In this paper, we propose a method for text-query-based keyword spotting from online Chinese handwritten documents using character classification model. The similarity between the query word and handwriting is obtained by combining the character classification scores. The classifier is trained by one-versus-all strategy so that it gives high similarity to the target class and low scores to the others. Using character classification-based word similarity also helps overcome the out-of-vocabulary (OOV) problem. We use a character-synchronous dynamic search algorithm to efficiently spot the query word in large database. The retrieval performance is further improved by using competing character confusion and writer-adaptive thresholds. Our experimental results on a large handwriting database CASIA-OLHWDB justify the superiority of one-versus-all trained classifiers and the benefits of confidence transformation, character confusion and adaptive thresholds. Particularly, a one-versus-all trained prototype classifier performs as well as a linear support vector machine (SVM) classifier, but consumes much less storage of index file. The experimental comparison with keyword spotting based on handwritten text recognition also demonstrates the effectiveness of the proposed method.

Keywords: Online Chinese handwritten documents; keyword spotting; one-versus-all classifiers; character-synchronous dynamic search; character confusion; adaptive threshold.
1. Introduction

With the increasing use of digitizing tablets, tablet PCs, digital pens and touch-based smart phones, online handwritten documents are produced constantly. This entails efficient retrieval techniques to exploit the semantic information in the documents, among which keyword spotting is widely adopted. For document retrieval from large database, it is necessary to build an index containing multiple candidate recognition results so as to overcome the recognition error. According to the indexing technique, handwritten document retrieval methods can be categorized into two groups: indexing by character recognition (transcription) and lexicon-driven indexing. Transcription-based text search relies on the character recognition accuracy. To guarantee the high recall rate of retrieval in cases of low recognition accuracy (e.g. on handwritten documents), transcription-based methods need to store a large number of candidate characters (generated by the recognizers). In lexicon-driven (or word-model-based) indexing, a word recognizer is used to generate multiple word candidates for the words contained in a pre-compiled lexicon, or the document is matched with all the word models for the words in the lexicon. However, when the query word is out-of-vocabulary (OOV), it cannot be spotted by the lexicon-driven method. The OOV problem can be partially solved by sub-word (such as character and letter) model matching.

With regard to keyword spotting without transcription, according to the word similarity scoring technique, the methods can be categorized into two groups: image-to-image and model-based scoring. Both can be applied to text (keyboard) query and handwriting query. The image-to-image matching technique is based on a distance measure between the query (input image or an image synthesized from the text query) and all candidate word images. Without model training, it suffers from low matching accuracy. In contrast, the word-model-based method can be used for retrieving multi-writer or writer independent documents in large database, by storing similarity scores computed by a trained character/word model in an index file. For fast retrieval, the index can be built on the results of word level recognition in lexicon-driven mode, in which the OOV problem can be partially solved by sub-word or character-level models.

This paper is concerned with keyword spotting from online Chinese handwritten documents using text query. In addition to the ambiguity of character segmentation and the shape variation, Chinese handwritten documents suffer from the large alphabet (over 5000 characters and 200,000 words are daily used) and the difficulty of word segmentation (there is no extra space between words). Most previous works of Chinese document retrieval have focused on printed documents. A few works have been performed on word retrieval from online Chinese handwriting and character retrieval from calligraphic Chinese archives using approximate shape matching.

In this paper, we propose a character-model-based method for text-query-based keyword spotting, aiming to retrieve keywords from large databases of multi-writer or writer independent documents. Because of the huge number of words in Chinese
language, it is inappropriate and inefficient to build word-level index. Hence, we instead train a character classifier for computing character similarities, particularly a one-versus-all (OVA) trained prototype classifier to cope with the large alphabet of Chinese. The OVA trained classifier has the merit that it gives high scores to target characters and low scores to nontarget characters (including mis-segmented characters). The word similarity is then obtained by combining character similarity scores. To overcome the difficulties of character and word segmentation, we search the query word from character candidate lattice generated by over-segmentation, and use a character-synchronous dynamic search algorithm for fast and efficient retrieval.

Text search using character segmentation and recognition candidates has been proven effective in Japanese document retrieval. These methods are similar to ours, but we have paid more attention to the training of character classifier for a better similarity measure, which has been shown to be an important issue. In this work, we show the superiority of the OVA trained prototype classifier over the linear support vector machine (SVM) and a prototype classifier trained with conventional multi-class objective. Transforming the classifier output to probabilistic confidence is shown to benefit the keyword spotting performance. We also use character confusion information and writer-adaptive thresholds to further improve the retrieval performance. As we know, this is the first time confidence transformation, character confusion, and writer-adaptive thresholds are used in this combination. This paper is an extension to our previous conference paper by adding confidence transformation, character confusion and writer-adaptive thresholds, experimenting with large database, and comparing with transcription-based keyword spotting.

