Engineering Applications of Artificial Intelligence 58 (2017) 49–61

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Keyword spotting in handwritten chinese documents using semi-markov conditional random fields

Heng Zhang[a,*], Xiang-Dong Zhou[b], Cheng-Lin Liu[c]

[a] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China
[b] Intelligent Media Technique Research Center, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, PR China
[c] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China

ARTICLE INFO

Keywords:
Online handwritten Chinese documents
Semi-Markov conditional random fields
Keyword spotting
Proxy-character driven search

ABSTRACT

This paper proposes a document indexing method for keyword spotting based on semi-Markov conditional random fields (semi-CRFs), which provide a theoretical framework for fusing the information of different contexts. The candidate segmentation-recognition lattice is first augmented based on the linguistic context to improve recognition results. For fast retrieval and to save storage space, the lattice is then purged by a forward-backward pruning procedure. In the reduced lattice, we estimate character similarity scores based on the semi-CRF model. The parameters of semi-CRF model are estimated using a binary classification objective, i.e., the cross-entropy (CE) to discriminate candidate characters in the lattice. To locate mis-recognized character instances in the lattice, we use confusing similar characters as proxies and search for proxy-characters in the index file. The proxy-character driven search can significantly improve the performance compared with our previous character-synchronous dynamic search (CSDS) method. Experimental results on the online handwriting database CASIA-OLHWDB justify the effectiveness of the proposed method.

1. Introduction

With the increasing use of pen-based input devices and user-interfaces, more and more research attentions have been paid on document analysis techniques including text segmentation, recognition and retrieval. In spite of the great progress on handwritten text recognition (Plamondon and Srihari, 2000; Graves et al., 2009), remaining recognition errors can still prevent locating keywords. Keyword spotting (Manmatha et al., 1996; Frinken et al., 2012; Fischer et al., 2012) is to locate words or phrases in the document without the need of accurate handwriting recognition. By computing a similarity measure between the query word and a segmented candidate in the document, the user can adjust the threshold to balance recall and precision rates for fulfilling different needs. The application background of keyword spotting is the retrieval of handwritten pages such as notes, bank checks, government files and historical documents.

For fast retrieval of documents from large database, it is necessary to build and store an index file beforehand, on which the spotting algorithm is run and gives spotting results for a query word (Zhang et al., 2014). The state-of-art for English handwriting recognition, i.e., long-short term memory recurrent neural network (LSTM-RNN) has been used for Chinese off-line handwritten text recognition (Messina and Louradour, 2015) but its performance is still inferior to methods based on character over-segmentation and classification (Wang et al., 2012, 2014). Therefore, for keyword spotting in handwritten Chinese documents, we build the index file based on the segmentation-recognition framework (Zhang et al., 2014, 2013a; Huang et al., 2013). In the document, each text line is first over-segmented into a sequence of components according to the overlapping between strokes, with the hope that each component is a character or part of a character. Subject to constraints of character width, consecutive components are combined to generate candidate characters, which constitute the segmentation candidate lattice. On assigning each candidate character a number of candidate classes using a character classifier, the segmentation-recognition candidate lattice (referred to as lattice for brevity) is constructed. Each path in the lattice corresponds to a segmentation-recognition hypothesis of the text line. Each character-label pair (a candidate character coupled with one of its candidate labels) in the lattice is referred to as an edge. The character similarity scores between each candidate character and its candidate classes, also referred to as edge scores, are calculated and stored. In text search, the query word is matched with sequences of candidate characters (partial paths in the candidate lattice) starting from each component in the lattice (Zhang et al., 2013a). The word similarity is obtained by

* Corresponding author.
E-mail addresses: heng.zhang@ia.ac.cn (H. Zhang), zhouxiangdong@cigit.ac.cn (X.-D. Zhou), liucl@nlpr.ia.ac.cn (C.-L. Liu).

http://dx.doi.org/10.1016/j.engappai.2016.11.006
Received 28 September 2015; Received in revised form 5 September 2016; Accepted 21 November 2016
Available online 28 November 2016
0952-1976/ © 2016 Elsevier Ltd. All rights reserved.

combining the character similarity scores (edge scores). When the word similarity is greater than a threshold, a word instance is located in the document. So, the similarity measure is critical to keyword spotting. For keyword spotting from large databases of multi-writer or writer independent documents, to alleviate the effects of character shape variation, edge scores are usually given by a character classifier (Oda et al., 2004; Zhang et al., 2013a; Cheng et al., 2013). Ideally, the score of a true character should be higher than any imposters. To improve the discriminability, additional information (Huang et al., 2013), such as geometric and linguistic contexts, are incorporated when calculating edge scores.

