Image and Vision Computing 31 (2013) 958–968

Contents lists available at ScienceDirect

Image and Vision Computing

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

Keyword spotting in unconstrained handwritten Chinese documents using contextual word model ☆

Liang Huang [a], Fei Yin [b], Qing-Hu Chen [a], Cheng-Lin Liu [b,*]

[a] School of Electronic Information, Wuhan University, 39 Luoyu Road, Wuhan, Hubei 430079, PR China
[b] National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing 100190, PR China

ARTICLE INFO

Article history:
Received 13 June 2012
Received in revised form 22 August 2013
Accepted 10 October 2013

Keywords:
Keyword spotting
Chinese handwritten documents
Word similarity
Contextual word model

ABSTRACT

This paper proposes a method for keyword spotting in off-line Chinese handwritten documents using a contextual word model, which measures the similarity between the query word and every candidate word in the document by combining a character classifier and the geometric context as well as linguistic context. The geometric context model characterizes the single-character likeliness and between-character relationship. The linguistic model utilizes the dependency of the word with the external adjacent characters. The combining weights are optimized on training documents. Experiments on a large handwriting database CASIA-HWDB demonstrate the effectiveness of the proposed method and justify the benefits of geometric and linguistic contexts. Compared to transcription-based text search, the proposed method can provide higher recall rate, and for spotting words of four characters, the proposed method provides both higher precision and recall rate.

1. Introduction

Due to the huge volume of existing documents and the ever increasing new documents in daily life, the need of efficient document retrieval techniques is prominent. On scanning documents into digital images, character recognition and retrieval techniques can help efficiently sort the documents, summarize, find documents and locate regions of interest. Among the techniques of document retrieval [1,2], keyword spotting finds relevant documents containing the queried words and locates the word instances for further investigation. For printed documents of degraded image or handwritten documents, keyword spotting is still an unsolved problem due to the difficulty of word segmentation and recognition. Traditional character and word recognition techniques do not give sufficiently high accuracy on these documents, such that text search based on transcription (text line recognition) does not perform satisfactorily. For example, a state-of-the-art handwriting recognizer reports word recognition rate of 79.7% on online data and 74.1% on off-line data [3].

Compared to transcription-based text search, keyword spotting has the advantage that the information related to the query word can be exploited adequately to improve the recall rate of retrieval. This is particularly important for the less frequently used words, which are usually recognized less accurately by traditional recognizers. As a two-class (binary) problem, keyword spotting can easily get variable points of precision–recall tradeoff by setting variable decision thresholds. Transcription-based search is less flexible in this respect, particularly, when the recognition accuracy is low. The recall rate of transcription-based search can be improved by giving multiple alternative recognition results, but this requires a large number of alternatives for guaranteeing high recall while sacrificing the precision.

Depending on the format of query, keyword spotting techniques can be grouped into two categories: query-by-example (QBE) and query-by-string (QBS). In QBE, the user manually locates a word instance in the document and uses it as a template for locating the other instances. This method works well only for documents of a specific style (imaging condition, font or handwriting style). QBS is more convenient for the users to enter the query by keyboard, and is flexible for spotting words of arbitrary style. By this method, the character/word model needs to be trained with a large number of samples. Previous works have contributed largely to efficient word template matching for example-based spotting [4–10] and character/word model learning and model inference for text-query-based spotting [11–17]. To avoid word segmentation errors, some methods spot words from the document without layout analysis [10,18], while some perform text line-based spotting without explicit word segmentation [15–17,19].

The retrieval and keyword spotting of Chinese [18,20,21] and Japanese documents [22–26] have some characteristics different from that of English documents. Chinese/Japanese documents have no difference between inter-character and inter-word spaces. Therefore, the recognition and retrieval methods of Chinese documents mostly adopt character models and perform character/word segmentation during recognition/matching. It is also complicated by the large number of character classes (over 5000 classes are frequently used). To alleviate the search complexity of recognition, a character classifier is usually

☆ This paper has been recommended for acceptance by Seong-Whan Lee.
* Corresponding author. Tel.: +86 10 82544797; fax: +86 10 82544594.
E-mail address: liucl@nlpr.ia.ac.cn (C.-L. Liu).

0262-8856/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2013.10.003

used to assign a small number of high confidence classes to the input character pattern [27].

Oriented for text query-based retrieval, this paper proposes a text line-based keyword spotting method for off-line handwritten Chinese documents using a contextual word model, which combines the character classification scores on candidate character patterns, the geometric relationship and linguistic dependency between characters. Candidate words are generated by concatenating consecutive primitive segments after character over-segmentation. By training the character classifier on large number of samples, the proposed method yields promising performance on multi-writer handwritten documents. The benefits of the geometric and linguistic contexts, especially those between the word and external characters, are justified. When retrieving four-character words, the proposed method provides both higher precision and recall rates than transcription-based text search. This paper is an extension to our previous conference paper [28] by adding external context dependency, optimizing the combing weights, performing evaluation on a larger dataset and comparing with transcription-based search.

The rest of this paper is organized as follows. Section 2 briefly reviews the related works. Section 3 gives an overview of our Chinese handwritten keyword spotting system. Section 4 describes the contextual word model and Section 5 describes the word model matching procedure for spotting. Experimental results are presented in Section 6 and concluding remarks are offered in Section 7.

2. Related works

In many works of example-based query, the query image (template) and the candidate word image have been represented as sequences of features and matched by dynamic programming (DP), also called as dynamic time warping (DTW) [4], which is noise and deformation tolerant. Lopresti and Tomkins show that the DP can be applied to the matching of multiple levels of representation [29]. For DTW matching, a word image is usually represented as a sequence of primitive segments (components), sampled projections and upper/lower profiles [4], or features extracted from sliding windows or segments.

Some researchers have sought for fast feature extraction and matching techniques for printed document retrieval. Zhang et al. [5] showed that using gradient-based binary features in word matching can provide higher accuracy and much faster speed than DTW matching of profile features. Kesidis et al. [6] calculate the Euclidean distance between word images on extracting features of pixel density of zones and profile projections, and refine the search procedure by user feedback. Some methods represent the word or document images as bags of shape codes [7,8]. This approach can apply to the retrieval of word, phrase or whole document, and accept either image or text query, unlike that some methods need to synthesize query images from texts [10,30]. Lu et al. [8] code the word features of ascenders/descenders, character holes and water reservoirs, and calculate the similarity between vectors of frequencies of shape codes. Khurshid et al. [9] represent a word image as a sequence of sub-patterns, each sub-pattern as a sequence of feature vectors, and calculate a segmentation-driven edit (SDE) distance between words. Leydier et al. [10] proposed a text search method using differential features and cohesive elastic matching based on zones of interest.

In keyword spotting using statistical models, Rath et al. [11] learn a probabilistic distribution about word image features for each keyword. Kuo and Aggazzi [12] learn two pseudo 2-D HMMs for each keyword, representing the actual keyword and all the other extraneous words, respectively. Rodriguez-Serrano and Perronnin [13] train word models (continuous HMM and semi-continuous HMM) using few examples per keyword, and normalize the word score against the background model. Howe et al. [14] train letter detectors using joint boosting with gradient histogram feature, and use ensemble of HMMs for keyword spotting. Cao et al. [15] calculate keyword similarity by combining character recognition scores and integrating the word segmentation probabilities so as to overcome imperfect word segmentation. Fischer et al. [16] calculate the keyword similarity using character HMMs. Frinken et al. [17] use a recurrent neural network to output a sequence of letter probabilities on a text line image, and infer the keyword probability on the sequence. These character model-based methods enable out-of-vocabulary (OOV) word spotting and spotting from text lines without explicit word segmentation.

Keyword spotting methods for Chinese/Japanese documents [22–26] commonly use character models because of the large number of character classes and the inability of explicit word segmentation. On over-segmenting the text line into primitive segments (each being a character or a part of a character), candidate characters are generated by concatenating primitive segments and are classified by a character classifier for assigning classes of high probability. The query keyword is searched from the lattice of candidate characters and classes. The character classifier is desired to have both high classification accuracy and resistance to noise and background patterns. A one-versus-all classifier training strategy [31] is shown to provide this property and has been successfully applied to Chinese handwritten keyword spotting [32].

The proposed method in this paper follows the strategy of over-segmentation and character classification-based spotting from text line image. To improve the spotting performance, we combine the classification scores and the geometric and linguistic contexts between characters within the word and out of the word, and the combining weights are optimized on labeled data of text lines.

3. System overview

Fig. 1 shows the block diagram of our system for keyword spotting in handwritten Chinese documents. First, the document image is segmented into text lines using a graph-based clustering algorithm [35]. Each line is over-segmented into primitive segments using the algorithm of [27]. Candidate characters generated by concatenating consecutive segments form a candidate segmentation lattice (Fig. 2). For keyword spotting, each segment in the text line is hypothesized as the start of the query word, and the constituent characters of the word are matched with candidate characters sequentially. A complete match of the word with similarity greater than a threshold gives a spotted word instance. If a partial word match is given at the end of a text line, the partial match is extended to the start of the succeeding text line. After searching through the whole document, if any two or multiple spotted instances have overlapping segments, only the one of maximum similarity is retained.