The rest of this paper is organized as follows. Section 2 gives a brief review of the related works. Section 3 gives an overview of our keyword spotting system. Section 4 is devoted to the word similarity scoring by combining character classification scores and confusion information. Section 5 presents the dynamic search algorithm. Experimental results are presented in Sec. 6 and concluding remarks are given in Sec. 7.

2. Related Works

Keyword spotting was originally formulated as detecting words or phrases in speech, and then extended to locate words in printed text documents. In Ref. 29, the searchability of electronic ink was first proposed for online handwriting using the dynamic programming technique, and in Ref. 33 keyword spotting was first proposed for offline handwritten documents. The keyword spotting technique has enabled the implementation of search engines for handwritten document images.

Among the image-to-image matching methods, dynamic time warping (DTW) is tolerant of shape variation and has been used in both online and offline handwritten documents. A method based on binary shape features was shown to perform better than the DTW in terms of efficiency and effectiveness. Leydier et al. introduced a text search algorithm based on differential features and
a cohesive elastic matching method based on zones of interest in order to match only the informative parts of the words. Lu et al.\textsuperscript{30} used word shape codes to retrieve document images by either text query or image query. Jawahar et al.\textsuperscript{13} employed a handwriting synthesis module to map the text query into the ink domain.

In model-based keyword spotting, the discriminative power of the features and model learning helps improve the word matching score, but a large number of training samples are needed to guarantee the generalizability of model. Rath et al.\textsuperscript{40} designed a search engine for historical documents using a set of transcribed page images to learn a probability distribution model for the transcript words. For English and Arabic handwritten documents, one frequently used model is the hidden Markov model (HMM).\textsuperscript{3,9,42} Other frequently used models include the classification models of SVM,\textsuperscript{1} boosted decision trees\textsuperscript{10} and neural networks.\textsuperscript{6} In Ref. 48, biologically inspired features based on the visual cortex model are used for quick access to new handwritten collections. This method is used to incrementally elicit word labels in a live Web-based annotation system.

To overcome the OOV problem, one of the earliest works\textsuperscript{37} used a family of sub-word units of varying complexity generated from the phonetic N-best lists produced during automatic speech recognition. In Ref. 43, word/sub-word-based lattice information is converted into a weighted finite-state machine (WFSM). Chan et al.\textsuperscript{3} describe a lexicon-free system utilizing general HMMs with a bigram letter transition model and KPCA/LDA for letter discrimination. Cao and Govindaraja\textsuperscript{2} search words in a series of character recognition scores. Howe et al.\textsuperscript{9} employ a powerful segmentation-free letter detection method and an ensemble of HMMs based on character n-gram and physical separation models for word inference.

The proposed keyword spotting method is designed for online Chinese handwritten documents, which, as well as Japanese documents, have no extra space between words. It uses a discriminative character classification model to avoid word segmentation and the OOV problem. The character classifier is trained with multi-writer samples and OVA objective so as to better score character and word similarities.

3. System Overview

Figure 1 shows the diagram of our keyword spotting system, which contains two main parts: data storage and keyword spotting. The input document is first segmented into text lines according to the time and space interval between consecutive strokes (the pre-segmentation step in the method of Ref. 55). Each line is over-segmented into primitive segments by stroke grouping according to off-stroke distances, and consecutive segments are concatenated to generate candidate character patterns, which are represented in a segmentation candidate lattice. Examples of text line segmentation and over-segmentation are shown in Fig. 2(a), and an example of candidate pattern generation is shown in Fig. 2(b). In over-segmentation, we re-arrange delayed strokes (strokes inserted to a previous part) using spatial information and designed some rules to split connected strokes.
Fig. 1. Block diagram of the keyword spotting system.

Fig. 2. (a) Text line segmentation and over-segmentation of a line; (b) Segmentation candidate lattice generation.
After candidate pattern generation, each candidate pattern is recognized by a character classifier to assign a number of candidate classes with corresponding similarity scores. Then the index of document can be built by storing the segmentation candidate lattice including character locations, the candidate classes and corresponding similarity scores. We build the index at character level since the number of Chinese words is huge to store.

In text search, the query word is matched with sequences of candidate patterns (partial paths in the candidate lattice) with every primitive segment as the start. The word similarity is obtained by combining the character similarity scores. When the word similarity is greater than a threshold, a word instance is located in the document.

