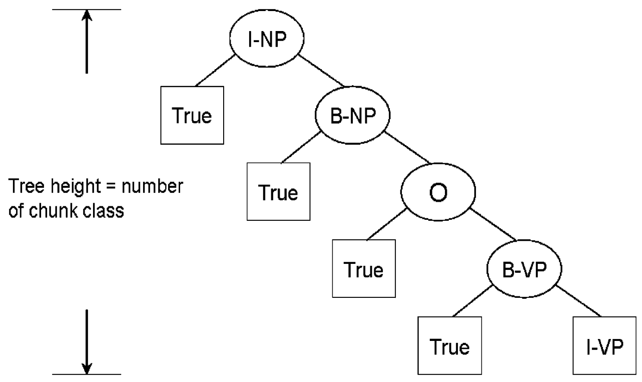

Fig. 3. Skewed tree structures for multiclass chunking strategy.

in the Interior-group. After classifying Begin/Interior-groups for each IOB tag, the final consistent matrix is re-organized by bridging the valid relationship between the two groups. That is, we set the column of each Begin-group as valid, and the validity of the Interior-group is valid only when its previous chunk is the same as current chunk. After this step, we can generate the final consistent matrix as Table 2.

Although the consistent matrix method can reduce half classification times for SVM, if the number of phrase type is still large, classifying is remaining time-consuming. Empirically, in the chunking task we found that about half ratio of the words belong to NP (in English and Chinese). In particular to the three phrase types, NP, VP, and PP that cover 90% of the training data. In Chinese base-chunking, the proportion of NP in the training set is 57% and the three phrase types NP, ADVP and VP denominate 83%. In most cases, the chunk class of each word

```
Algorithm 2: HC-OVA training (Chunk_class $C$, set_of_labeled_training examples)
Notation: $C$ is the number of chunk classes

1:)  Construct the consistent matrix using Algorithm 1.
2:)  Build the ordering list: Or[$i$] based on counting the frequency of each chunk class in the training
     set. The Or list stores all chunk classes.
3:)  For $i$ := 1 to $C$-1
4:)      Collecting the positive examples from chunk class Or[$i$]
5:)          For $j$ := $i$+1 to $C$
6:)              Collecting negative examples from chunk class Or[$j$]
7:)      Training SVM for chunk class Or[$i$]
```

Fig. 4. Hierarchical constraint one-versus-all algorithm for multiclass SVM training.

```
Algorithm 3: HC-OVA testing (list Or, term $t_i$)
Notation: $t_i$: term $_i$; $Ch_i$: is the chunk class of $t_i$.
1:)  For each $t_i$, {
2:)      For $j$ := 1 to $C$ {
3:)          /*  check the validity between previous and current chunk classes


             If (consistent ($Ch_{i-1}$, Or[$j$])) {
4:)              $Ch_i$ := Or[$j$]
5:)              Classify $t_i$ with SVM Or[$j$]
6:)              If $t_i$ is classified as positive, then
7:)                  Stop classifying
             }//End if
         }//End for
     }//End for
```

Fig. 5. Hierarchical constraint one-versus-all algorithm for multiclass SVM testing.

should be one of those high-frequent classes. Therefore, we design a new multiclass strategy for SVM to restrict the classification of high-frequent chunk classes at higher priority. The main spirit of this method is to reduce unnecessary classifications through classifying the high-frequent chunk class first. This schema can be illustrated with a skewed binary tree as in Fig. 3.

Every internal node in Fig. 3 represents a decision of one class and the remaining non-visited classes. Each leaf node denotes as a chunk class, and the higher frequent the chunk class observed, the higher level it is in the tree. Determining the chunk class of a term is equivalent to visit the skewed binary tree. Once the leaf node is visited at higher level, the remaining nodes will no longer be classified. Thus, most rare chunk classes are not compared in testing phase. Combining with the consistent matrix, some of the internal nodes can be further ignored without performing SVM classifying by checking the validity. The overall training and testing algorithms are outlined in Figs. 4 and 5.

As outlined in Algorithm 2, we construct an ordering list via estimating the frequency in the training data. For example, as shown in Fig. 2, Or[0] is I-NP. These chunk classes are trained by following this order (the third step). Note that we discard a subset of training example that belongs to previous chunk class repeatedly (see steps 5 and 6 in Algorithm 2). On the other hand, the testing algorithm (Algorithm 3) is performed by following the ordering list and the built consistent matrix (by Algorithm 1). Hence, the consistent matrix can be used to limit the classification only when previous-current chunk class pair is valid in the matrix.

3.2. Constraint-one-versus-all

The main idea of constraint OVA method is to reduce half of the comparison times using the consistent matrix only. During testing, we only put emphasis on picking up the whole Begin-group and one from Interior-group. For the beginning of a sentence, the Begin-group is only taken into consideration. As described in Section 3.1, the Interior-group should follow the Begin-group with the same phrase type. For the previous example, three chunk classes belongs to Begin-group, B-NP, B-VP, and O, while the I-NP and I-VP were assigned to Interior-group using Algorithm 1. In testing, for the first word, we only focus on classifying with all of the chunk classes in the Begin-group, i.e. B-NP, B-VP, O. If the first word is classified as B-NP then the second term should not be the other chunk classes in Interior-group other than I-NP. Therefore the B-NP, B-VP, O and I-NP should be used for classifying.

Based on this constraint, the training examples of the Interior-group could be further reduced. It is impossible to compare two chunk classes of the Interior-group simultaneously. On the contrary, the Begin-group is invited to be classified with a specified chunk class of Interior-group. In basic, we cannot reduce any training example for Begin-group. Overall, training the chunk class of the Begin-group is equivalent to the conventional OVA method, while training the Interior-group, only one chunk class is used against the whole Begin-group.

Table 4
Training/testing time complexity of different multiclass models

Multiclass SVM model	Balanced data distribution		Unbalanced data distribution	
	Training	Testing	Training	Testing
One-versus-all (OVA)	CN	C	CN	C
One-versus-one (OVO)	$(C-1)N$	$C(C-1)/2$	$(C-1)N$	$C(C-1)/2$
Directed acyclic graphs (DAG)	$(C-1)N$	$C-1$	$(C-1)N$	$C-1$
Dense error correcting output codes (Dense-ECOC)	$(\log C)N$	$\log C + C\log C$[a]	$(\log C)N$	$\log C + C\log C$[a]
Balanced tree-based	$(\log C)N$	$\log C$	$(\log C - 1)N$	$\log C$
Unbalanced tree-based	$(C+1)N/2$	$1\sim C-1$	$N\sim(C-1)N$	$1\sim C-1$
Constraint-OVA (C-OVA)	$B+(C-B)(B+1)N/2$	$B+1$	$BN\sim(B+1)N$	$B-1$
Hierarchical constraint-OVA (HC-OVA)	$(C+1)N/2$	$1\sim B+1$	N	$1\sim B+1$

[a]The dense-ECOC requires not only the $\log C$ SVM classification times, but also $C\log C$ times for decoding.