For matching the query word with a sequence of primitive segments at a specific start, we use a character-synchronous beam search algorithm [27]. An illustrative example of the search procedure for a query word is shown in Fig. 3. Starting with a primitive segment, the candidate characters are matched with the first character in the query word with the similarity given by a character classifier. A number of candidate characters with maximum similarity are retained, and their succeeding segments form new candidate characters to match with the second character in the query word. This continues until the last character of the query word is matched or the word matching is terminated due to the low similarity of partial match. In Fig. 3, the path of thick line gives a complete match of word. Fig. 4 shows the document with the query word spotted.

The above procedure is for spotting keyword from one document. When searching a collection of multiple documents, the documents are processed one by one. For fast search of a large collection of documents, the documents can be processed off-line with the keywords in a lexicon (containing a large number of words of possible use) to be spotted. The locations of the spotted words are stored in index files for online search. This is an implementation issue for practical application and needs no detailed description in this paper. Instead, this paper focuses on the key issue of word similarity calculation.

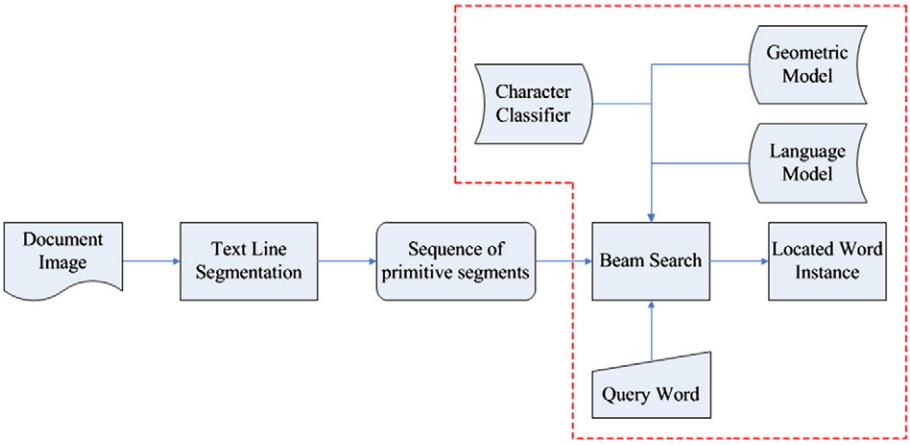

Fig. 1. Block diagram of the keyword spotting system.

4. Contextual word model

The key issue of word matching is to calculate the similarity between a query word $W = c_1 \cdots c_L$ and a sequence of primitive segments $\langle s_1 \cdots s_n \rangle$, which is segmented into a sequence of candidate character patterns $X = x_1 \cdots x_m$ (each pattern is represented as a feature vector). The similarity measure is designed such that true positives of query word have high scores and the negatives have low scores. For measuring the word similarity, take the logarithm of the posterior probability $p(W|X)$:

$$logp(W|X) = logp(X|W) + logp(W) - logp(X) \quad (1)$$

where $logp(W|X)$ is modeled by the character classifier, $logp(W)$ is the score of language model, but when the query word is given, the language score is a constant and can be ignored. $logp(X)$ can be seen as the score of geometric context, measuring the likelihood of character segmentation and between-character geometric relationship. The character classifier, the geometric model and the context fusion model are detailed in the following.

4.1. Character classifier

During keyword search, each candidate character pattern is classified to assign a few classes. The character classifier is desired to give high score to the true class of the input pattern and low scores to all the other classes. The recently proposed one-versus-all prototype classifier, with prototypes learned by optimizing a one-versus-all (OVA) objective [31], is shown to perform well for multi-class pattern retrieval [32]. In the OVA trained prototype classifier, each prototype is attached with a threshold of distance to measure the probability of accept/reject input pattern, and all the prototype vectors and thresholds are trained to optimize a one-versus-all cross-entropy objective. This is different from most other prototype learning algorithms (e.g. [33]) that aim to optimize a multi-class classification objective disregarding the rejection of outlier patterns. We use this OVA prototype classifier for scoring character similarity in our keyword spotting system. We also use a one-versus-all linear SVM (support vector machine) classifier [34]. Nonlinear SVM has very high complexity of training and classification, and thus is not used in our case of large category set classification.

Besides classifier design, the preprocessing and feature extraction of character pattern are also important. We use a state-of-the-art feature extraction technique in handwritten Chinese character recognition: normalization-cooperated gradient feature extraction [36] combined with a pseudo two-dimensional shape normalization method, and line density projection interpolation [37]. By 8-direction gradient decomposition and 8×8 sampling, the resulting 512-D feature vector is reduced to 160-D by Fisher linear discriminant analysis (FLDA).

Since the classifier learning algorithms are not the contribution of this work, they are not described in details here. The prototype classifier has one or multiple prototype vectors for each class, which are optimized on a training dataset by stochastic gradient descent. For classification, the test pattern is classified to the class of the nearest prototype in squared Euclidean distance. The discriminant function of one class is the minus of the distance from the test pattern to the nearest prototype of the class. More details can be found in [31]. The one-versus-all linear SVM classifier has similar run-time complexity with a prototype classifier. It gives a linear discriminant score to each class, and the test pattern is classified to the class of maximum discriminant

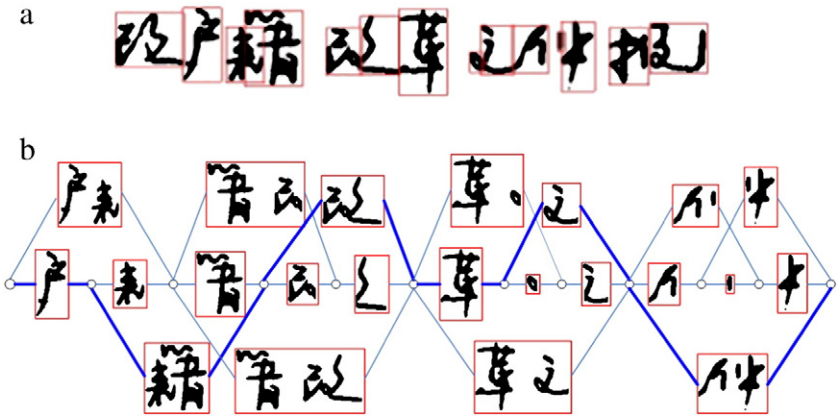

Fig. 2. Over-segmentation into primitive segments (a) and candidate segmentation lattice (b).

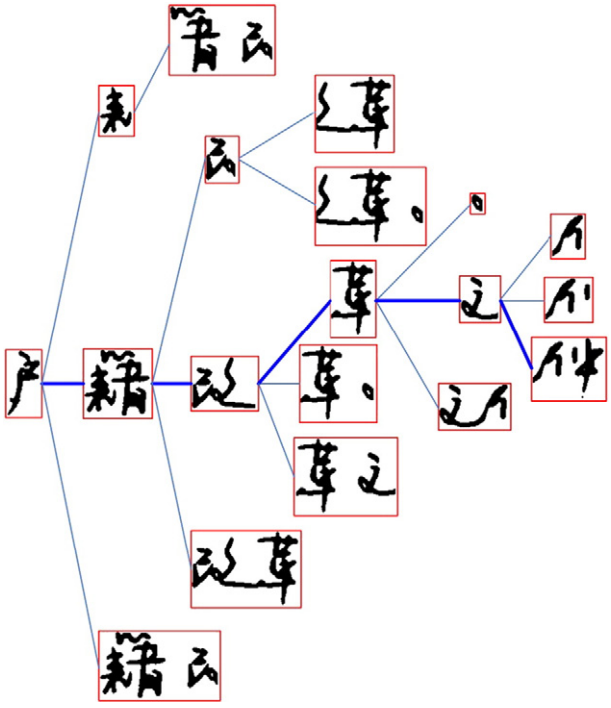

Fig. 3. Beam search for spotting a word.

score. For training the linear SVM, we use the successive over-relaxation (SOR) algorithm of [38], which is efficient for training with large sample set.

4.2. Geometric context model

The character classifier utilizes the character shape information after position normalization and size normalization. The size and position of character in the text line, and the geometric relationship between adjacent characters, are important cues for judging the legibility of characters. We call these as geometric features and use statistical models to measure the geometric likelihood. The statistical geometric models are dichotomized into single-character and between-character ones, and into class-dependent and class-independent ones [39]. We follow the scheme of [39] to cluster the character classes into six super-classes, and estimate six Gaussian models for class-dependent single-character geometry. The logarithm of Gaussian density, also called as quadratic discriminant function (QDF), is taken as the geometric score. The class-dependent between-character geometry is modeled as 36 Gaussian densities for 36 pairs of super-classes. The class-independent single-character and between-character geometric scores are measured using binary linear SVM, to evaluate the legibility of a candidate character or a pair of candidate characters. The details of geometric features can be found in [39].