Text search can be performed in two modes: real-time matching mode and offline indexing mode. In real-time mode, the query characters are matched with the candidate patterns at the time of search, which is suitable for retrieving a small number of documents generated online. In offline mode, all the candidate patterns in the lattice are recognized to assign multiple candidate classes with similarity scores, which are stored in the index database for search. The search engine only has to compare the stored scores with thresholds, which enables fast retrieval in large databases. In terms of retrieval accuracy as well as the search algorithm, the two modes have no difference.

4. Word Similarity Measure

In text search, the query word $W = c_1, \ldots, c_n$ is matched with each sequence of candidate patterns $X = x_1, \ldots, x_n$ (each pattern represented as a feature vector) in the document. The similarity measure is designed so that true words have high scores and imposters have low scores. A measure approximating the probability $P(W|X)$ is desired, but to obtain an accurate estimate of probability is nontrivial. While this deserves close investigation in the future, we adopt the logarithm of the similarity measure defined in Ref. 1:

$$
\text{Sim}(W, X) = \log P(c_1, \ldots, c_n| x_1, \ldots, x_n) \approx \log \left( \prod_{i=1}^{n} P(c_i|x_i) \right)^{\frac{1}{n}}
= \frac{1}{n} \sum_{i=1}^{n} \log P(c_i|x_i).
$$

(1)

When the probability $P(c_i|x_i)$ is not available without confidence transformation, the similarity can be taken as:

$$
\text{Sim}(W, X) = \frac{1}{n} \sum_{i=1}^{n} \text{sim}(c_i, x_i),
$$

(2)

where $\text{sim}(c_i, x_i)$ is the similarity between candidate pattern $x_i$ and character $c_i$, given by the classifier.
The character classifier is desired to give high similarity to the genuine class of the input pattern and low scores to all the other classes. On an input noncharacter pattern (mis-segmented character), the scores of all classes should be low. For large alphabet Chinese character recognition, the nearest prototype classifier is often used due to its good tradeoff between classification accuracy and complexity. We recently proposed an algorithm for training prototype classifier with OVA objective such that the prototypes of each class perform as a binary classifier for separating the class from the others. This property is desirable for retrieval. For comparison, we also use an OVA SVM classifier and a prototype classifier trained with multi-class objective.

4.1. Prototype classifiers

The nearest prototype classifier classifies the input pattern to the class of the nearest prototype under a distance metric (usually the Euclidean distance). Prototype learning methods based on empirical error minimization, such as minimum classification error (MCE) and negative log-likelihood of margin, have been demonstrated to give high classification accuracies. However, all these methods aim to optimize the multi-class accuracy and may make the prototypes less representative of sample distribution, and thus the classifier becomes weak in outlier rejection.

4.1.1. Training with OVA criterion

The OVA prototype learning method decomposes the multi-class problem into multiple binary classification problems and train the prototypes using a OVA criterion, the cross-entropy (CE).

In the OVA formulation, each prototype \( m_{ij} \) (jth prototype of class i) functions as a binary discriminant function, and the discriminant function of a class is the maximum over the prototypes:

\[
f_i(x) = \max_j f_{ij}(x) = -\min_j (\|x - m_{ij}\|^2 - \tau_{ij}),
\]

where \( \tau_{ij} \) is the threshold for prototype \( m_{ij} \). In training, the discriminant function is transformed to binary posterior probability using the logistic (sigmoidal) function:

\[
p_i(x) = \sigma[\xi f_i(x)] = \frac{1}{1 + e^{-\xi f_i(x)}},
\]

where \( \xi \) is a constant to control the hardness of nonlinearity, which is usually set to be inversely proportional to the average within-class nearest prototype distance on training data. The training objective is to minimize the CE on a training dataset \( \{(x^n, c^n) | n = 1, \ldots, N\} \) (\( c^n \) is the class label of \( x^n \)):

\[
\min \text{CE} = - \sum_{n=1}^{N} \left\{ \sum_{i=1}^{M} \left[ y^n_i \log p_i + (1 - y^n_i) \log(1 - p_i) \right] - \alpha \|x^n - m_{c^n k_i}\|^2 \right\},
\]
where \( M \) is the number of classes, \( y^n_i = 1 \) if the training sample \( x^n \) belongs to class \( i \), and 0 otherwise. \( \alpha \|x^n - m_{c=k}\|^2 \) is a regularization term and \( m_{c=k} \) is the nearest prototype of ground-truth class \( c^n \) from \( x^n \). \( \alpha \) is set as inversely proportional to the average within-class nearest prototype distance on training data. Both the prototypes and their thresholds are optimized in training. Since the CE is a binary criterion, the classifier is trained to give high score to the genuine class of input pattern and low scores to the other classes.