3.3. Computational time complexity analysis

Recent studies [36–38] had presented that the linear kernel SVM could be trained in linear time, i.e., $O(N)$ where N is the number of training examples. Based on this hypothesis, we can derive the training time complexity for multiclass SVM models. For OVA, the training time complexity is $O(CN)$ which involves in performing C times leaning on the complete training set. Testing time of OVA is intuitively $O(C)$.

Different from OVA, the OVO constructs $\binom{C}{2}$ SVMs for arbitrary two class pairs. When training data is balanced distributed, i.e. there are N/C examples in each category, the training time complexity is

$$\binom{C}{2}\left(\frac{2N}{C}\right) = \frac{C(C-1)}{2}\frac{2N}{C} = (C-1)N$$

When data is unbalanced distributed, all training examples centralize to a specific class, the training time complexity is $O(C-1)N)$. It is clear that the DAG-based multiclass SVM had the same training time complexity as OVO method, while testing time complexity was less than OVA one time, i.e., $O(C-1)$. The main idea of DAG is based on the OVO learning and its testing phase is equivalent to visit the acyclic graph.

The derivation of the tree-based method is somewhat complicated. We discuss it in four cases, balanced/unbalanced trees joint with unbalanced and balanced data distributions. Since our focus is not the proofs here, we leave the detail derivations in Appendix A. By means of the inference in Appendix A, we show that the tree-based method is a very efficient model in terms of training and testing. Table 4 lists the detail time complexity of the above methods. Note that the balanced tree denotes as the tree is height balanced.

We do not circumstantially derive the time complexity of the adopted ECOC approach that depends on the code word length. In this paper, we only employ the "dense" encoding method to generate code matrix, i.e., encode C categories with $\log C$ Boolean bits. Using the dense encoding for ECOC, the training time complexity is equivalent to the balanced tree-based models, while it requires $\log C$ times SVM classification plus $C\log C$ times for decoding the code vectors. In contrast, the dense-ECOC needs additional decoding costs than the balanced tree-based methods. One can also adopt various encoding methods for ECOC, like OVO and BCH codes, however, when the code length largely scales, the training and testing times greatly increased for the chunking task. In our experiments (Tables 6 and 7), the testing time of OVO that is quadratic scaled is 10–50 times larger than the other types where it requires 60 s to classify 47 000 words.

On the other hand, we analyze the time complexity of the proposed two methods, C-OVA, and HC-OVA. In terms of HC-OVA, the training time complexity is only $O(N)$ when data biases to one class. In the balanced case, each category contains roughly N/C examples so that we have,

$$\left(\frac{C}{C}N\right) + \left(\frac{C-1}{C}N\right) + \cdots + \left(\frac{3}{C}N\right) + \left(\frac{2}{C}N\right)$$

As described in Section 3.1, at each time, we use single SVM learns to classify one class. At the next step, the training data is reduced by removing the positive examples of previous step. In the balanced case, we can almost remove N/C examples at each step from the whole set. The above equation can be re-written and simplified as follows:

$$\left(\frac{2}{C}N\right) + \left(\frac{3}{C}N\right) + \cdots + \left(\frac{C-1}{C}N\right) + \left(\frac{C}{C}N\right)$$
$$\leqslant \frac{1+2+\cdots+C}{C}N = \frac{(C+1)}{2}N$$

For testing time complexity, in the best case HC-OVA only compares once in testing, i.e., $O(1)$, while in the worst case, it should compare all of the chunk classes that belong to Begin-group and a class from Interior-group, i.e., $O(B+1)$ where B is the number of chunk classes in the Begin-group. In contrast, the training time complexity of C-OVA in the balanced case is

Begin-group: $O(BN)$
Interior-group: $O((C-B)(B+1)N/C)$
Overall: $O(B+(C-B)(B+1)N/C)$

In the unbalanced case, when dataset biased to the Begin-group, the training time complexity is $O(BN)$. On the contrary, if the dataset biased to a chunk class of Interior-group, the overall training time is $O((B+1)N)$. For the testing time, C-OVA compares the whole Begin-group and one chunk class from

Table 5
Top five multiclass SVM models for training and testing

Rank	Name	Distribution type	Training time complexity	Rank	Name	Testing time complexity
1	HC-OVA	Unbalanced	N	1	Balanced-tree	$\log C$
2	Unbalanced-tree	Unbalanced	$N \sim (C-1)N$	2	HC-OVA	$1 \sim B+1$
3	Balanced-tree	Balanced	$(\log C)N$	3	Unbalanced-tree	$1 \sim C-1$
4	Dense-ECOC	–	$(\log C)N$	4	C-OVA	$B+1$
5	HC-OVA	Balanced	$(C-1)N/2$	5	DAG	$C-1$

Interior-group, thus, the time complexity is precisely $O(B+1)$ which is the same as the worst case of HC-OVA.

A full comparison of multiclass SVM models and our methods is listed in Table 4. We also summarize the most efficient top five approaches for training and testing in Table 5. As summarized in Table 5, when data is unbalanced distributed, the best "training time" can be compassed with HC-OVA, and unbalanced tree-based method. If the data is balanced distributed, then the tree-based and dense-ECOC are the best choices. In terms of "testing time", the most efficient models are: tree-based, dense-ECOC, and HC-OVA. It is important to note that we could not exactly analyze the actual testing time complexity for HC-OVA and unbalanced tree-based approaches, since they depend on the testing data and the actual SVM performance. In the next sections, we will present experimental results and the analysis of the actual testing time costs on the real data.

In NLP community, most data are unbalanced distributed where relatively small parts of categories cover most proportion. This situation is also true in many similar tasks, like named entity recognition and SRL. In statistics, NP, VP, and PP dominate 90% of the training data in the CoNLL-2000 chunking task. Alternatively 83% of the training data of the Chinese base-chunking task belong to NP, VP, and ADVP. In such an unbalanced case, our methods (HC-OVA) are very suitable for phrase chunking tasks.

4. Experiments

In this section, we exploit the performance of our methods on the real data. Sections 4.1 and 4.2 describe the experimental settings and the used data. Section 4.3 presents the experimental results of our methods and the other multiclass SVM models. Finally we compare our method to the published studies in Section 4.4.

4.1. Dataset and evaluation metrics

To fairly compare with related studies, we use the chunking dataset from CoNLL-2000 shared task [39], which is the standard benchmark and widely evaluated in many research studies, e.g., [10–12,15–17,19,39]. The benchmark was derived from the English Peen-Treebank Wall Street Journals (WSJ) where sections 15–18 were used for training and section 20 for testing. The dataset contain tokens (words and punctuation marks, etc), information about the location of sentence boundaries, auto-labeled POS tag information, and information of chunk boundaries (i.e., IOB2 representation style). The POS tag of this data had been labeled by a standard POS tagger [40] in order to reflect the realistic performance rates for which no human-made POS tags are available. In this dataset the chunk tag had been represented with the IOB2 methods. Users can employ the training data to train a chunking system. In testing the system should be able to predict the chunk tag of each word in the testing data and the performance was evaluated by how accurate the system is. It is a public benchmark corpus and available.[2] The organizers [39] also provide a perl-script[3] which enables users to convert the original Treebank structures into chunk structures.