4.3. Confidence transformation

Unlike statistical language model that gives probabilities of character/word n-grams directly, the character classifier or geometric models (two-class or multi-class classifiers) output discriminant scores that are not probabilities. For regulating the scales of scores and facilitating fusion in the contextual word model, the classifier outputs can be transformed to probabilities using the logistic (sigmoidal) function:

$$P(c_i|\mathbf{x}) = \frac{1}{1 + exp[-a_i d_i(\mathbf{x}) + b_i]}, \tag{2}$$

where $d_i(\mathbf{x})$ is the discriminant score (classifier output) for class c_i. The parameters $\{a_i, b_i\}$ are estimated by optimizing a log-likelihood (cross-entropy) objective on a validation dataset [40].

After confidence transformation (CT), the logarithm $logP(c_i|\mathbf{x})$ is used as a new classification/geometric score for the class c_i. Either the original classifier output $d_i(\mathbf{x})$ (before CT) or the new score $logP(c_i|\mathbf{x})$ is fused in the contextual word model.

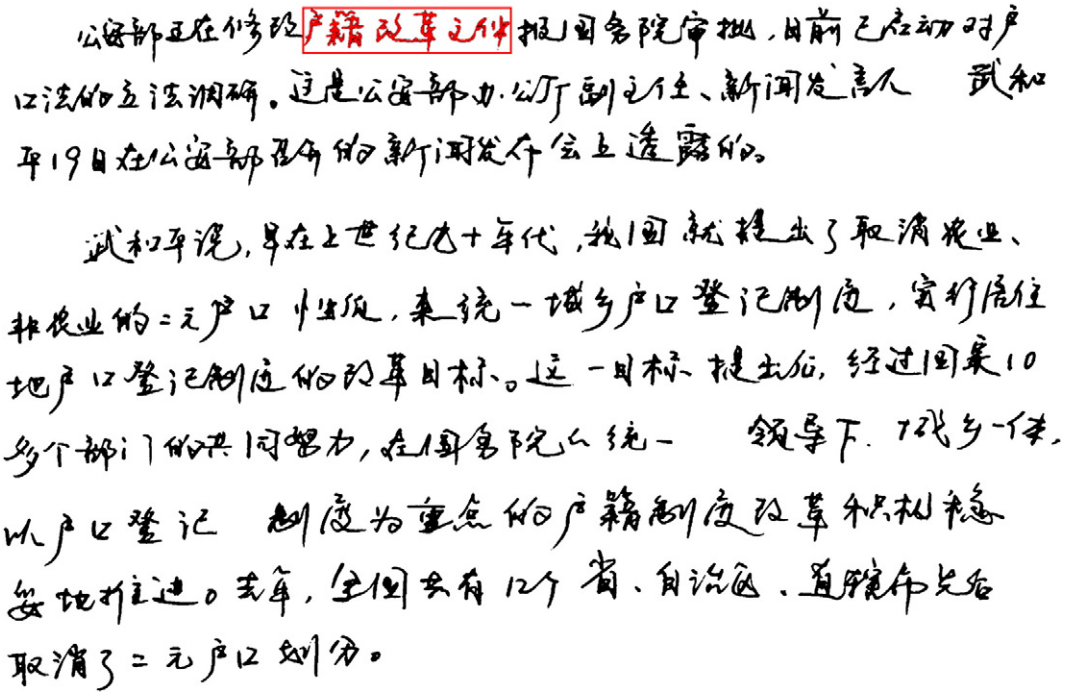

Fig. 4. Document image with keyword spotted.

4.4. Fused word model

Given a query word $W = c_1 \cdots c_L$ and a sub-sequence of primitive segments $\langle s_p \cdots s_q \rangle$ as a candidate word instance, we measure the word similarity by combining the character classification score and geometric scores. We assume that each query character c_i is matched with a candidate character formed by concatenating k_i segments $\langle s_j \cdots s_{j+k_i-1} \rangle$. For scoring a single character, we combine the classification score and single-character geometry:

$$g_c\left(c_i, s_j \cdots s_{j+k_i-1}\right) = \sum_{h=0}^{2} \alpha_h \cdot f_h - \tau_{c_i}, \tag{3}$$

where α_h, and $h = 0, 1, 2$, are the combining weights. f_0, f_1 and f_2 represent the character classification score (discriminant function), class-dependent and class-independent single-character geometric scores, respectively. They are either the classifier output or the logarithm of transformed probability. The parameter τ_{c_i} can be either class-dependent or class-independent.

For measuring the word similarity, we combine the character classification score and four geometric scores:

$$g_W(A) = \frac{1}{L} \sum_{h=0}^{4} \alpha_h \cdot F_h - \tau_W, \tag{4}$$

where $A = \left\{ \left(c_1; s_p \cdots s_{p+k_1-1}\right), \cdots, \left(c_L; s_{q-k_L+1} \cdots s_q\right) \right\}$ denotes the alignment of the sub-sequence $\langle s_p \cdots s_q \rangle$ with the query word W. α_h, $h = 0, \cdots, 4$, are the combining weights. F_0 is the sum of character classification scores. F_h, $h = 1, \cdots, 4$, are the sums of single-character geometric scores and between-character geometric scores, respectively. Specifically, the sums of scores are

$$F_h = \sum_{i=1}^{L} f_h\left(c_i, s_{j_i} \cdots s_{j_i+k_i-1}\right), \quad h = 0, 1, \tag{5}$$

$$F_2 = \sum_{i=1}^{L} f_2\left(s_{j_i} \cdots s_{j_i+k_i-1}\right), \tag{6}$$

$$F_3 = \sum_{i=2}^{L} f_3\left(\left(c_i, s_{j_i} \cdots s_{j_i+k_i-1}\right) \middle| \left(c_{i-1}, s_{j_i-k_{i-1}} \cdots s_{j_i-1}\right)\right), \tag{7}$$

and

$$F_4 = \sum_{i=2}^{L} f_4\left(s_{j_i} \middle| s_{j_i-1}\right). \tag{8}$$

In the above, f_0/f_1 are class-dependent single-character classification/geometric scores, f_2 is class-independent single-character geometric score, f_3 is class-dependent between-character geometric score, and f_4 is class-independent between-character geometric score. f_4 characterizes the relationship between two adjacent primitive segments, and thus, evaluates the likelihood of the segmentation point between two segments [39].

4.5. Fusion of external context

The relationship between the word instance and its preceding and succeeding characters, called external context, can be incorporated in the word similarity to further improve the accuracy of the model. This is like a query extension but the classes of preceding and succeeding characters are unknown. We obtain the fused model with external context by marginalizing the similarity with respect to the external adjacent character classes.

Given a query word $W = c_1 \cdots c_L$ and its instance with alignment $A = \left\{ \left(c_1; s_p \cdots s_{p+k_1-1}\right), \cdots, \left(c_L; s_{q-k_L+1} \cdots s_q\right) \right\}$, we can extend the alignment by assuming the extended word $W^e = c_0 c_1 \cdots c_L c_{L+1}$ as

$$A^e = \left\{ \left(c_0; s_{p-k_0} \cdots s_{p-1}\right), \left(c_1; s_p \cdots s_{p+k_1-1}\right), \cdots \left(c_L; s_{q-k_L+1} \cdots s_q\right), \left(c_{L+1}; s_{q+1} \cdots s_{q+k_{L+1}}\right) \right\}. \tag{9}$$

The fused discriminant function for the extended word is

$$g_{W^e}(A^e) = \frac{1}{L} \sum_{h=0}^{4} \alpha_h \cdot F_h + \sum_{h=0}^{5} \beta_h \cdot G_h - \tau_{W^e}, \tag{10}$$

where F_h, $h = 1, \cdots, 4$, have the same meaning as in the fused word model (Eq. (4)). The external context scores are given by G_h, $h = 1, \cdots, 5$, as specified by

$$G_h = \sum_{i=0,L+1} f_h\left(c_i, s_{j_i} \cdots s_{j_i+k_i-1}\right), \quad h = 0, 1, \tag{11}$$

$$G_2 = \sum_{i=0,L+1} f_2\left(s_{j_i} \cdots s_{j_i+k_i-1}\right), \tag{12}$$

$$G_3 = \sum_{i=1,L+1} f_3\left(\left(c_i, s_{j_i} \cdots s_{j_i+k_i-1}\right) \middle| \left(c_{i-1}, s_{j_i-k_{i-1}} \cdots s_{j_i-1}\right)\right), \tag{13}$$

$$G_4 = \sum_{i=1,L+1} f_4\left(s_{j_i} \middle| s_{j_i-1}\right), \tag{14}$$

and

$$G_5 = \sum_{i=1,L+1} logp(c_i | c_{i-1}), \tag{15}$$

where G_5 represents the linguistic dependency (given by a bi-gram language model) between the query word and the preceding character and the succeeding character. G_h, $h = 0, 1, 2$, represents the single-character classification and geometric scores, and G_3 and G_4 represent the between-character (binary) geometric scores.