Though the criterion (5) is class-decomposable, we train the prototypes of all classes simultaneously by stochastic gradient descent to overcome the imbalance of samples. For acceleration, on each training sample, we only update the parameters of the correct class and a few selected rival classes. Please see Ref. 22 for details.

After training the OVA prototype classifier, we simply use the class discriminant function (3) as the similarity: \( \text{sim}(c_i, x) = f_i(x) \), or take the logarithm of probability after confidence transformation.

4.1.2. Training with multi-class objective

For comparison, we also trained a prototype classifier using a multi-class objective, the log-likelihood of hypothesis margin (LOGM),\(^\text{15}\) which was shown to perform superiorly in multi-class classification. On a training sample \( x \) from class \( i \), the loss function is defined based on the hypothesis margin

\[
\text{f}_i(x) = g_i(x) - g_r(x),
\]

where \( g_i(x) = - \min_j(||x - m_{ij}||^2) = - ||x - m_{ik}||^2 \) is the discriminant function over prototypes to the genuine class \( i \), \( g_r(x) = \min_{i \neq i} g_l(x) = - ||x - m_{rl}||^2 \) is the discriminant function to the closest rival class \( r \). The probability of correct classification is \( p_i(x) = \sigma[\xi f_i(x)] \). On a training dataset \( \{(x^n, c^n)\}_{n=1}^N \), the objective is to minimize the negative log-likelihood loss (NLL):

\[
\min \text{NLL} = - \sum_{n=1}^N [\log p_{c^n}(x^n) - \alpha ||x - m_{c=k}||^2].
\]

The prototypes are optimized by stochastic gradient descent. Using this classifier in keyword spotting, the character similarity is defined as \( \text{sim}(c_i, x) = g_i(x) \) or the logarithm of probability after confidence transformation.

4.2. SVM classifier

Considering the large number of Chinese characters, we use a OVA SVM classifier with linear kernel. The similarity of each class is a linear discriminant function:

\[
\text{f}_i(x) = w_i^T x - \tau_i.
\]

The weight vector and the threshold of each class are calculated from support vectors selected by large margin training. For the large number of classes and huge sample
set, we use the successive overrelaxation (SOR) algorithm of Ref. 34 for SVM training.

4.3. Confidence transformation
The classifiers do not output class posterior probabilities \( P(c|x) \), so we convert classifier outputs to posterior probabilities by confidence transformation, which has been shown to benefit handwritten text line recognition.\(^{49}\) We adopt the scheme that transforms classifier outputs to sigmoidal probabilities and combines into multi-class probabilities by the Dempster–Shafer (D–S) theory of evidence. Denote the class discriminant functions as \( f_i(x), i = 1, \ldots, M \), the sigmoidal probabilities are

\[
P^{\text{sg}}(c_i|x) = \sigma[\beta_{i1} f_i(x) + \beta_{i0}] = \frac{1}{1 + \exp[-\beta_{i1} f_i(x) - \beta_{i0}]}, \quad i = 1, \ldots, M,\tag{9}
\]

where \( \{\beta_{i1}, \beta_{i0}\} \) are transformation parameters, which are estimated on a validation dataset by minimizing the CE criterion. The sigmoidal probabilities are binary ones, i.e. each one denotes the probability that the input pattern belongs to the specified class versus the others. The sigmoidal probabilities are combined into multi-class probabilities by the D–S theory of evidence:

\[
P^{\text{ds}}(c_i|x) = \frac{\exp[\beta_{i1} f_i(x) + \beta_{i0}]}{1 + \sum_{j=1}^{M} \exp[\beta_{j1} f_j(x) + \beta_{j0}]}, \quad i = 1, \ldots, M.\tag{10}
\]

The combined multi-class probabilities have the desired properties that the summation of class probabilities is guaranteed to be equal to or smaller than one, and the complement to one denotes the probability that the input pattern is an outlier (does not belong to any of the \( M \) defined classes).

4.4. Character confusion
A study has shown that the confusion between competing words is a major source of false positive errors in keyword spotting.\(^{11}\) Thus, the matching scores of competing words were used to modify the matching score of query word for improving the retrieval performance. In Chinese documents, it is not trivial to apply the word confusion information because of the difficulty of word segmentation. Hence, we instead exploit the character confusion information to modify the character similarity in word spotting. The character classifier can not only give the matching score of a specified character class (in the query word), but also provide a list of high-score classes matched with the candidate character pattern. These high-score classes are confusing characters to the queried character. We modify the character similarity in a similar way to Ref. 11. On a candidate pattern \( x \), the classifier gives the probability \( P(c_0|x) \) (after confidence transformation) of the top-rank class \( c_0 \). The similarity for the queried character \( c \) is then modified as:

\[
\text{sim}(c, x) = \eta \log P(c|x) + (1 - \eta)\left[\log P(c|x) - \log P(c_0|x)\right],\tag{11}
\]
where $0 < \eta < 1$. When the queried character is not the top-rank class on $x$, its similarity will be decreased.