In this paper, we propose an indexing method for keyword spotting using semi-CRFs (Zhou et al., 2013), which are probabilistic graphical models defined in the candidate lattice and provide a theoretical framework for fusing the information of different contexts. With this model, we first augment the original lattice to supplement candidate character classes and then reduce the lattice complexity by a forward-backward pruning procedure used in Zhou et al. (2013) which avoids breaking high-probability paths, and edge scores are derived from the marginal probabilities of edges. The semi-CRF model is derived from the Chinese handwriting recognition system in Zhou et al. (2013), but the proposed method is different from text line recognition. Handwritten text recognition only retains the 1-best recognition result which is limited for retrieval. Keyword spotting is performed in the lattice which contains alternative candidate characters/words, such that more instances can be located than 1-best list and the user can adjust the threshold to balance recall and precision rates. The relations between keyword spotting and handwriting recognition have been discussed in Frinken et al. (2012). Different from handwriting recognition, keyword spotting is binary classification of edges in the lattice desiring high similarities to target characters and low scores to all the others. Hence, we propose a binary classification objective, i.e., the cross-entropy (CE) to optimize semi-CRF parameters. Besides, to enhance the recognition performance of the semi-CRF model, we propose a proxy-character driven search algorithm to locate mis-recognized character instances in the lattice. Confusing similar characters can be used as proxies to search in the index file so that the candidate character mis-recognized as its proxies can be matched. Compared with the traditional character-synchronous dynamic search, the use of proxy-characters can improve the keyword spotting performance significantly. Keyword spotting from Japanese handwritten documents (Cheng et al., 2013; Oda et al., 2004) is also performed by scoring candidate characters in the lattice and locate instances by the character-synchronous search. Here, we have paid more attentions to character scoring and the search method to improve the word matching accuracy and search efficiency.

The remainder of this paper is organized as follows: Section 2 reviews related works. Section 3 gives the overview of our keyword spotting system. Section 4 details the index generation based on the semi-CRF model. Section 6 introduces the proxy-character driven search algorithm. Section 6 presents our experimental results on an online handwriting database CASIA-OLHWDB and Section 7 draws concluding remarks.

2. Related work

Keyword spotting was originally formulated as detecting words or phrases in speech (Myers et al., 1981), and extended to locate words in printed text documents (Kuo and Agazzi, 1994) a decade later. This technique was first performed on online handwriting in Lopresti and Tomkins (1994) for annotation retrieval and on handwritten document images in Manmatha et al. (1996). With regard to word similarity scoring techniques, keyword spotting methods can be categorized into two groups: word shape matching (Jain and Namboodiri, 2003; Jawahar et al., 2009; Rodriguez-Serrano and Perronnin, 2012; Sarkar, 2013) and model-based scoring (Van der Zant et al., 2008; Rodriguez-Serrano and Perronnin, 2009; Howe et al., 2009; Kumar et al., 2013). The word shape matching technique is based on a distance measure between the template/query (input image or an image synthesized from the text query) and all candidate word images. Without model training, it is vulnerable to word shape variation and suffers from low matching accuracy. In contrast, the model-based method can be used for retrieving multi-writer or writer independent documents in large database, such as handwritten Chinese documents used in our experiments, by storing similarity scores computed based on handwriting recognition method.