The fused score $g_{W^e}(A^e)$ can be transformed to a word matching probability by

$$p(c_0 W c_{L+1} | X) = \frac{1}{1 + exp[-g_{W^e}(A^e)]}. \tag{16}$$

Since the numbers of primitive segments and the class labels of the preceding and succeeding characters are uncertain, the word similarity is marginalized with respect to all the possible candidate patterns and classes for the preceding and succeeding characters:

$$P(W|X) = \sum_{c_0 \in D_0, c_{L+1} \in D_{L+1}} p(c_0 W c_{L+1} | X), \tag{17}$$

where D_0 denotes the set of candidate patterns with hypothesized class c_0 preceding c_1, and D_{L+1} denotes the set of candidate patterns with hypothesized class c_{L+1} succeeding c_L. To avoid summing over a large number of hypothesized candidate patterns and classes for c_0 and c_{L+1}, the marginalization can be simplified by the maximum:

$$P(W|X) = \max_{c_0 \in D_0, c_{L+1} \in D_{L+1}} p(c_0 W c_{L+1} | X). \tag{18}$$

$P(W|X)$ is finally compared with a probability threshold to decide whether to accept the spotted word or not.

4.6. Fusion parameter estimation

In the above fused character similarity model (Eq. (3)), word similarity model (Eq. (4)) and extended word model (Eq. (10)), there are some weights and thresholds to be estimated. The objective of parameter estimation is to promote the similarity of correct character/word matching and depress the similarity of incorrect matching. For training the character similarity of Eq. (3), we use character samples and non-character samples segmented from text line images, where geometric features are available. Again we minimize a regularized cross-entropy (CE) criterion on these samples. On a sample set $\{(x^n,c^n)|\ n=1,\cdots,N\}$ (c^n is the class label of sample $x^n=s_j\cdots s_{j+k-1}$, which contains both character classification score and single-character geometric scores), the training objective is

$$min\ CE_1=-\sum_{n=1}^{N}\left\{\sum_{i=1}^{M}[y_i^n log(p_i)+(1-y_i^n)log(1-p_i)]\right\}+\frac{\gamma}{2}\sum_{h=0}^{2}\alpha_h^2, \quad (19)$$

where M is the number of character classes, $y_i^n=1$ if x^n is a sample of class i and $y_i^n=0$ otherwise, $\gamma\in[0,1]$ is the regularization coefficient, and

$$p_i(x^n)=\frac{1}{1+exp[-g_c(c_i,x^n)]}=\sigma[g_c(c_i,x^n)]. \quad (20)$$

The parameters α_h, $h=0,1,2$ and τ_{c_i}, $i=1,\cdots,M$, are optimized by stochastic gradient descent. The objective (Eq. (19)) is the union of multiple binary problems. Each true character sample is the positive sample of its labeled class and the negative sample of all the other classes, and a non-character sample is the negative sample for all the M classes. To overcome the imbalance of samples (some classes have no samples in the text lines), we use a common threshold shared by all classes: $\tau_{c_i}=\tau_c$. To accelerate training, we may select a portion of non-characters that have high character similarity using the model with up-to-date parameters.

For estimating the parameters of word similarity in Eq. (4), we extract positive and negative word samples from training text lines. We use all the concatenations of adjacent two, three, or four characters in the text lines as positive word samples. For each word in the positive samples, the negative samples are extracted during training by using the word similarity model with up-to-date parameters to spot the word from training text lines. The spotted word instances with similarity higher than a threshold and the overlap ratio with positive samples lower than 0.9 are selected as negative samples. The calculation of overlap ratio will be specified in the experimental section. The similarity threshold is selected such that there are about five times of negative samples as many as positive samples. Assuming there are M' words in the training text lines and each word W_i has N_i samples (including positive and negative samples), the objective of parameter estimation is

$$min\ CE_2=-\sum_{i=1}^{M'}\left\{\sum_{n=1}^{N_i}[y_i^n log(p_i)+(1-y_i^n)log(1-p_i)]\right\}+\frac{\gamma}{2}\sum_{h=0}^{4}\beta_h^2, \quad (21)$$

where $y_i^n=1$ for positive samples and $y_i^n=0$ for negative samples, and for a word sample (W_i,A_i^n), where A_i^n denotes the alignment of a word W_i with image segments, the probability is

$$p_i(A_i^n)=\frac{1}{1+exp[-g_W(A_i^n)]}=\sigma[g_W(A_i^n)], \quad (22)$$

where $g_W(A_i^n)$ is calculated as in Eq. (4). The objective (Eq. (21)) is minimized to optimize the parameters by stochastic gradient descent.

For estimating the parameters for the extended word model (Eq. (10)), we use the same word samples as for the word similarity parameter estimation above. The positive word samples are extended to include the preceding and succeeding true characters in the text lines, while the negative samples are extended to include the preceding and succeeding candidate patterns with maximum extended similarity (Eq. (18)). The training objective is now:

$$min\ CE_3=-\sum_{i=1}^{M'}\left\{\sum_{n=1}^{N_i}[y_i^n log(p_i)+(1-y_i^n)log(1-p_i)]\right\}+\frac{\gamma}{2}\left(\sum_{h=0}^{4}\alpha_h^2+\sum_{h=0}^{5}\beta_h^2\right). \quad (23)$$

For an extended word sample (W_i^e,A_i^{en}), where A_i^{en} is the alignment of extended word W_i^e with image segments, the probability is

$$p_i(A_i^{en})=\frac{1}{1+exp[-g_{W^e}(A_i^{en})]}=\sigma[g_{W^e}(A_i^{en})]. \quad (24)$$

where $g_{W^e}(A_i^{en})$ is calculated as in Eq. (10). Again, the objective (Eq. (23)) is minimized to optimize the parameters by stochastic gradient descent.

5. Keyword spotting procedure

The query word is searched for in the text lines of documents using a character-synchronous beam search strategy as detailed below. After text line segmentation and character over-segmentation, each text line image is represented as a sequence of primitive segments and the document is viewed as a concatenated sequence of primitive segments $I=s_1\cdots s_n$. Each segment combines with its successors to form candidate patterns subject to constraints of maximum number of constituent segments, constraints of character width and width-to-height ratio. We assume that there is a subsequence of primitive segments $I_W=s_p\cdots s_q$ to align with the query word, and a character c_i is aligned with $I_{c_i}=s_{j_i}\cdots s_{j_i+k_i-1}$. We allow a candidate character pattern to be formed by at most four primitive segments ($1\le k_i\le 4$).

To accelerate word spotting, we use two thresholds to prune candidate matches: a character threshold T_c to prune those candidate character patterns with $g_c(c_i,I_{c_i})<T_c$, and a word threshold T_W to prune those word candidates with $g_W(A)<T_W$. For a partial sequence of primitive segments $s_p\cdots s_{j_i+k_i-1}$ matched with partial word $w=c_1\cdots c_i$ ($i>L$), the partial word similarity $g_W(c_1\cdots c_i,s_p\cdots s_{j_i+k_i-1})$ has the same form as those in Eqs. (4)–(8) except that L is replaced with i. Using the two pruning thresholds, word spotting is conditioned on the character spotting: only when each character in the word is spotted (character similarity over a threshold), will the (partial) word similarity be calculated. When the last character of the query word is reached and the full word similarity exceeds the word threshold, the extended word similarity (Eq. (10)) is calculated and compared with a final threshold T_{W^e} to give decision of accept (spotted) or reject. The final threshold can be variable for balancing the tradeoff between precision and recall rates.

The character-synchronous beam search algorithm is similar to that used in lexicon-driven character string recognition [27]. The search process repeats with every primitive segment as start. For searching from a specific start segment s_{j_1}, the detailed search process is shown in Algorithm 1. In the description below, $g_c(\cdot)$ is the character similarity, $g_W(\cdot)$ is the word similarity, and $g_{W^e}(\cdot)$ is the extended word similarity. PM denotes the set of partial word matches. j_i denotes the start segment index for character c_i, k_i is the number of segments concatenated, and K is the maximum number of segments to be concatenated.

After character-synchronous search, if the query word has multiple spotted instances with the same start primitive segment, only the one with maximum word similarity is retained.

6. Experimental results

We conducted experiments on a Chinese handwriting database CASIA-HWDB [41], which contains isolated characters (in three datasets DB1.0, DB1.1 and DB1.2) and handwritten texts (in three datasets DB2.0,

DB2.1 and DB2.2) produced by 1020 writers, and is partitioned into a training set of 816 writers and a test of 204 writers. The keyword spotting performance was evaluated on the test set of 1015 handwritten text pages of 204 writers.