In CoNLL-2000 chunking task, there are $11*2+1=23$ chunk classes (11 phrase types with B/I or I/E tags plus an outside tag O). We also use the Chinese Peen-Treebank to evaluate the efficiency of the scaled chunk classes. The front 0.28 million words were used for training, while the remaining 0.08 million words for testing. In Chinese, the 23 phrase types produce to $23*1+1=47$ chunk classes. However, there is not a standard POS-tagger for Chinese, instead, we simply use the gold hand-annotated tags.

The performance of the chunking task is usually measured with three rates, namely recall, precision, and $F_{(\beta=1)}$ [39]. First, the recall rate is to estimate the ratio of *phrases* found by the system. Second, the precision rate measures the percentage of the predicted *phrases* that are correct. Finally, the $F_{(\beta=1)}$ rate combines both recall and precision rates into one single measurement by the following:

$$F_{(\beta=1)} = \frac{2 * \text{recall} * \text{precision}}{\text{precision} + \text{recall}}$$

For a fair comparison, we use the perl-script evaluator released by CoNLL[4] to evaluate the three measures for the following experimental results.

4.2. Settings

Before experiments, we try to optimize the settings of adopted multiclass SVM methods such as DAG and tree-based models. To determine the classification order for DAG is somewhat ad hoc. We had conducted many trials and found that the natural order (order by alphabet) achieved the optimal

[2] The dataset is available at: http://www.cnts.ua.ac.be/conll2000/chunking/.
[3] Is available at: http://ilk.kub.nl/~sabine/chunklink/.
[4] See http://lcg-www.uia.ac.be/conll2000/chunking/conlleval.txt.

Table 6
Experimental results of different multiclass SVM strategies on CoNLL-2000 chunking task

CoNLL-2000	Training time (h)	Speed-up ratio	Routine processing time	SVM classification time (47 377 words) (s)	Speed-up ratio (%)	$F_{(\beta)}$
OVA	2.09	240%	2.38 s	2.09	436	94.25
OVO	1.27	146%		54.34	11321	**94.35**
DAG				4.73	986	94.31
Dense-ECOC	3.38	388%		1.53	316	90.73
Tree-based	1.04	119%		1.82	380	93.89
C-OVA	2.09	239%		1.01	211	94.25
HC-OVA	**0.87**	100%		**0.48**	100	94.10

The chunking speed of HC-OVA is about $47\,377/(2.38 + 0.48) \cong 16500$ terms/s while the OVO is 835 terms/s.

Table 7
Experimental results of different multiclass SVM strategies on chinese base-chunking task

CoNLL-2000	Training time (h)	Speed-up ratio (h)	Routine processing time	SVM classification time (47 377 words)	Speed-up ratio	$F_{(\beta)}$
OVA	~ 7	241	3.53 s	8.125 s	400	92.30
OVO	~ 5	172		> 5 min[a]	> 10000	**92.34**
DAG				> 5 min[a]	> 10000	**92.34**
Dense-ECOC	9.5	327		4.07 s	200	90.07
Tree-based	4.1	141		8.59 s	423	92.24
C-OVA	6.9	237		5.57 s	274	92.31
HC-OVA	**2.9**	100		**2.03 s**	100	92.22

The chunking speed of HC-OVA is about $85\,409/(3.53 + 2.03) \cong 15\,600$ terms/s while the OVO is about 100 terms/s.

[a]The OVO/DAG were performed in an inefficient data structures due to the quadratic scaled categories and support vectors that could not be processed with efficient arrays.

performance in both chunking tasks. For the tree-based method, we adopted the average link-based bottom-up clustering algorithm to build the binary tree instead of DB2. Recall that DB2 is an NP-hard algorithm, which is impractical and cannot partition the Chinese data (NP dominate more than half of the training set). We use the mean vector (centroid) to represent each chunk class and the clustering algorithm merged the two closest clusters (highest cosine value) until there was only one group.

In this paper, we employ SVM$^{\text{light}}$ [41] as the classification algorithm, which has been successfully applied to many classification problems. As reported by Refs. [12,22] working on linear kernel is far more efficient than polynomial kernels. To take the time efficiency into account, we choose the linear kernel type.

To further enhance the chunking result, we utilize the mask method (proposed by Refs. [11,12]) to create more training examples. We encourage readers to see the literatures [11,12] in detail. The setting of the mask method was the same as the original studies. In addition we use IOE2 with backward chunking direction for CoNLL-2000 chunking and IOB2 with forward direction for Chinese. Words that only appear once in the training data are not indexed as features. These settings had been shown to be very effective in previous researches [10–12,15,16,19].

4.3. Results for multiclass SVM models

The actual chunking performances of different multiclass SVM methods are listed in Table 6 (CoNLL-2000) and Table 7 (Chinese). Both CoNLL-2000 and Chinese base-chunking tasks agree that the "one-versus-one" (OVO) achieves the best system performance in terms of accuracy, and OVO, C-OVA, DAG performs marginally worse. In addition, our HC-OVA performs slightly worse than these methods. It is not surprising that dense-ECOC achieves the worst result than the above methods since it encodes multiclass with "dense" Boolean codes. In basic, the ECOC was designed to correct errors with additional codes, while the dense-ECOC does not provide any additional bit to be corrected and thus affects the chunking accuracy. Nevertheless the use of longer codes does increase both training and testing time costs. Besides, how to select a better strategy to encode the multiclass is an NP-hard problem. One can adopt the OVO/OVA-based encoding method for ECOC (as reported by Ref. [26]), but essentially, it was a variant OVO/OVA prototype.

In terms of time efficiency, it is clear that HC-OVA is the most efficient model in both training and testing on the two phrase chunking tasks. The reports here include the routine processing time (disk I/O and preprocessing (like feature mapping)), and SVM classification time. The overall turn around time is the sum of the above two factors. The observed results also support the analysis as in Tables 4 and 5 that orders the multiclass models in terms of time efficiency. However, in both experiments the HC-OVA were faster than the other methods. The situation is more salient when the number of chunk class increases. In the CoNLL-2000 chunking task the chunking speed of OVO is 835 term/s, while it decreases to < 100 term/s in the Chinese base-chunking task since the number of chunk class scale from 23 to 47.

Table 8
Statistical significance test on the benchmark corpus

Method A	Method B	CoNLL-2000		Chinese base-chunking	
		Probability-test	McNemar-test	Probability-test	McNemar-test
OVO	DAG	$\sim$	$\sim$	$\sim$	$\sim$
OVO	OVA	$\sim$	$\sim$	$\sim$	$\sim$
OVO	C-OVA	$\sim$	$\sim$	$\sim$	$\sim$
OVO	HC-OVA	$\sim$	$\sim$	$\sim$	$\sim$
OVO	Tree-based	$\sim$	$\sim$	$\sim$	$\sim$
OVO	Dense-ECOC	$>$	$\gg$	$\gg$	$\gg$
DAG	HC-OVA	$\sim$	$\sim$	$\sim$	$\sim$
C-OVA	HC-OVA	$\sim$	$>$	$\sim$	$\sim$
C-OVA	Tree-based	$\sim$	$>$	$\sim$	$\sim$
HC-OVA	Tree-based	$\sim$	$\sim$	$\sim$	$\sim$
HC-OVA	Dense-ECOC	$>$	$\gg$	$\gg$	$\gg$

"$\gg$" means P-value < 0.01; "$>$" means $0.01 < P$-value < 0.05; "$\sim$" means P-value > 0.05.