### 4.5. Writer-adaptive thresholds

In keyword spotting, two thresholds $T_c$ and $T_w$ are used on the similarity scores to reject candidate characters and words, respectively. To adapt to the variable styles and writing quality of different writers, we use an adaptive threshold model\textsuperscript{16,20,51} under the reasonable Gaussianity assumption of scores:

$$T_c = \mu_m + T_{c0}\sigma_m,$$

$$T_w = \mu_m + T_{w0}\sigma_m,$$

where $\mu_m$ is the mean score of the candidate patterns (characters or words) in the whole document, and $\sigma_m$ is the standard deviation of the scores. $T_{c0}$ and $T_{w0}$ are user-defined thresholds (in $[-5,5]$) to control the acceptance rate. To calculate $\mu_m$ and $\sigma_m$ for each document, we use all the candidate scores greater than a secure threshold $t$ (=10 in our experiments), which is set empirically to accept most of the real instances of characters or words.

The adaptive thresholds enable the search algorithm to adapt to the writing style and quality of documents. For example, in neatly written documents, the average matching score is high, and so, a higher threshold is more suitable for detecting query words and reject imposters.

### 5. Word Spotting by Dynamic Search

After text line segmentation and character over-segmentation, the online handwritten document is represented as a sequence of primitive segments $s_1 \cdots s_L$, where $L$ is the total number of primitive segments in the document. Each segment combines with its successors to form candidate patterns subject to constraints of maximum number of segments, character width, width-to-height ratio and within-character gaps. The query word $W = c_1 \cdots c_n$ is matched with subsequences of $s_j \cdots s_{j+K-1}$ (partitioned into candidate patterns and $K$ is the number of primitive segments in the subsequence) with $j = 1, \ldots, L - n + 1$. The word matching similarity is calculated by Eq. (1).

To accelerate text search, we use two thresholds to prune candidate matches: a character threshold $T_c$ to prune those characters with $\text{sim}(c_i, x_i) < T_c$ and a word threshold $T_w$ to prune those words with $\text{Sim}(W, X) < T_w$.

We use a character-synchronous dynamic search algorithm for efficient search of query word from sequences of primitive segments. The algorithm is similar to that for lexicon-driven character string recognition.\textsuperscript{23} The search process repeats with every primitive segment as start. For a specific start segment $s_j$, the process is described in
Algorithm 1: Character-Synchronous Dynamic Search

**Input:** query word $W = c_1 \cdots c_n$, primitive segments from $s_j$.

**Output:** located word instance $s_j \cdots s_{j+K-1}$.

**Start:** $i = 1$, $k = j$, root node $(0,0)$ in OPEN.

**Step 1:** Match character $c_i$ with all the candidate patterns $x_i$ starting with $s_k$. Each pair $(c_i, x_i)$ satisfying $\text{sim}(c_i, x_i) \geq T_c$ and $\frac{1}{i} \sum_{p=1}^{i} \text{sim}(c_p, x_p) \geq T_w$ is stored as an OPEN node.

**Step 2:** if OPEN is empty, go to End; otherwise, go to Step 3.

**Step 3:** label all the OPEN nodes as CLOSED and expand each of them. For an OPEN node $(c_i, x_i)$ with $x_i$ formed of primitive segments $s_k \cdots s_{k+l-1}$,

- if $i = n$, go to Step 4;
- else if $k + l - 1 = L$, go to End;
- otherwise, set $i = i + 1$, $k = k + l$, go to Step 1.

**Step 4:** if there are multiple instances $s_j \cdots s_{j+K-1}$ matched with the query, retain the one of maximum score.

**End**

Algorithm 1, wherein a pair of matched character and candidate pattern is stored as a node in the state space.

In the dynamic search for a cross-line word, if the matched instance for $c_i$ ($i < n$) is the end of the line and this line is not the last one of the document, we search the next character $c_{i+1}$ from the start of the next line. Figure 3(a) shows an illustrative example of dynamic search for spotting a word from a sequence of primitive segments as shown in Fig. 2(b). After matching the last character, two segmentation paths are matched, and the one with maximum word score is retained as the spotting result. After the dynamic search in Algorithm 1, if there are multiple overlapping instances matched with a query word, the ones other than the one of maximum score will be deleted. Figure 3(b) shows two instances of the query word in a document.