For training character classifiers, we used a training set containing 4,198,494 samples of 7356 classes, which are the isolated character samples and the characters segmented from the text pages of the 816 training writers. For confidence transformation (CT), we need some samples for estimating the confidence parameters (two parameters in sigmoidal probability for each class, i.e., class-dependent confidence parameters for the purpose of retrieval). To do so, we used 4/5 of training character samples for classifier training and the 1/5 for confidence parameter estimation, in the same way as [42]. For estimating the geometric models, we used 41, 781 training text lines, and also transform the geometric model outputs to confidence, as done in [42]. After CT, the logarithm of probability output by character classifier and geometric models is used in the similarity models of Eqs. (3), (4), and (10) as f_h, $h = 0, \cdots, 4$. The character bi-gram language model used in the extended word model is the same one as used in [42].

For fusion parameter estimation (Section 4.6), we used all the text lines of the 4076 training text pages of 816 writers. The regularization coefficient was set as $\gamma = 1$ (a value in [0,1] was found to be not influential, nevertheless).

For evaluating the word spotting performance, we selected 60,000 frequently used words from the corpus collected by Sogou Labs[1] as query words. The query words maybe either present or absent in the test documents (an absent query word may result in false positives). They include 39,057 two-character words, 9975 three-character words, 9451 four-character words, and 1517 words containing more than four characters. Word spotting was performed on segmented text lines provided by the database.

Algorithm 1. Word spotting by character-synchronous search from a start segment s_p.

Input: A sequence of primitive segments $< s_1 \cdots s_n >$ in a document,
A query word represented by $W = c_1 \cdots c_L$.
Output: A partial sequence of primitive segments $< s_p \cdots s_q >$ matched with W.
Start: $i = 1$, $j_1 = p$, $PM = NULL$.
Character_Procedure
for $k_i = 1$ **to** K **do**
  Generate candidate character $I_{c_i} = s_{j_i} \cdots s_{j_i+k_i-1}$,
  if $g_c(c_i, I_{c_i}) \geq T_c$ **and** $g_W(c_1 \cdots c_i, s_{j_1} \cdots s_{j_i+k_i-1}) \geq T_W$ **then**
    save partial match in PM.
  end if
end for
if $PM = NULL$ **then**
  goto **End Character_Procedure**.
else
  for all partial match $pm = (c_1 \cdots c_i, s_{j_1} \cdots s_{j_i+k_i-1})$ **do**
    if $i < L$ **then**
      remove pm from PM,
      $i = i + 1, j_i = j_i + k_i$,
      goto **Character_Procedure**
    end if
  end for
end if
End Character_Procedure
ExtendedWord_Procedure
for all word match $pm = (c_1 \cdots c_L, s_{j_1} \cdots s_{j_L+k_L-1})$ **do**
  Extend the word alignment to get A^e,
  if $g_{W^e}(A^e) \geqslant T_{W^e}$ **then**
    output spotted word instance $s_{j_1} \cdots s_{j_L+k_L-1}$.
  end if
end for
End ExtendedWord_Procedure

[1] http://www.sogou.com/labs/dl/w.html.

To judge whether a spotted word instance is correct or not, we calculate the overlap ratio of area between the spotted instance and the ground-truthed word location. We denote the bounding boxes of ground-truthed word and spotted word by g and d, respectively, the overlapping area between them by ov, and the overlap ratio is calculated by

$$r = \frac{ov}{g + d - ov}. \tag{25}$$

When the overlap ratio is over a threshold (empirically set as 0.9), the spotted instance is regarded as a correct detect (true positive, TP), otherwise a false positive (FP). If a ground-truthed word in the document is not spotted, it results in a false negative (FN). Based on these counts, the metrics recall rate (R), precision (P) and F-measure ($F1$) are then calculated.

6.1. Character detection performance

For evaluating the effects of character classifier and single-character geometric context, we give the results of single character detection. As introduced in Section 4.1, we used two types of classifiers: one-versus-all prototype classifier and linear SVM classifier. We tested three prototype classifiers with 1/3/5 prototypes per class, denoted as "Prototype(1)", "Prototype(3)" and "Prototype(5)", respectively. We evaluated several variations of character detection model: (a) using character classifier (f_0) only; (b) fusing f_0 and class-dependent single-character geometry (f_1); and (c) fusing f_0, f_1, and class-independent single-character geometry (f_2). The 7356 classes defined by the character classifier were used as queries for detecting from the test documents. Table 1 shows the recall/precision rates under the maximum F-measure over various thresholds, and Fig. 5 shows the precision–recall curves. Because confidence transformation turns out to improve the retrieval performance compared to that without CT (shown in Table 1), we show only the curves of character detection with CT.

Table 1 shows that for the prototype classifier, increasing the number of prototypes helps improve the detection performance. However, the linear SVM classifier even outperforms the prototype classifier with multiple prototypes per class. For all the classifiers, fusion of geometric models (f_1 and f_2) improves the detection rates significantly (e.g., the F-measure is improved from 57.86% to 80.91% for Prototype(1) classifier, and from 75.35% to 88.98% for SVM classifier),

Table 1
Character detection rates (%) using different classifiers. CT means confidence transformation for classifier and geometric model outputs.

Character classifier	Method	Without CT			With CT		
		R	P	F1	R	P	F1
Prototype(1)	f_0	50.47	59.37	54.56	53.89	62.47	57.86
	f_0+f_1	71.43	79.23	75.13	73.65	80.79	77.05
	f_0+f_2	67.89	74.69	71.13	70.56	76.89	73.59
	$f_0+f_1+f_2$	74.57	83.71	78.88	**76.74**	**85.56**	**80.91**
Prototype(3)	f_0	53.69	61.87	57.49	56.97	64.58	60.54
	f_0+f_1	74.84	81.57	78.06	76.23	82.96	79.45
	f_0+f_2	71.34	77.26	74.18	72.58	78.63	75.48
	$f_0+f_1+f_2$	77.59	85.31	81.27	**78.46**	**86.50**	**82.29**
Prototype(5)	f_0	54.23	62.39	58.02	57.69	65.13	61.18
	f_0+f_1	75.83	81.97	78.78	76.53	83.46	79.84
	f_0+f_2	72.41	78.39	75.28	73.69	79.76	76.60
	$f_0+f_1+f_2$	77.63	84.57	80.95	**78.79**	**86.73**	**82.57**
SVM	f_0	69.43	74.74	71.99	72.03	78.98	75.35
	f_0+f_1	79.38	84.75	81.98	82.15	86.76	84.39
	f_0+f_2	74.85	79.46	77.09	76.16	82.97	79.42
	$f_0+f_1+f_2$	84.47	89.25	86.79	**87.62**	**90.38**	**88.98**

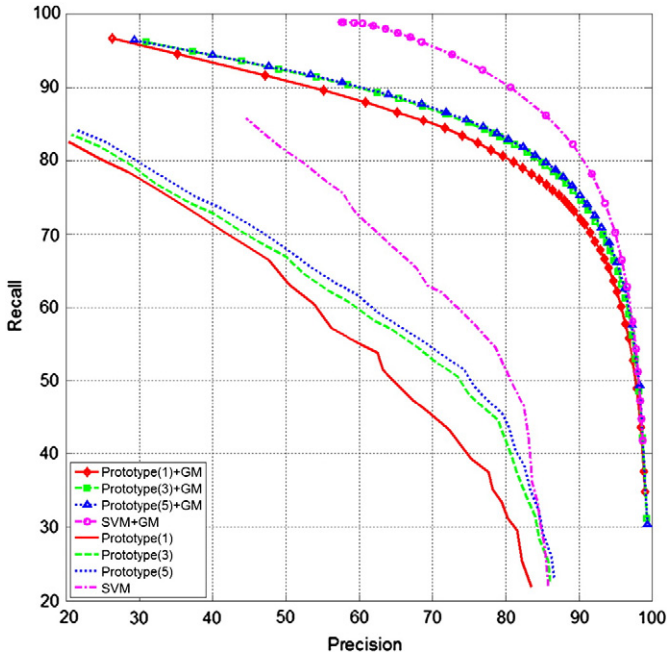

Fig. 5. Precision–recall curves of character detection using different classifiers, GM means geometric models ($f_1 + f_2$ in the case of character detection).

and the class-dependent geometry (f_1) is more effective than the class-independent geometry (f_2). The precision–recall curves in Fig. 5 confirm the benefits of geometric models and that the SVM classifier outperforms the prototype classifier.

6.2. Word spotting performance

For word spotting, the between-character geometric models are available. We show the results using the fused word model (Eq. (4)) using all the geometric models ($f_1 \sim f_4$) in Table 2. We can see that the fusion of geometric models also improves the word spotting performance significantly. For a fixed word length, the comparison of different prototype classifiers shows that the performance of the one-versus-all prototype classifier is improved with increased number of prototypes per class, and the SVM classifier performs better than prototype classifiers for two-character and three-character words. Spotting longer words results in higher precision and recall rates than that of short words. This is because with more characters in longer words, the word similarity incorporates more contextual information.