In principal, the testing time complexity of DAG should be equal or a little bit more efficient than the OVA. The experimental results of the Chinese base-chunking task did conflict it. The main reason is that the theoretical analysis ignores the practical system and model scalability problems. In practice, if the class number or dimensionality tends to be huge, for example Chinese base-chunking task, it largely increases the memory loading. We had performed several trials with smaller dataset and found the theoretical proof only holds when the practical system can normally handle the SVs. However, for example, in the Chinese chunking task, both dimension and class number scales where the quadratic created pairwise classifiers made the system difficult to handle such a huge and high-dimensional SVs efficiently as OVA. We have no choice but to select an inefficient implementation for OVO and DAG in this case.

On the other hand, we adopt the statistical tests to evaluate the performance difference among these methods. We first employed the proportion-test [42], and McNemar-test [42] to evaluate the system output tests on the standard benchmark corpora. Second, the one-way ANOVA evaluation metrics is adopted to see the difference via 10-fold cross-validation. Table 8 lists the proportion-test and McNemar-test results on the two chunking tasks. The two statistical tests of the two tasks mostly agree that there is no significant difference between our methods (C-OVA and HC-OVA) and the other top-performed multiclass SVM models, especially OVO, whereas the dense-ECOC is quite different from C-OVA and HC-OVA at least under 99% confidence values.

Table 9 summarizes the average recall, precision, and $F_{(\beta)}$ rates for different multiclass SVM methods by doing 10-fold cross-validation within the standard CoNLL-2000 training data and the adopted Chinese base chunking training set. By means of the 10-fold cross-validation, Table 10 lists the significant test results using one-way ANOVA. Instead of comparing the system output, the one-way ANOVA merely test the system performance according to the $F_{(\beta)}$ scores of the 10 trials. In this test, we found that the HC-OVA are significantly different from the other methods except for the dense-ECOC and tree-based methods, while the one-way ANOVA also showed that there is no significant difference between C-OVA and the OVO.

Table 9
The average scores in 10-fold cross-validation on the two chunking tasks

Method	CoNLL-2000			Chinese base-chunking		
	Recall	Precision	$F_{(\beta)}$	Recall	Precision	$F_{(\beta)}$
OVA	94.39	94.51	94.45	93.50	94.80	94.14
OVO	94.46	94.60	94.53	93.57	94.82	94.19
DAG	94.41	94.58	94.49	93.58	94.82	94.20
Dense-ECOC	92.10	89.82	90.95	92.59	91.49	92.03
Tree-based	94.23	94.31	94.26	93.49	94.78	94.13
C-OVA	94.38	94.52	94.45	93.50	94.80	94.15
HC-OVA	94.19	94.25	94.22	93.37	94.56	93.96

Table 10
One-way ANOVA tests on the two chunking tasks using the 10 fold cross-validation results

Method A	Method B	CoNLL-2000	Chinese base-chunking
OVO	DAG	$\sim$	$\sim$
OVO	OVA	$\sim$	$\sim$
OVO	C-OVA	$\sim$	$\sim$
OVO	HC-OVA	$\gg$	$\gg$
OVO	Tree-based	$\gg$	$\sim$
OVO	Dense-ECOC	$\gg$	$\gg$
DAG	HC-OVA	$\gg$	$\gg$
C-OVA	HC-OVA	$\gg$	$\gg$
C-OVA	Tree-based	$\gg$	$\sim$
Tree-based	HC-OVA	$\sim$	$>$
HC-OVA	Dense-ECOC	$\gg$	$\gg$

"$\gg$" means P-value < 0.01; "$>$" means $0.01 < P$-value < 0.05; "$\sim$ means P-value > 0.05.

Fig. 6 illustrates the learning curves of the above top-performed multiclass SVM models (other than dense-ECOC) on the two chunking tasks. The learning curves implied the relationship between the amount of annotated corpus and the system performance. The right-hand side of Fig. 6, we can see that HC-OVA is not very effective on the Chinese base-chunking task when there is only a very small amount of training data, while C-OVA performs very satisfactory accuracy as well as OVO.

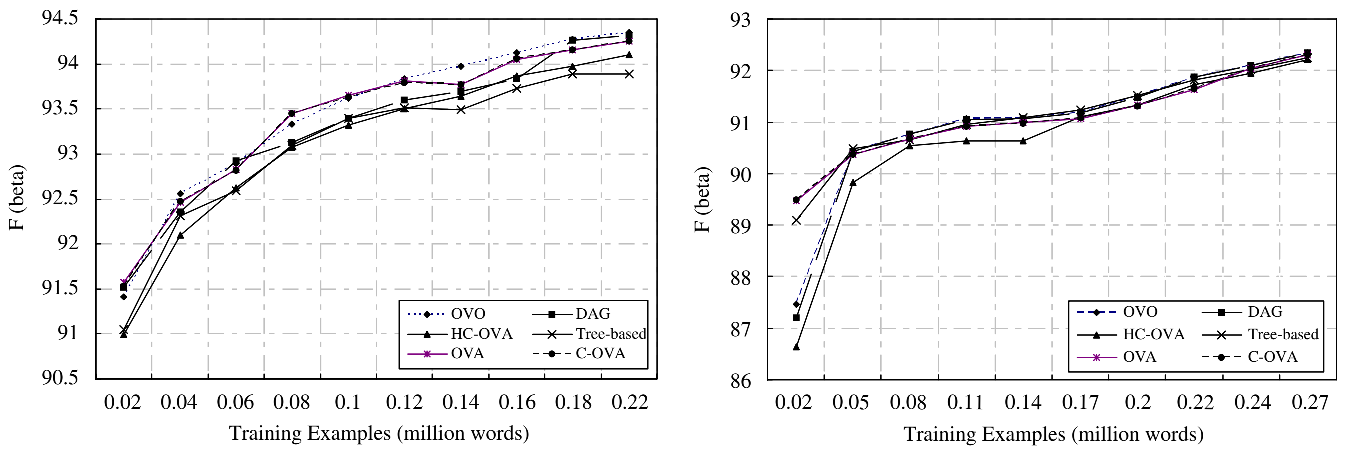

Fig. 6. Learning curve of different multiclass SVM models in CoNLL-2000 (left) and Chinese base-chunking (right) tasks.

Table 11
Comparisons of chunking performance for CoNLL-2000 task

Chunking system	Recall	Precision	$F_{(\beta)}$
Semi-supervised structure learning[a] [19]	94.20	94.57	94.39
C-OVA	**94.18**	**94.31**	**94.25**
SVM [12]	94.12	94.13	94.12
HC-OVA	**94.05**	**94.15**	**94.10**
Voted-SVMs [10]	93.89	93.92	93.91
Voted-perceptrons [15]	93.38	94.20	93.79
Structural learning [19]	93.37	93.83	93.60
Generalized winnow [16]	93.60	93.54	93.57
Voted-memory-based methods [18]	91.00	94.04	92.50
Specialized HMM [17]	92.41	91.96	92.19

[a]The semi-supervised learning, which boosted thousands of the learners with large unlabeled data.