---

**Fig. 3.** (a) Dynamic search for spotting a word; (b) Two spotted instances.
6. Experimental Results

We evaluated the performance of the proposed keyword spotting method on a database of online Chinese handwriting: CASIA-OLHWDB. This database is divided into six datasets, three for isolated characters (DB1.0–1.2) and three for handwritten texts (DB2.0–2.2). There are 3,912,017 isolated character samples and 52,220 handwritten pages (composed of 1,348,904 character samples) in total. Both the isolated data and handwritten text data have been divided into standard training and test subsets.

6.1. Experimental setting

We trained three character classifiers for computing the character similarity scores: nearest prototype classifier trained by one-versus-all criterion (NPC-OVA), NPC trained by the multi-class LOGM criterion (NPC-LOGM), and linear SVM. In training sets of DB1.0–1.2 (isolated characters) and DB2.0–2.2 (text pages), there are totally 4,207,801 character samples of 7356 classes (7185 Chinese characters and 171 alphanumeric characters and symbols). We used 4/5 of the character samples for training the classifiers, and the remaining 1/5 character samples as well as 268,569 outlier samples for estimating the confidence parameters. The outlier samples were selected from the noncharacter candidate patterns generated in over-segmentation on the training text pages.

For character feature extraction, we use the popularly used stroke direction histogram feature, implemented by the normalization-cooperated feature extraction method of Ref. 26 with bi-moment normalization. The resulting 512D feature vector is projected onto a 160D subspace learned by Fisher linear discriminant analysis (FLDA).5

For evaluating the retrieval performance on the test dataset of DB2.0–2.2, we used the high-frequency words in the lexicon of the Sogou labs as query words. The top 60,000 frequently used words, including 39,057 two-character words, 9,975 three-character words and 9,451 four-character words, were tested in our experiments. The keyword spotting performance is measured using three metrics: recall ($R$), precision ($P$) and the $F$-measure.54

Our experiments were implemented on a PC with Intel(R) Core(TM)2 Duo CPU E8400 3.00 GHz processor and 2GB RAM, and the algorithms were programmed in C++.

6.2. Character classification and detection performance

The word spotting performance largely depends on the character classification and detection performance of the classifier. So, we first evaluate the character classification and detection performance of the three classifiers (NPC-OVA, NPC-LOGM, and SVM). The NPC has one prototype per class such that its computational complexity is comparable to the linear SVM classifier.
On the isolated characters segmented from the test documents of DB2.0–2.2, the classification accuracies of three classifiers are 87.03\% for the NPC-LOGM, 86.11\% for the NPC-OVA, and 83.92\% for the SVM classifier. For word spotting, we retain a number of top-rank candidate classes output by each classifier such that the true class is contained at high probability. We retain 20 candidate classes for the NPC-LOGM, 30 candidate classes for the NPC-OVA, and 100 candidate classes for the SVM. The accumulated accuracies are 98.12\%, 98.12\% and 97.92\%, respectively. The number of candidate classes trades off the retrieval performance and the storage complexity in the index file. Further increasing the number of candidate classes for all the three classifiers was shown to give little improvement of retrieval performance.

We should note that the SVM classifier has much higher training complexity than the prototype classifier because the SVM training is a quadratic programming problem. The higher classification accuracy of prototype classifier over the SVM makes it retains much less candidate classes. Based on these facts, the prototype classifier is more suitable for our application of keyword spotting. Anyway, we use the SVM classifier for comparing the spotting performance.

For evaluating the character detection performance, we used the 7,356 characters modeled by the classifiers as query characters, and detected them in the test documents of DB2.0–2.2. We set variable thresholds to the class discriminant value or probabilistic confidence output by the classifier, and obtained variable results of recall and precision rates. The thresholds are empirically set to get the recall rates ranging from low to high so that the results could include the best $F$ measure. The results corresponding to the best $F$ measure are shown in Table 1. We can see that by confidence transformation (CT), the detection performance of each classifier was improved. The NPC-LOGM got the most significant improvement. This is because the confidence parameter estimation is a OVA training process and outlier samples were used in this process, while the NPC-LOGM without CT was trained with multi-class objective and thus yields poor detection performance. However, the detection performance of NPC-LOGM after CT is still inferior to that of NPC-OVA and SVM, which perform comparably. The precision-recall curves of the classifiers are shown in Fig. 4, where the NPC-OVA and SVM with CT are shown to have comparable detection performance and NPC-OVA can outperform SVM as the recall rate gets higher in the results with CT.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Without CT</th>
<th></th>
<th></th>
<th>With CT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R$ (%)</td>
<td>$P$ (%)</td>
<td>$F$ (%)</td>
<td>$R$ (%)</td>
<td>$P$ (%)</td>
<td>$F$ (%)</td>
</tr>
<tr>
<td>NPC-LOGM</td>
<td>55.72</td>
<td>58.48</td>
<td>57.07</td>
<td>78.23</td>
<td>86.55</td>
<td>82.18</td>
</tr>
<tr>
<td>NPC-OVA</td>
<td>75.31</td>
<td>87.09</td>
<td>80.77</td>
<td>80.31</td>
<td>89.45</td>
<td>84.63</td>
</tr>
<tr>
<td>SVM</td>
<td>78.12</td>
<td>89.23</td>
<td>83.30</td>
<td>81.95</td>
<td>90.95</td>
<td>86.21</td>
</tr>
</tbody>
</table>