The extended word model with external context (Eq. (10)) can be dichotomized into one with external geometric model (GM) only and one with both external GM and linguistic model (LM, character bi-gram). According to the marginalization rule, it can be dichotomized into one of summation (Eq. (17)) and one of maximization (Eq. (18)). We observed that the two marginalization rules perform comparably, for saving space, we only show the results of maximization marginalization as in Table 3.

Comparing the performance of extended word model with external GM (Table 3) and that of word model without external context (Table 2), it is evident that the external GM improves the performance. For example, in spotting four-character words, the F-measure of Prototype(1) classifier is improved from 76.23% to 78.87%, and the F-measure of SVM classifier is improved from 79.35% to 81.98%.

Comparing the extended word model with both external GM and LM and the one with external GM only (Table 3), it is shown that the external LM improves the spotting performance significantly. For example, in spotting four-character words, the F-measure of Prototype(1) classifier is improved from 78.87% to 84.21%, and the F-measure of SVM classifier is improved from 81.98% to 87.38%.

6.3. Comparison with transcription-based retrieval

To justify the advantages of word model-based word spotting over text line recognition (transcription)-based word search, we conducted experiments to compare their performance. We implemented text line recognition using the method of Wang, Yin and Liu [42]. For fair comparison, both the word model and the text line recognizer use the same character classifier, same geometric models (GMs), and same language model (LM, character bi-gram). We also compared contextual word model and text line recognizer without LM. Comparison of retrieval without LM is meaningful because in some applications, the language domain is unknown or largely variable which makes the modeling of linguistic context difficult. The results of word spotting and transcription-based search for two-character words with LM are shown in Fig. 6. The figure of transcription-based search without LM is omitted for saving space, but it shows similar tendency as that of transcription with LM.

Fig. 6 shows that the SVM classifier yields inferior performance in transcription-based word retrieval compared to the prototype classifiers, though it performs better in word model-based spotting. This is due to the low accuracy of text line recognition using linear SVM classifier for character classification. The prototype classifier, also trained under the one-versus-all objective, performs much better than the SVM classifier in text line recognition. Particularly, the Prototype(5) classifier performs best in transcription-based word search.

The text line recognition accuracy relies on the character classification accuracy and cumulated accuracy of multiple ranks. We tested the four classifiers on the segmented characters of the test documents. The top rank accuracies of Prototype(1), Prototype(3), Prototype(5) and SVM are 84.69%, 85.54%, 85.87% and 84.76%. The

Table 2
Word spotting performance of variable word lengths with and without GM.

Word length	Character classifier	Without GM			With GM		
		R	P	F1	R	P	F1
2	Prototype(1)	57.67	72.89	64.39	62.37	78.23	69.41
	Prototype(3)	58.79	73.41	65.29	63.03	79.14	70.17
	Prototype(5)	59.13	73.95	65.71	63.57	**79.68**	**70.72**
	SVM	60.59	72.38	65.96	**64.81**	76.71	70.26
3	Prototype(1)	61.98	71.47	66.39	67.98	83.57	74.97
	Prototype(3)	62.73	71.98	67.04	68.57	84.97	75.89
	Prototype(5)	63.59	72.29	67.66	69.38	85.64	76.66
	SVM	64.28	72.58	68.18	**71.48**	**86.05**	**78.09**
4	Prototype(1)	62.97	71.59	67.00	69.28	84.74	76.23
	Prototype(3)	63.58	72.03	67.54	70.19	85.27	77.00
	Prototype(5)	64.17	72.98	68.29	71.25	**85.79**	77.85
	SVM	66.32	74.56	70.20	**74.32**	85.12	**79.35**

Table 3
Word spotting performance of variable word lengths using extended word model of maximization marginalization.

Word length	Character classifier	External GM			External GM + LM		
		R	P	F1	R	P	F1
2	Prototype(1)	63.59	82.68	71.89	68.94	88.67	77.57
	Prototype(3)	63.92	83.57	72.44	69.15	89.46	78.00
	Prototype(5)	64.03	84.12	72.71	69.36	90.13	78.39
	SVM	64.15	85.89	73.44	**69.57**	**91.79**	**79.15**
3	Prototype(1)	71.31	85.87	77.92	76.28	91.49	83.20
	Prototype(3)	72.09	86.69	78.72	77.09	92.34	84.03
	Prototype(5)	72.63	87.25	79.27	77.64	92.87	84.57
	SVM	73.17	87.97	79.89	**78.37**	**93.62**	**85.32**
4	Prototype(1)	72.03	87.14	78.87	76.98	92.95	84.21
	Prototype(3)	73.15	87.46	79.67	77.89	93.77	85.10
	Prototype(5)	74.03	87.69	80.28	78.56	94.25	85.69
	SVM	76.85	87.85	81.98	**81.79**	**93.79**	**87.38**

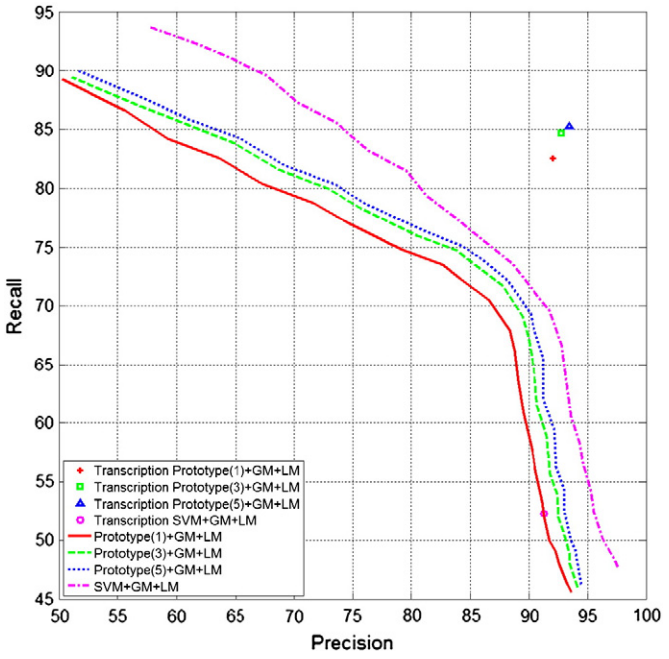

Fig. 6. Performance of model-based word spotting and transcription-based retrieval with LM for two-character words.

cumulated accuracies of 20 top ranks are 96.84%, 97.71%, 97.93% and 88.48%. The low cumulated accuracy of SVM leads to low performance of text line recognition. Increasing the number of candidate classes (top ranks) for the SVM classifier gives only slight improvement of text line recognition accuracy because a large number of candidate classes complicates the confusion of candidate segmentation–recognition paths.

Transcription-based retrieval gives one unique point of precision–recall tradeoff since the text line recognizer gives unique text output. On the other hand, the model-based word spotting provides flexible precision–recall tradeoff by tuning the threshold of word matching score. At certain threshold, word model-based spotting can yield much higher recall rate than transcription-based search, as is evident in Fig. 6.

Table 4 shows the precision and recall rates of transcription-based search for words of variable lengths. Transcription-based word search yields high precision but low recall rates. This is because many characters in handwritten documents are misrecognized. The words with misrecognized characters cannot be retrieved, thus lead to low recall rate. On the other hand, the words retrieved in the recognition output are more likely to be correct, thus lead to high precision. Moreover, the recall rate decreases when the length of query word increases, unlike

Table 4
Transcription-based word spotting results (%) (with GM by default).

Word length	Character classifier	Without LM (%)			With LM (%)		
		R	P	F1	R	P	F1
2	Prototype(1)	63.51	92.41	75.28	82.53	92.07	87.04
	Prototype(3)	65.47	93.18	76.91	84.67	92.78	88.54
	Prototype(5)	66.38	93.76	77.73	**85.23**	**93.45**	**89.15**
	SVM	52.29	91.30	66.50	61.07	85.18	71.13
3	Prototype(1)	53.48	93.87	68.14	76.58	93.17	84.06
	Prototype(3)	55.64	94.25	69.97	78.43	93.97	85.50
	Prototype(5)	56.37	94.87	70.72	**79.14**	**94.57**	**86.17**
	SVM	41.49	92.66	57.31	51.52	91.82	66.01
4	Prototype(1)	44.10	91.87	59.59	72.83	91.37	81.05
	Prototype(3)	46.37	92.35	61.74	74.79	92.46	82.69
	Prototype(5)	47.58	93.08	62.97	**75.38**	**93.24**	**83.36**
	SVM	32.26	91.10	47.65	42.59	90.60	57.94

Table 5
Average time (ms) for spotting a query word from a document image.

Character classifier	Word length		
	2	3	4
Prototype(1)	75	96	122
Prototype(3)	86	108	138
Prototype(5)	108	124	159
SVM	63	89	120

that in word model-based spotting, the recall rate increases with word length (cf. Table 3). Comparing the best performance of word model extended with external GM (left in Table 3, best results are given by SVM) with that of the transcription-base search with GM (left in Table 4, best results are given by Prototype(5) classifier), we can see that for two-character words, transcription-based search performs better; for words of three-character or longer, word model-based spotting yields significantly higher recall rate but slightly lower precision. When comparing the best performance of two methods with GM and LM (right in Tables 3 and 4), word model-based spotting is still inferior to transcription-based search for two-character words. For three-character words, the best results of two methods are comparable; for four-character words, word model-based spotting gives both higher precision and recall rates.