Table 12
Chunking results for the CoNLL-2000 phrase chunking task

Phrase type	C-OVA			Phrase type	HC-OVA		
	Recall	Precision	$F_{(\beta)}$		Recall	Precision	$F_{(\beta)}$
ADJP	75.80	83.00	79.24	ADJP	75.11	81.23	78.05
ADVP	81.76	84.59	83.15	ADVP	81.18	82.22	81.70
CONJP	55.56	50.00	52.63	CONJP	66.67	50.00	57.14
INTJP	100.00	100.00	100.00	INTJP	100.00	40.00	57.14
LST	0.00	0.00	0.00	LST	0.00	0.00	0.00
NP	94.52	94.82	94.67	NP	94.36	94.74	94.55
PP	98.30	96.71	97.50	PP	98.25	96.98	97.61
PRT	78.30	76.15	77.21	PRT	75.47	71.43	73.39
SBAR	87.10	88.59	87.84	SBAR	86.73	89.06	87.88
UCP	94.44	94.32	94.38	UCP	0.00	0.00	0.00
VP	75.80	94.31	94.25	VP	94.44	94.26	94.35
All	94.18	83.00	79.24	**All**	94.05	94.15	94.10

As shown in above experiments, we conclude that OVO achieves the best chunking performance but slow in testing. In contrast, the HC-OVA is the most efficient multiclass SVM model, which substantially is faster than conventional OVO without a discernible change in accuracy.

4.4. Comparisons

We have shown the best chunking performance is obtained by OVO, while HC-OVA achieved the best training and testing time efficiency. C-OVA had no significant difference to OVO model. Now, we select the two multiclass SVM models (C-OVA and HC-OVA) to be compared with the published studies. It is worth to note that the use of external knowledge, like parsers or additional corpora is disallowed since it is not a fair comparison. Furthermore, it is quite difficult to perform the statistical tests, most system outputs were not easily available. Nevertheless all of them were evaluated in the same benchmark, it still explicitly reflects the actual chunking performance. Table 11 lists the comparison results of the CoNLL-2000 chunking task, and Table 12 summarizes the overall chunking accuracies of HC-OVA and C-OVA on each phrase type.

In the CoNLL-2000 chunking task, the C-OVA method attends the best system performance, while HC-OVA achieves a very competitive $F_{(\beta)}$ score (Rank 3). We ignore the comparison of Wu's work [12] here, since we had included his OVA model. For the next best systems: voted-SVMs [10], and voted-perceptrons [15] they employed the polynomial kernels instead of the linear kernel. As discussed in Section 2.1, the use of polynomial kernel enormously increases both training and testing costs. In contrast, our methods are not only more efficient but also more accurate than the two. For example, regardless of training time cost, even the OVO only can handle 835 terms per second, it still substantially faster than polynomial kernels that just chunk 20–30 words per second [10]. On the contrary, the structural learning [19] and Winnow-based methods [16] were presented the linear model-based methods and achieved slightly worse accuracy than the polynomial kernel-based approaches.

Recently, the semi-supervised structural learning (multitask learning) [19] showed the state-of-the-art performance (94.39) via boosting classifiers with large unlabeled dataset. However, the multitask prototype aims to integrate tens of thousand classifiers (top-1000 learners were selected by Ref. [19]). Although the use of redundant multitask learners slightly outperform our

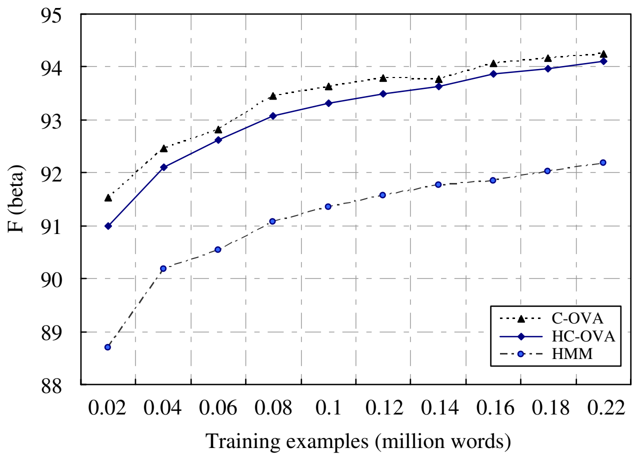

Fig. 7. A comparative learning curve of specialized HMM, C-OVA, and HC-OVA for CoNLL-2000 chunking task.

C-OVA (94.25 versus 94.35), in terms of efficiency, it is far slower than C-OVA. Each individual learner (single task) in the multitask learning framework adopted the similar OVA prototype. By assuming the same testing time of the single task learner as OVA, the use of 1000 classifiers is 1000 times slower than one OVA. In other words, in the CoNLL-2000 chunking task, the *full chunking time* of the multitask learning is at least 4000 s (performs full OVA with 1000 times). On the contrary, our HC-OVA and C-OVA required less than 2.86 and 3.4 s and achieved very comparable performance.

Now we turn to compare the system training and testing time costs. It is difficult to explicitly compare the actual time costs with previous studies. Most studies do not report their actual training and testing times in their works. The specialized HMM [17] was known as its efficient training and testing. It only spends few seconds for training while handling more than 40 000 terms per second. Another efficient method is the voted-memory-based system [18] that adopted the information gain tree (IG tree). The height of IG-tree is equal to the number of attributes (features). Its testing process is to visit the tree. Although the actual training and testing times were not found in Tjong Kim Sang's study, it is clear that the time costs of the IG-tree is much more efficient than SVM. However, these efficient methods do not show high performance (see Table 11). We replicate the specialized HMM models [17] and report the actual learning curve of the specialized HMM to compare with C-OVA and HC-OVA. Fig. 7 peaks the chunking performance of different training size for CoNLL-2000 chunking task. Compared to Fig. 6, it is clear that the need of training examples of HC-OVA is much less than the specialized HMM.

On the other hand, even the voted-SVM and voted-perceptrons seem to be high performance, they were not efficient. As described above, to train "one" polynomial kernel SVM with OVO type requires one day [10] and subsequently obtain a slow chunker (20–30 terms/s). In Kudoh's work, they further combined eight trained chunkers to improve the performance. On the contrary, the linear models, like structural learning [19], and Winnow-based [16] seem to be much faster. The two methods can be viewed as a variant OVA multiclass SVM chunking model as ours.[5] But they are not as accurate as the polynomial kernel methods.

Table 13 summarizes the experimental results of the Chinese base-chunking task. In this task, the settings of our chunker are the same as previous chunking tasks. It is worth to note that the affix feature in Chinese base-chunking task is not explicit. Thus, we use the atomic Chinese character to represent the affix features. Although, several Chinese phrase chunking systems have been proposed in recent years, such as HMM-based [43]. It is difficult to compare with these systems because they were performed in different dataset. Thus, we only report the actual results of the proposed chunking model for Chinese base-chunking task.