Table 1. Character detection results of three classifiers (CT: confidence transformation).
6.3. Word spotting performance

In word spotting from the test documents, we set a safe low character threshold $T_c$ and adjusted the word threshold $T_w$ to obtain variable precision-recall points. The precision–recall curves using three classifiers with CT for two-character words are shown in Fig. 5(a). Using the NPC-OVA with CT, the precision-recall curves for words of different lengths are shown in Fig. 5(b). In Fig. 5(a), we can see that the NPC-OVA and the SVM classifier perform comparably, and better than the
NPC-LOGM. This is owing to the superior outlier rejection and character detection capability of NPC-OVA and SVM, which were trained with OVA objective. Figure 5(b) shows that longer words tend to have higher spotting performance. This is because longer words have richer linguistic context to enhance the word similarity.

The recall and precision rates corresponding to the best $F$ measure for three classifiers with and without CT are shown in Table 2. The results again justify the benefit of CT and the superiority of NPC-OVA and SVM over NPC-LOGM. The NPC-LOGM, however, performs comparably well on four-character words, due to the rich linguistic context in the words.

### 6.4. Effects of character confusion and writer-adaptive thresholds

For evaluating the effects of character confusion and writer-adaptive thresholds, we selected the classifier NPC-OVA with CT and evaluated on two-character words. The weight $\eta$ for character fusion was empirically set as 0.6. The precision–recall curves are shown in Fig. 6, where NPC-OVA-CON denotes spotting with character fusion, and NPC-OVA-ADA denotes spotting with both character confusion and writer-adaptive thresholds. We can see that the character fusion can slightly improve the spotting performance, and adapting thresholds further improves the performance. Looking at the rates of best $F$ measure, character confusion improves the $F$ measure from 74.95% to 75.36%, and threshold adaptation further improves to 75.90%.

### 6.5. Comparing with transcription-based word search

We compared the performance of the proposed character-model-based keyword spotting method with that of transcription-based word search. For transcription, we implemented a character string recognition method similar to that\(^{50}\) for recognizing the text lines in the test documents. For fair comparison, the string recognizer did not use the geometric context, which was not used in keyword spotting. The string recognizer integrates character classification scores and statistical language model

---

**Table 2. Keyword spotting results of three classifiers.**

<table>
<thead>
<tr>
<th>Word Length</th>
<th>Classifier</th>
<th>Without CT</th>
<th>With CT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$R$ (%)</td>
<td>$P$ (%)</td>
</tr>
<tr>
<td>2</td>
<td>NPC-LOGM</td>
<td>53.44</td>
<td>55.70</td>
</tr>
<tr>
<td></td>
<td>NPC-OVA</td>
<td>65.90</td>
<td>70.54</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>68.80</td>
<td>71.99</td>
</tr>
<tr>
<td>3</td>
<td>NPC-LOGM</td>
<td>72.87</td>
<td>72.02</td>
</tr>
<tr>
<td></td>
<td>NPC-OVA</td>
<td>74.30</td>
<td>81.03</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>76.76</td>
<td>82.34</td>
</tr>
<tr>
<td>4</td>
<td>NPC-LOGM</td>
<td>84.05</td>
<td>86.37</td>
</tr>
<tr>
<td></td>
<td>NPC-OVA</td>
<td>81.29</td>
<td>87.85</td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>81.94</td>
<td>86.70</td>
</tr>
</tbody>
</table>
where $n$ is the number of candidate character patterns in the path, $X$ denotes the sequence of candidate patterns, $C$ denotes the string class (character sequence); $P(c_i|x_i)$ is the classification confidence for class $c_i$ on candidate pattern $x_i$, $P(c_i|c_{i-1})$ is the probability of character bi-gram; $k_i$ is the number of primitive segments composing the candidate pattern $x_i$. The weighting parameter $\lambda$ is estimated on a training set of handwritten texts (randomly selected from the training documents of DB2.0–2.2) under the MCE criterion. The optimal path is searched for by dynamic programming (DP), which has been used in radical-based Chinese character recognition, to give the string recognition result. The character classifier is NPC-LOGM which gives the best string recognition performance among the three classifiers.