6.4. Computational complexity

For computational efficiency, we use classifiers with low complexity: linear SVM and prototype classifiers. The linear SVM classifier stores the weight vectors of one per class. The prototype classifier has comparable complexity with linear SVM when each class has one prototype, and the complexity is proportional to the prototype number. In our system, the feature vector of candidate character is reduced to 160D by FLDA, and the number of classes is 7356. So, the linear SVM classifier has $160 \times 7356 \simeq 1.177\ M$ parameters, consuming 4.7 MB memory if using 4 bytes per number. Compared to the classifier parameter storage, the class-dependent (6-class and 36-class) and class-independent (2-class) geometric models are very small. The memory size of the bi-gram language model is proportional to the square of character class number (7356). Exploiting the sparsity of bi-grams, the language model actually consumes 6.19 MB memory.

In keyword spotting from document image, the processing time is mainly due to candidate character pre-processing, feature extraction, and character similarity and word similarity computations. On a candidate character, the average times of pre-processing and feature extraction are 0.011 ms and 0.014 ms, respectively. The average times of similarity computation for four classifiers, Prototype(1/3/5) and SVM, are 0.009 ms, 0.012 ms, 0.015 ms[2] and 0.008 ms, respectively. We implemented the algorithms on a personal computer with CPU of Intel XeonE3-1230 3.2 GHz.

The keyword spotting time depends on the query word length and document length as well. In our database, each document has around 300 characters on average. By concatenating consecutive primitive segments into candidate characters after over-segmentation, the number of candidate characters is about three times as many as the number of true characters. Table 5 gives the average times for spotting query words of variable length from a document image using the proposed contextual word model with different classifiers. For words of two or three characters, the spotting time using Prototype(1) or linear SVM is less than 100 ms. This speed is acceptable for real-time retrieval from a set of less than 100 documents. For retrieving a larger

[2] When there are multiple prototypes per class, the time for computing the minimum distance per class is not proportional to the prototype number.

Table 6
Classification times and accuracies of small character set.

Classifier	CV acc (%)	Train time (s)	Test acc (%)	Test time (ms)
Prototype(1)	94.42	108	93.87	0.21
Prototype(3)	95.00	248	94.40	0.29
Prototype(5)	95.22	383	94.60	0.37
SVM-linear	94.23	217	93.61	0.21
SVM-poly	96.84	37,140	96.15	20.03
SVM-rbf	96.90	48,557	96.19	23.24

document set, real-time search can be enabled by offline computing of character similarities and storing in index files.

6.5. Experiments with small category set

Due to the large number of character classes in Chinese (7356 classes in our experiments), we only used classifiers of low complexity in the above experiments. For comparing more classifiers, including some of high complexity, we conducted experiments on a small character set (small category), 100 classes of most frequent Chinese characters (according to the character frequencies in the corpus of Sogou Labs). Besides the enabling of evaluating various classifiers, evaluation of small category classifiers is meaningful because in some applications, the users are interested in a small set of query words.

In addition to the one-versus-all trained prototype classifier and linear SVM, we also evaluate two nonlinear SVM classifiers, with 4th order polynomial kernel and radial basis function (RBF) kernel, referred to as SVM-poly and SVM-rbf, respectively. For the polynomial kernel, the input vectors are empirically re-scaled such that the average self-inner product of training vectors is 2. For the RBF kernel, the variance parameter is set as 0.25 times the average square Euclidean distance to class mean of training samples. The SVM classifiers were trained using the successive over-relaxation (SOR) algorithm of Mangasarian and Musicant [38], with the upper bound of multiplier set as 1 and 10 for SVM-poly and SVM-rbf, respectively.

For 100-class classification, we use the samples in datasets DB1.0, DB1.1, DB2.0 and DB2.1 in CASIA-HWDB. DB1.0 and DB1.1 contain isolated character images, DB2.0 and DB2.1 contain handwritten text pages. DB1.0 and DB2.0 were produced by 420 writers, and DB1.1 and DB 2.1 were produced by 300 writers. In this experiment, we do not use the datasets DB1.2 and DB2.2 (produced by another 300 writers) because the DB1.2 does not have samples of high-frequency characters. Each dataset was partitioned into a training subset of 4/5 writers and a test subset of 1/5 writers. The training subsets of four datasets have 289,309 character samples falling in the 100 high-frequency classes. We selected 100,000 training samples from this set randomly with same percentage from all classes for reducing the training time of nonlinear SVM classifiers. The 100-class character samples in handwritten text datasets DB2.0 and DB2.1, 57,854 in total, were used as test set. For each training or test sample, the extracted 512-D feature vector (Section 4.1) was reduced to 99-D by FLDA.

The keyword spotting performance of 100-class classifiers was evaluated on the test data of datasets DB2.0 and DB2.1, in total of 715 pages. The query words are those high-frequency words in the corpus of Sogou Labs composed of the 100 character classes. There are 2936 such words, including 1922 two-character words, 713 three-character words and 280 four-character words.

We compare the classifiers in respect of isolated character classification accuracy as well as word spotting performance. For isolated character classification, we report the 5-fold cross validation (CV) accuracy on the training set as well as the accuracy on the test set. For cross validation, the training set was randomly partitioned into five subsets with each class uniformly distributed in the subsets. The classification times and accuracies are shown in Table 6. Comparing the training time and average test time, it is evident that the nonlinear SVM classifiers have far higher complexity than the linear SVM and prototype classifiers, but give higher classification accuracies.

The word spotting performance of 100-class (small category) classifiers is shown in Table 7. For comparison with the spotting results of large category (7356 classes), we give the results of spotting variable-length words, with internal GM (geometric models), external GM and external GM + LM (language model). Comparing the results of spotting with internal GM in Table 7 with the results of large category in Table 2, we can see that small category classifiers give slightly higher spotting performance. This is because small category classifiers have higher discrimination accuracies. It is noteworthy that in small category setting, the linear SVM classifier yields inferior spotting performance than the prototype classifier, while in large category setting, it outperforms the prototype classifier. This can be explained that when training with small category samples, the linear SVM classifier has weaker rejection ability to out-of-class samples (there are many characters beyond the 100 classes in the test documents). The nonlinear SVM classifiers give higher spotting performance than the prototype classifier and linear SVM owing to the superior discrimination ability of nonlinear SVM.

Table 7
Word spotting performance of small character set using contextual word model.

Word length	Character classifier	Internal GM			External GM			External GM + LM		
		R	P	F1	R	P	F1	R	P	F1
2	Prototype(1)	66.28	81.06	72.93	68.79	81.86	74.76	70.81	83.01	76.43
	Prototype(3)	67.83	82.47	74.44	70.08	83.47	76.19	72.53	84.36	78.00
	Prototype(5)	68.37	83.51	75.17	70.89	84.51	77.10	73.15	85.53	78.86
	SVM-linear	64.14	84.63	72.97	66.89	85.47	75.05	68.91	86.67	76.78
	SVM-poly	70.34	86.32	77.51	72.98	87.45	79.56	75.06	88.57	81.26
	SVM-rbf	72.29	87.59	79.21	74.63	88.36	80.92	76.21	89.63	82.38
3	Prototype(1)	71.78	85.64	78.10	74.25	86.87	80.07	76.04	87.91	81.55
	Prototype(3)	72.63	86.61	79.01	75.36	87.83	81.12	77.42	88.96	82.79
	Prototype(5)	73.59	87.23	79.83	76.14	88.64	81.92	78.23	89.78	83.61
	SVM-linear	69.23	88.57	77.71	71.76	89.37	79.60	73.46	90.48	81.09
	SVM-poly	75.53	88.91	81.68	78.16	90.56	83.90	80.27	91.56	85.54
	SVM-rbf	77.64	89.73	83.25	80.11	90.98	85.20	82.19	92.34	86.97
4	Prototype(1)	72.93	86.95	79.33	75.64	88.47	81.55	77.56	89.63	83.16
	Prototype(3)	73.45	87.61	79.91	77.59	89.63	83.18	79.26	90.54	84.53
	Prototype(5)	74.56	88.65	81.00	78.46	90.13	83.89	80.65	91.27	85.63
	SVM-linear	69.45	90.63	78.64	72.45	91.56	80.89	74.68	92.46	82.62
	SVM-poly	76.73	92.58	83.91	79.46	93.29	85.82	81.53	94.58	87.57
	SVM-rbf	78.46	93.29	85.23	81.06	94.32	87.19	83.27	95.02	88.76

Comparing the results of spotting with external GM in Table 7 with the results of large category in Table 3, we can see that in the case of spotting with external GM, small category classifiers give higher spotting performance. However, in the small category setting, the adding of external LM does not improve the spotting performance as remarkably as that of large category setting. This is because many external characters adjacent to the queried words in the test documents are beyond the 100 classes, and so, cannot be recognized. Without recognizing the external characters, the effect of linguistic dependency is limited.