Consequently we found that the OVO achieves the state-of-the-art chunking performance at one hand. But the main limitation is the slow testing time, in particular when the number of phrase type scales to high. On the other hand, the HMM-based method is quite efficient, but it is not the state-of-the-art. In addition, our HC-OVA held a better trade-off between time cost and accuracy. Its chunking speed largely outperforms the other multiclass SVM models which can handle 16 000 terms per second while keeping the chunking performance as high (rank 3: $F_{(\beta)} = 94.10$). In addition, the statistical significant tests also agree that there is no significant difference between HC-OVA, and OVO on the benchmark corpus.

5. Time cost analysis and discussions

5.1. Estimating training and testing time costs

Recall that we could not provide the direct proof of the testing time cost of HC-OVA and unbalanced tree-based methods. The two approaches deeply depend on the actual data distribution and SVM performance. In this section we will present the detailed analysis of the actual training and testing time costs on the CoNLL-2000 chunking task. The analysis is also replicable to Chinese and other similar tasks. At the beginning, we summarize the data percentages of each chunk class which was encoded with IOE2 style in Table 14. Again we assume the training time complexity of an SVM is O(N) where N is the number of training examples. Using the similar way to compute HC-OVA's complexity, the actual training time cost can be derived as follows:

$$\begin{aligned}&(\mathrm{All}) + (\mathrm{All} - \{\mathrm{I-NP}\}) + (\mathrm{All} - \{\mathrm{I-NP}, \mathrm{E-NP}\}) + (\mathrm{All}\\ &\quad - \{\mathrm{I-NP}, \mathrm{E-NP}, \mathrm{O}\}) + \cdots + (\{\mathrm{E-UCP}, \mathrm{I-PRT}\})\\ &\qquad = N + (1-0.299)N + (1-0.299-0.26)N\\ &\qquad\quad + (1-0.299-0.26-0.13)N + \cdots + 0.0018N\\ &\qquad \cong 2.89N\end{aligned}$$

In CoNLL-2000 chunking training data there are about 0.2 million words. Thus, we can replace N by the actual number

[5] The two methods also employed the dynamic programming technique to re-rank the chunking result. On the contrary, we perform the deterministic strategy.

Table 13
Chunking results for the Chinese base-chunking task

Phrase type	$F_{(\beta)}$ (C-OVA)	$F_{(\beta)}$ (HC-OVA)	Phrase type	$F_{(\beta)}$ (C-OVA)	$F_{(\beta)}$ (HC-OVA)
ADJP	98.56	98.40	PP	0.00	0.00
ADVP	99.46	99.45	PRN	0.00	0.00
CLP	99.77	99.77	QP	97.96	97.48
CP	0.00	0.00	VCD	55.79	55.45
DNP	0.00	0.00	VCP	93.33	87.50
DP	99.40	99.40	VNV	33.33	46.15
DVP	0.00	0.00	VP	92.93	93.04
FRAG	97.68	96.72	VPT	68.97	55.17
LCP	0.00	0.00	VRD	87.80	89.08
LST	90.20	88.46	VSB	21.78	25.21
NP	89.15	89.11	**All**	92.31	92.22

Table 14
Percentages of words per chunk class in the training and testing data

CoNLL-2000	E-ADJP	E-ADVP	E-CONJP	E-INTJ	E-LST	E-NP	E-PP	E-PRT	E-SBAR	E-UCP	E-VP	
Training	0.97	1.99	0.02	0.01	0.004	26.01	10.05	0.26	1.04	0.0009	10.13	
Testing	0.92	1.82	0.01	0.004	0.01	26.21	10.15	0.22	1.12	~ 0	9.83	
	I-ADJP	I-ADVP	I-CONJP	I-INTJ	I-LST	I-NP	I-PP	I-PRT	I-SBAR	I-UCP	I-VP	O
Training	0.30	0.20	0.03	0.004	0.00	29.90	0.13	0.0009	0.03	0.002	5.66	13.17
Testing	0.35	0.18	0.027	~ 0	0.004	30.34	0.10	~ 0	0.008	~ 0	5.58	13.04

Table 15
Training time costs for different multiclass SVM models

CoNLL-2000	OVA	OVO	DAG	Dense-ECOC	Tree-based	C-OVA	HC-OVA
Training cost	$22N$	$21N$	$21N$	$5N$	$11.28N$	$12.86N$	$2.89N$

Table 16
Testing time costs for different multiclass SVM models

CoNLL-2000	OVA	OVO	DAG	Dense-ECOC	Tree-based	C-OVA	HC-OVA
Expected testing cost (using empirical distribution of training data)	22	231	21	$5+(22\log 22)$[a]	11.26	13	2.89
Expected testing cost (using empirical distribution of testing data)	22	231	21	$5+(22\log 22)$[a]	10.97	13	2.85
Actual testing cost	22	231	21	$5+(22\log 22)$[a]	11.29	10.82	2.34
SVM classification time (47 377 words) (s)	2.09	54.34	4.73	1.53	1.82	1.01	**0.48**

[a]The dense-ECOC approach requires five times for SVM classification and 22 log 22 times for bit decoding.

of training examples. Note that the tree-based method relies on the tree construction. The actual constructed binary tree can be shown in Appendix B. By means of similar derivations, we observed that the training time of the tree-based method is about $11.28N$. We also analyze the other multiclass SVM models. The detail training time costs can be found in Table 15.

To evaluate the testing cost, we calculate the number of times performing SVM classification. Here we adopt the expected tree height to reflect the testing cost based on empirically accounting the training and testing data. The expected tree height is defined as follows:

$$ExpectedTreeHeight = \sum_{i=1}^{C} P(Chi) \times Height(Chi) \quad (6)$$

where $P(Ch_i)$ denotes as the probability of chunk class Ch_i in the dataset and height(Ch_i) is the tree level of the Ch_i. In HC-OVA, the tree is a skewed binary tree which is made by ordering the probability of each chunk class. Thus, we can derive the actual testing time cost of HC-OVA as follows:

$$P(\mathrm{I-NP}) * Height(\mathrm{I-NP}) + P(\mathrm{E-NP}) * Height(\mathrm{E-NP}) + \cdots + P(\mathrm{I-UCP}) * Height(\mathrm{I-UCP}) = 0.299 * 1 + 0.260 * 2 + \cdots + 0.0009 * 21 \cong 2.89$$

Alternatively, for the tree-based method we can use the similar way to estimate its testing cost, i.e., 11.26. We therefore generalize the above computation to the other multiclass SVM models. Table 16 lists the testing time cost of different multiclass

SVM models. The "actual testing cost" means the number of time performing SVM classification during testing phase.

There is a problem to estimate $P(Ch_i)$, which can be observed via the empirical probability in either training or testing data. However, in real case the actual probability is not available. Usually we assume that the probability distribution of training and testing should be consistent. If they were dissimilar with each other, the annotated data is not representative and cannot reflect the actual empirical distribution. Besides, it also decreases the chunking performance. For example, if the "NP(noun phrase)" class received no training data, the misclassification intuitively occurs when NPs appear in the testing data.