For word search, the query word is searched for from the transcribed texts of test documents. Since string recognition gives unique text output, transcription-based word search gives a unique point of precision–recall rates. The results are given in Table 3, and are compared with the precision–recall curves of NPC-OVA model-based spotting in Fig. 7. Both the NPC-LOGM and NPC-OVA are with CT in string recognition or word spotting. When comparing the precision rate at the same recall rate or comparing the recall rate at the same precision rate, we can see that transcription-based search outperforms the proposed character model-based spotting. However, the proposed method can flexibly trade off the precision and recall rates, and can raise the recall rate (which is desired in many applications) by lowering the precision. Similar results have been reported in Ref. 7.
It is noteworthy that the string recognizer utilizes the linguistic context, while the character-model-based spotting method does not use a language model. How to effectively exploit the linguistic dependency of spotted word with its context deserves in-depth study in the future.

7. Concluding Remarks

We presented a character-model-based keyword spotting method for online Chinese handwritten documents, overcoming the difficulty of word segmentation and the OOV problem. We proposed to use OVA trained character classifier, which gives high similarity to target characters and low similarity to imposters. The word similarity is obtained by combining the character similarity scores. We also use the character confusion information and writer-adaptive thresholds to further improve the spotting performance. The experimental results demonstrate the superiority of OVA classifier training and the effectiveness of confidence transformation, character confusion and writer-adaptive thresholds. Our OVA trained prototype classifier performs better than the one trained with multiple class objective function, and consumes much less storage of index file than the SVM classifier. The future works are aimed to further improve the word matching similarity by combining the

\[
\text{Table 3. Transcription-based word search results for words of different lengths.}
\]

<table>
<thead>
<tr>
<th>Word Length</th>
<th>R (%)</th>
<th>P (%)</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>81.71</td>
<td>93.68</td>
<td>87.29</td>
</tr>
<tr>
<td>3</td>
<td>78.72</td>
<td>98.47</td>
<td>87.49</td>
</tr>
<tr>
<td>4</td>
<td>73.51</td>
<td>99.35</td>
<td>84.50</td>
</tr>
</tbody>
</table>

Fig. 7. Precision–recall curves of word spotting and results of transcription-based search.
geometric context and language model. The proposed method is also applicable to offline handwriting keyword spotting provided that the text line segmentation, character segmentation and feature extraction techniques are customized to offline handwriting.

Acknowledgment

This work has been supported in part by the National Natural Science Foundation of China (NSFC) Grants 60825301 and 60933010, the CAS Visiting Professorship for Senior International Scientists Grant 2011T1G20.

References

Keyword Spotting from Online Chinese Handwritten Documents


Horst Bunke joined the University of Bern as a Professor of Computer Science in 1984. He is now a Professor Emeritus. Horst served as 1st Vice-President and Acting President of the International Association for Pattern Recognition (IAPR). He is a former Editor-in-Chief of the International Journal of Pattern Recognition and Artificial Intelligence, and former member of the editorial board of various journals. Horst is the recipient of the 2010 KS Fu Prize, awarded by the IAPR. Moreover, he received the IAPR/ICDAR Outstanding Achievements Award in 2009 and an honorary doctorate degree from the University of Szeged, Hungary, in 2007. He held visiting positions at the University of Notre Dame (Melchor Visiting Professor 2012), Chinese Academy of Science, Beijing (1987, 2012), the IBM Los Angeles Scientific Center (1989), the University of Szeged, Hungary (1991), the University of South Florida at Tampa (1991, 1996, 1998–2007 yearly), the University of Nevada at Las Vegas (1994), Kagawa University, Takamatsu, Japan (1995), Curtin University, Perth, Australia (1999), Australian National University, Canberra (2005), Autonomous University, Barcelona (2005, 2012, 2013), and NICTA, Brisbane (2009) and Melbourne (2011). Horst Bunke has more than 700 publications to his name, including over 40 authored, co-authored, edited or co-edited books and special editions of journals. His h-index is 61, according to Google Scholar. In the DBLP Computer Science Bibliography, which captures more than 2.2 million papers from the whole discipline of Computer Science, Horst Bunke ranks among the 120 most prolific authors.