It is noteworthy that the number of classes in classifier does not affect the computation time of word spotting, since the word similarity involves only the scores of the classes in the query word. Hence, the spotting times of small category setting remain the same as those in Table 5. The spotting time of nonlinear SVM is about 20 times as that of linear SVM. The independence of spotting time on category size and the promising spotting performance of large category classifiers with external GM + LM (Table 3 compared to Table 7) indicate that word spotting using large category classifiers is preferable.

With small category classifier, document transcription (text line recognition) is not possible. Hence, the comparison of word model-based spotting with transcription-based search is not available in this case.

7. Conclusion

We proposed a keyword spotting method for off-line handwritten Chinese documents using contextual word model. For computing the similarity between text query word and candidate word image (sub-sequence of primitive segments), the contextual word model fuses the character classification score, geometric and linguistic contexts within and out of word. Experimental results show that one-versus-all prototype classifier and linear SVM are suitable for word model-based spotting. The benefits of geometric and linguistic contexts, particularly the dependency with external adjacent characters, were justified. Comparing with transcription-based text search, the proposed method can give flexibly higher recall rate, and when the query word is four-character or longer, it can give both higher precision and recall rate. Experiments of small category classifiers show the superiority of nonlinear SVM classifiers despite their high computational complexity. Our future works aim to further improve the spotting performance by combining different classifiers and combining word model matching with transcription-based search.

Acknowledgement

This work was supported in part by the National Basic Research Program of China (973 Program) Grant 2012CB316302, the National Natural Science Foundation of China (NSFC) Grants 60933010 and 60825301.

References

[1] D. Doermann, The indexing and retrieval of document images: a survey, Comput. Vis. Image Underst. 70 (3) (1998) 287–298.
[2] M. Mitra, B. Chaudhuri, Information retrieval from documents: a survey, Inf. Retr. 2 (2–3) (2004) 141–163.
[3] A. Graves, M. Liwicki, S. Fernandez, R. Bertonami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868.
[4] T.M. Rath, R. Manmatha, Word image matching using dynamic time warping, CVPR, vol. 2, 2003, pp. 521–527.
[5] B. Zhang, S.N. Srihari, C. Huang, Word image retrieval using binary features, Document Recognition and Retrieval XI, 2004.
[6] A.L. Kesidis, E. Galiotou, B. Gatos, I. Pratikakis, A word spotting framework for historical machine-printed documents, Int. J. Doc. Anal. Recognit. 14 (2) (2011) 131–144.
[7] E. Saykol, A.K. Sinop, U. Gudukbay, O. Ulusoy, A.E. Cetin, Content-based retrieval of historical Ottoman documents stored as textual images, IEEE Trans. Image Process. 13 (3) (2004) 314–325.
[8] S. Lu, L. Li, C.L. Tan, Document image retrieval through word shape coding, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (2008) 1913–1918.
[9] K. Khurshid, C. Faure, N. Vincent, Word spotting in historical printed documents using shape and sequence comparisons, Pattern Recogn. 45 (7) (2012) 2598–2609.
[10] Y. Leydier, F. Lebourgeois, H. Emptoz, Text search for medieval manuscript images, Pattern Recogn. 40 (12) (2007) 3552–3567.
[11] T.M. Rath, R. Manmatha, V. Lavrenko, A search engine for historical manuscript images, 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004.
[12] S.S. Kuo, O.E. Agazzi, Keyword spotting in poorly printed documents using pseudo 2-d hidden Markov models, IEEE Trans. Pattern Anal. Mach. Intell. 16 (8) (1994) 842–848.
[13] J.A. Rodriguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden Markov models and universal vocabularies, Pattern Recogn. 42 (9) (2009) 2106–2116.
[14] N.R. Howe, S. Feng, R. Manmatha, Finding words in alphabet soup: inference on freeform character recognition for historical scripts, Pattern Recogn. 42 (12) (2009) 3338–3347.
[15] H. Cao, A. Bhardwaj, V. Govindaraju, A probabilistic method for keyword retrieval in handwritten document images, Pattern Recogn. 42 (12) (2009) 3374–3382.
[16] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recogn. 33 (7) (2012) 934–942.
[17] V. Frinken, A. Fischer, R. Manmatham, H. Bunke, A novel word spotting method based on recurrent neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 211–224.
[18] Y. Lu, C.L. Tan, Chinese word searching in imaged documents, Int. J. Pattern Recognit. Artif. Intell. 18 (2) (2004) 229–246.
[19] A. Kolcz, A line-oriented approach to word spotting in handwritten documents, Pattern. Anal. Applic. 3 (2) (2000) 153–168.
[20] Y. He, Z. Jiang, B. Liu, H. Zhao, Content-based indexing and retrieval method of Chinese document images, Proc. 5th ICDAR, 1999, pp. 685–688.
[21] Y. Zhuang, X. Zhang, J. Wu, X. Lu, Retrieval of Chinese calligraphic character image, Advances in Multimedia Information Processing - PCM, 2004, pp. 17–24.
[22] S. Senda, M. Minoh, K. Ikeda, Document image retrieval system using character candidates generated by character recognition process, Proc. 2nd ICDAR, 1993, pp. 541–546.
[23] T. Kameshiro, T. Hirano, Y. Okada, E. Yoda, A document image retrieval method tolerating recognition and segmentation errors of OCR using shape-feature and multiple candidates, Proc. 5th ICDAR, 1999, pp. 681–684.
[24] T. Kameshiro, T. Hirano, Y. Okada, A document retrieval method from handwritten characters based on OCR and character shape information, Proc. 6th ICDAR, 2001, pp. 597–601.
[25] T. Nagasaki, T. Takahashi, K. Marukawa, Document retrieval system tolerant of segmentation errors of document images, Proc. 9th IWFHR, 2004, pp. 280–285.
[26] H. Oda, A. Kitadai, M. Onuma, M. Nakagawa, A search method for on-line handwritten text employing writing-box-free handwriting recognition, Proc. 9th IWFHR, 2004, pp. 545–550.
[27] C.-L. Liu, M. Koga, H. Fujisawa, Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading, IEEE Trans. Pattern Anal. Mach. Intell. 24 (11) (2002) 1425–1437.
[28] L. Huang, F. Yin, Q.-H. Chen, C.-L. Liu, Keyword spotting in offline Chinese handwritten documents using a statistical model, Proc. 11th ICDAR, 2011, pp. 78–82.
[29] D. Lopresti, A. Tomkins, On the searchability of electronic ink, Proc. 4th IWFHR, 1994, pp. 156–165.
[30] C.V. Jawahar, A. Balasubramanian, M. Meshesha, A.M. Namboodiri, Retrieval of online handwriting by synthesis and matching, Pattern Recogn. 42 (7) (2009) 1445–1457.
[31] C.-L. Liu, One-vs-all training of prototype classifier for pattern classification and retrieval, Proc. 20th ICPR, 2010, pp. 3328–3331.
[32] H. Zhang, D.-H. Wang, C.-L. Liu, Keyword spotting from online Chinese handwritten documents using one-vs-all trained character classifier, Proc. 12th ICFHR, 2010, pp. 271–276.
[33] X.-B. Jin, C.-L. Liu, X. Hou, Regularized margin-based conditional log-likelihood loss for prototype learning, Pattern Recogn. 43 (7) (2009) 2428–2438.
[34] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New Work, 1995.
[35] F. Yin, C.-L. Liu, Handwritten text line segmentation by clustering with distance metric learning, Pattern Recogn. 42 (12) (2009) 3146–3157.
[36] C.-L. Liu, Normalization-cooperated gradient feature extraction for handwritten character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 29 (8) (2007) 1465–1469.
[37] C.-L. Liu, K. Marukawa, Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition, Pattern Recogn. 38 (12) (2005) 2242–2255.
[38] O.L. Mangasarian, D.R. Musicant, Successive over-relaxation for support vector machines, IEEE Trans. Neural Netw. 10 (5) (1999) 1032–1037.
[39] F. Yin, Q.-F. Wang, C.-L. Liu, Integrating geometric context for text alignment of handwritten Chinese documents, Proc. 12th ICFHR, 2010, pp. 7–12.
[40] L. Gillick, Y. Ito, J. Young, A probabilistic approach to confidence estimation and evaluation, Proc. ICASSP, vol. 2, 1997, pp. 879–882.
[41] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang, CASIA online and offline Chinese handwriting databases, Proc. 11th ICDAR, 2011, pp. 37–41.
[42] Q.-F. Wang, F. Yin, C.-L. Liu, Handwritten Chinese text recognition by integrating multiple contexts, IEEE Trans. Pattern Anal. Mach. Intell. 34 (8) (2012) 1469–1481.