We can use the weighted Jensen–Shannon-divergence (JS-divergence) [44] to evaluate the difference between the training (P) and testing (Q) data distributions. The weighted JS-divergence is defined as follows:

$$Weighted\_JS(P\|Q) = \frac{P}{P+Q}KL(P\|Q) + \frac{Q}{P+Q}KL(Q\|P) \quad (7)$$

$$KL(P\|Q) = \sum_{x \in chunk\_class} p(x)\log\left(\frac{p(x)}{q(x)}\right) \quad (8)$$

where $p(x)$ and $q(x)$ are the probability of chunk class x in training and testing data, respectively. Using Table 14, the distributional difference between the training and testing data can be computed with (7). By means of the JS-divergence estimation, we find that the probability distance of the two distributions is quite similar (the value is less than 0.0005). In other words, the empirical analysis of the testing cost using the training data is very close to the testing data.

5.2. Discussion

As shown in Tables 15 and 16, the HC-OVA achieved the best time costs in terms of training and testing. The above two estimations are performed based on the original time complexity analysis and actual data distribution. If we do know the data are whether balanced or not, the above training cost computation can first be analyzed before training the SVM. By analyzing, we can find that the main limitation of HC-OVA is that it assumes the data are unbalanced distributed. If the data does not obey this assumption, the time cost will be higher than the dense-ECOC and tree-based methods. However, it is often the case that the data are unbalanced distributed. For example, in the CoNLL-2003 named entity recognition task [45] the most frequent word is the non-entity term that dominates more than half of the training set, and the multilingual dependency parsing (about 13-languages treebanks) [46], data are also unbalanced distributed. Therefore by theoretical and experimental demonstration, we conclude that the HC-OVA is not only efficient but also kept the SVM robust.

Although the tree-based is another efficient method, it was shown that the performance is not as effective as the OVO and OVA. Even we use the state-of-the-art clustering technique to establish the binary tree. The key is still the tree construction. A good binary tree might lead to better result. But it is difficult to obtain the optimum tree for multiclass SVM. Similar to ECOC approaches, finding the optimum tree (or optimum encoding strategy) is an NP-hard problem. Nevertheless the bottom-up clustering is not a good choice to build the tree. In our case, the algorithm should be performed until the *complete clustering tree* is built. But the time complexity of bottom-up clustering is $O((DC)^3)$ where D is the average active number of dimensions. In our case, the average active datum per vector is about 50.12 and 39.21 for CoNLL-2000 and Chinese chunking tasks where the highest dimension is more than 0.85 million. Especially when the number of chunk class scales to hundred of thousands such as text IE, and dependency parsing, the cubic clustering time is much more intractable than SVM training.

In addition, in terms of testing time, both OVO and DAG suffer from the memory loading problems when the number of SVs exponentially scales. The situation is even worse to the other nonlinear kernel types for SVM. We can use a weight vector to represent the linear combined SVs, while the nonlinear kernels could not. For the OVO and DAG models, the critical key-point is that they construct pairwise classifiers, which cause the quadratic scaled SVs. Imagine the case: there are 30 training examples normally distributed in 10 categories. OVO/DAG method should build 45 OVO classifiers where each of which supposedly generates four SVs. Totally, they produce 180 SVs whereas there are only 30 training instances in this example. In our implementations, we cannot efficiently and exhaustively handle such a high dimension and large category set for Chinese chunking task where there are $47 * 46/2 = 1081$ pairwise classifiers with 0.85 million dimensions. Thus, we employ the inefficient data structures, i.e., tree-based linked list to perform OVO and DAG classification. Due to the scalability problem, the DAG performed surprisingly slower than OVA method in this task. This finding does disobey the theoretical time complexity analysis where the DAG should be slightly faster than OVA. The main reason is that the scaled categories and SVs which make it difficult to handle the classification task with efficient data structures as OVA. To efficiently classify with such a large-scale category and high-dimensional problems with OVO and DAG, a better implementation technique should be newly designed. We left the improvement as the future work since in basic it is necessary to compare with the quadratic scaled classifiers.

6. Conclusions

SVM-based phrase chunking had shown to be robust for phrase pattern recognition problems. But its inefficiency limits actual use to handle large amount of documents efficiently. This paper presents two methods (C-OVA and HC-OVA) that make the SVM substantially faster. The two methods firstly construct the consistent matrix to reduce the unnecessary SVM classification and HC-OVA further speed-up training and testing through a constraint binary tree. By the theoretical computational time complexity analyzing, HC-OVA, dense-ECOC, and tree-based methods can be efficiently trained and tested when the data are unbalanced distributed. The experimental results also show that our HC-OVA is the most efficient multiclass SVM model in terms of training and testing for chunking tasks. In particular, the testing speed of HC-OVA is at least 100 times faster than the state-of-the-art OVO model (OVO-based SVM) while resulting a marginal decrease in accuracy. In addition, we also found that the OVO and DAG are inefficient in testing due to the quadratic scaled pairwise classifiers, which make the practical system inefficient to handle large-scale classification. The situation is even worse when using other nonlinear kernels such as polynomial kernel. Although the dense-ECOC model was shown to be the second fast method, it was quite far away from the other multiclass SVM models in terms of accuracy. On the other hand, even the tree-based was another efficient models, the cubic clustering time is needed.

The proposed methods can be also applied to other similar tasks such as bottom-up chunking to parsing, and named entity recognition where a named entity can be treated as a phrase chunk. It is quite often that novel question answering systems requires a fast and accurate named entity recognizer to handle thousands of retrieved documents in a very short time. The proposed methods can be found at (http://140.115.112.118/bcbb/Chunking.htm) for online demonstration purpose.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at 10.1016/j.patcog.2008.02.010.

References

[1] S. Abney, Parsing by chunks, Principle-Based Parsing (1991) 257–278.
[2] M. Surdeanu, S. Harabagiu, J. Williams, P. Aarseth, Using predicate-argument structures for information extraction, in: Proceedings of 41st Annual Meetings of the Association for Computational Linguistics, 2003, pp. 8–15.
[3] S.B. Park, B.T. Zhang, Co-trained support vector machines for large scale unstructured document classification using unlabeled data and syntactic information, Inf. Process. Manage. 40 (2004) 421–439.
[4] R. Florian, A. Ittycheriah, H. Jing, T. Zhang, Named entity recognition through classifier combination, in: Proceedings of Conference on Natural Language Learning, 2003, pp. 168–171.
[5] P. Koehn, K. Knight, Feature-rich statistical translation of noun phrases, in: Proceedings of 41st Annual Meetings of the Association for Computational Linguistics, 2003, pp. 311–318.
[6] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J.H. Martin, D. Jurafsky, Support vector learning for semantic argument classification, Mach. Learn. 60 (2004) 11–39.
[7] D. Gildea, M. Palmer, The necessity of parsing for predicate argument recognition, in: Proceedings of 40th Annual Meetings of the Association for Computational Linguistics, 2002, pp. 239–246.
[8] A. Ratnaparkhi, A Linear observed time statistical parser based on maximum entropy models, in: Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, 1997, pp. 1–10.
[9] E.F. Tjong Kim Sang, Transforming a chunker to a parser, Computational Linguistics in the Netherlands (2000) 177–188.
[10] T. Kudoh, Y. Matsumoto, Chunking with support vector machines, in: Proceedings of 2nd Meetings of the North American Chapter and the Association for the Computational Linguistics, 2001, pp. 192–199.
[11] Y.S. Lee, Y.C. Wu, A robust multilingual portable phrase chunking system, Expert Syst. Appl. 33 (3) (2007) 1–26.
[12] Y.C. Wu, C.H. Chang, Y.S. Lee, A general and multi-lingual phrase chunking model based on masking method, in: Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science (LNCS), vol. 3878, 2006, pp. 144–155.
[13] Y.C. Wu, T.K. Fan, Y.S. Lee, S.J. Yen, Extracting named entities using support vector machines, in: Knowledge Discovery in Life Science Literature, Lecture Notes in Bioinformatics (LNBI), vol. 3886, 2006, pp. 91–103.
[14] Y.C. Wu, J.C. Yang, Y.S. Lee, S.J. Yen, Efficient and robust phrase chunking using support vector machines, in: Information Retrieval Technology, Lecture Notes in Computer Science (LNCS), vol. 4182, 2006, pp. 350–361.
[15] X. Carreras, L. Marquez, J. Castro, Filtering-ranking perceptron learning for partial parsing, Mach. Learn. 59 (2005) 1–31.
[16] T. Zhang, F. Damerau, D. Johnson, Text chunking based on a generalization winnow, J. Mach. Learn. Res. 2 (2002) 615–637.
[17] A. Molina, F. Pla, Shallow parsing using specialized HMMs, J. Mach. Learn. Res. 2 (2002) 595–613.
[18] E.F. Tjong Kim Sang, Memory-based shallow parsing, J. Mach. Learn. Res. 2 (2002) 559–594.
[19] R.K. Ando, T. Zhang, A high-performance semi-supervised learning method for text chunking, in: Proceedings of 43rd Annual Meetings of the Association for Computational Linguistics, 2005, pp. 1–9.
[20] C.L. Goh, M. Asahara, Y. Matsumoto, Chinese word segmentation by classification of characters, Int. J. Comput. Linguist. Chin. Lang. Process. 10 (3) (2005) 381–396.
[21] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[22] J. Giménez, L. Márquez, Fast and accurate part-of-speech tagging: the SVM approach revisited, in: Proceedings of the International Conference on Recent Advances in Natural Language Processing, 2003, pp. 158–165.
[23] R. Rifkin, A. Klautau, In defense of One-vs-all classification, J. Mach. Learning Res. 5 (2004) 101–141.
[24] U. Kreßel, Pairwise classification and support vector machines, Advances in kernel methods: support vector learning (1999) 255–268.
[25] J.C. Platt, N. Cristianini, J. Shawe-Taylor, Large margin dags for multiclass classification, Adv. Neural Inf. Process. Syst. 12 (2000) 547–553.
[26] A. Klautau, N. Jevtic, A. Orlitsky, On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines, J. Mach. Learn. Res. 4 (2003) 1–15.
[27] T.K. Huang, R.C. Weng, C.J. Lin, Generalized bradley-terry models and multi-class probability estimates, J. Mach. Learn. Res. 7 (2006) 85–115.
[28] F. Takahashi, S. Abe, Decision-tree-based multicalss support vector machines, in: Proceedings of the 9th International Conference on Neural Information Processing, vol. 2, 2002, pp. 1418–1422.
[29] W. Sun, J. Chen, A multi-class classifier based on SVM decision tree, In Forum of IEEE Comput. Intell. Soc. Electronic Letter, 2006.
[30] V. Vural, J.D. Dy, A hierarchical method for multi-class support vector machines, in: Proceedings of the 21st International Conference on Machine Learning, 2004, pp. 831–838.
[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., McGraw-Hill Higher Education, 2002.
[32] K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines, J. Mach. Learn. Res. 2 (2001) 265–292.
[33] T. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large margin methods for structured and interdependent output variables, J. Mach. Learn. Res. 6 (2005) 1453–1484.
[34] L.A. Ramshaw, M.P. Marcus, Text chunking using transformation-based learning, in: Proceedings of the 3rd Workshop on Very Large Corpora, 1995. pp. 82–94.
[35] E.F. Tjong Kim Sang, J. Veenstra, Representing text chunks, in: Proceedings of 9th Conference of the European Chapter of the Association for Computational Linguistics, 1999, pp. 173–179.
[36] S.S. Keerthi, O. Chapelle, D. DeCoste, Building support vector machines with reduced classifier complexity, J. Mach. Learn. Res. 7 (2006) 1493–1515.

[37] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, 2006, pp. 217–226.
[38] I.W. Tsang, J.T. Kwok, P.M. Cheung, Core vector machines: fast SVM training on very large data sets, J. Mach. Learn. Res. 6 (2005) 363–392.
[39] E.F. Tjong Kim Sang, S. Buchholz, Introduction to the CoNLL-2000 shared task: chunking, in: Proceedings of 4th Conference on Natural Language Learning, 2000, pp. 127–132.
[40] E. Brill, Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging, Comput. Linguist. 21 (4) (1995) 543–565.
[41] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of 10th European Conference on Machine Learning, 1998, pp. 137–142.
[42] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (1998) 1895–1923.
[43] H. Li, J.J. Webster, C. Kit, T. Yao, Transductive HMM based Chinese text chunking, in: Proceedings of Natural Language Processing and Knowledge Engineering, 2003, pp. 257–262.
[44] I.S. Dhillon, S. Mallela, R. Kumar, A divisive information-theoretic feature clustering algorithm for text classification, J. Mach. Learn. Res. 3 (2003) 1265–1287.
[45] E.F. Tjong Kim Sang, F.D. Meulder, Introduction to the CoNLL-2003 shared task: language-independent named entity recognition, in: Proceedings of 7th Conference on Natural Language Learning, 2003, pp. 142–147.
[46] S. Buchholz, E. Marsi, A. Dubey, Y. Krymolowski, CoNLL-X shared task on multilingual dependency parsing, in: Proceedings of 10th Conference on Natural Language Learning, 2006, pp. 149–164.

About the Author—YU-CHIEH WU received the M.S. degree in the Department of Information Management in 2003 from Ming Chuan University, Taiwan. He is currently a Ph.D. candidate of Department of Computer Science and Information Engineering in National Central University. He is also the student members of ACL and IEICE. His research interests include natural language processing, machine learning and video information systems.

About the Author—YUE-SHI LEE received the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1997. He is currently an associate professor in Department of Computer Science and Information Engineering, Ming Chuan University, Taoyuan, Taiwan. His initial research interests were computational linguistics and Chinese language processing, and over time he evolved toward data warehousing, data mining, information retrieval and extraction and Internet technology. He is a member of the IEEE Computer Society.

About the Author—JIE-CHI YANG received the Ph.D. degree in Human System Science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. He is currently an associate professor in Graduate Institute of Network Learning Technology, National Central University, Taiwan. His research interests include natural language processing, mobile learning, web-based learning environment and digital game-based learning.