For model-based scoring of handwritten Chinese documents, context information is usually integrated when calculating character similarity scores. In Huang et al. (2013), four geometric models were combined with character classifier outputs, with combining weights trained by optimizing a one-vs-all discrimination objective so as to maximize similarities of true words and minimize similarities of imposters. Similar methods could be found in Cheng et al. (2013) for text retrieval in online handwritten Japanese documents. Cao et al. (2009) proposed to incorporate probabilities of candidate word segmentation into word similarities to enhance the spotting performance in English documents. Different from these character/word model based methods, character confidences can be estimated in the N-best list (Rueber, 1997) or pruned lattice (named "word graph" in speech recognition) (Quiniou and Anquetil, 2007), based on text line recognition. The posterior probability of an edge is computed by summing up probabilities of all the paths that pass this edge, integrating character classification scores, linguistic and geometric contexts. Our former work (Zhang et al., 2012) estimated the character confidence based on a N-best recognition list and performed better than the transcription-based keyword spotting. Compared with the N-best list, lattices can incorporate more competing hypotheses and the estimation of edge scores will be more accurate. On the word graph (Ortmanns et al., 1997) or word lattice (Kemp and Schaaf, 1997), there are too many candidate paths to compute the path probability individually even if the lattice is pruned (Ney et al., 1997; Wessel et al., 2001; Sixtus and Ortmanns, 1999), so posterior probabilities are usually computed using a forward-backward procedure (Quiniou and Anquetil, 2007; Wessel et al., 2001).

Candidate word scoring in a document, i.e., probability calculation of candidate paths, is a sequence classification problem, which is usually formulated using Markovian sequence probabilistic models such as hidden Markov models (HMMs) (Rabiner, 1989; Su et al., 2009; Fischer et al., 2010), maximum entropy Markov models (MEMMs) (McCallum et al., 2000), conditional random fields (CRFs) (Lafferty et al., 2001; Do and Artieres, 2006; Yang et al., 2009). HMM is a generative model, which assumes conditional independence of observations given states. MEMMs use a conditional model to represent the probability of reaching a state given an observation and the previous state. These conditional probabilities are specified by exponential models, following from a maximum entropy argument. CRFs have all the advantages of MEMMs but also solve the label bias problem in a theoretical way. In addition to advantages of CRFs, semi-CRFs (Sarawagi and Cohen, 2004) also output a segmentation S of the observation sequence X, together with the label sequence Y assigned to segments (sub-sequences) of X, rather than label individual elements of X.

During keyword spotting in the lattice, dynamic programming algorithms are often used for word matching between the query and candidate characters. Inspired by beam-search for handwritten text line recognition (Liu et al., 2002), the character-synchronous search algorithm has been proposed for keyword spotting in Chinese (Zhang et al., 2013b) and Japanese (Cheng et al., 2013) handwritten documents. The search process repeats with every primitive component (a character or part of a character consisting a block of strokes after over-segmentation of the text line) as start by a tree search strategy and matched results are output as located instances. Using the Viterbi

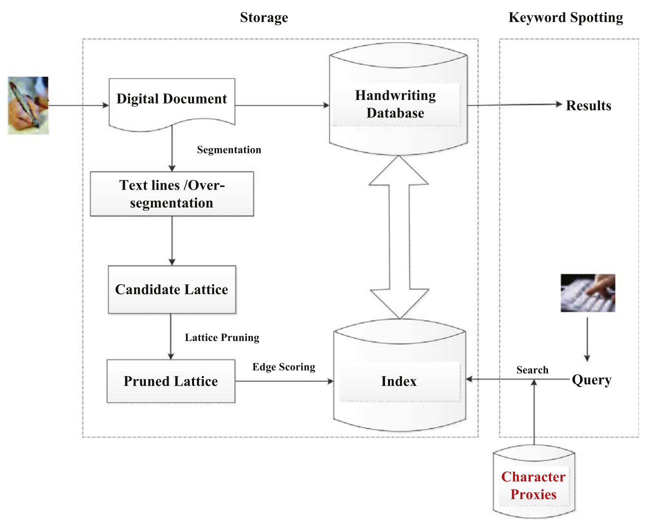


Fig. 1. Block diagram of the keyword spotting system.

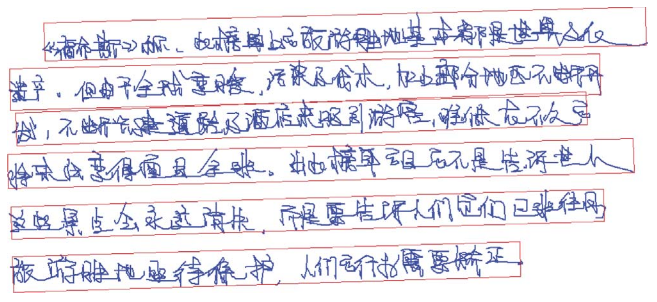


Fig. 2. Text line segmentation of a document.

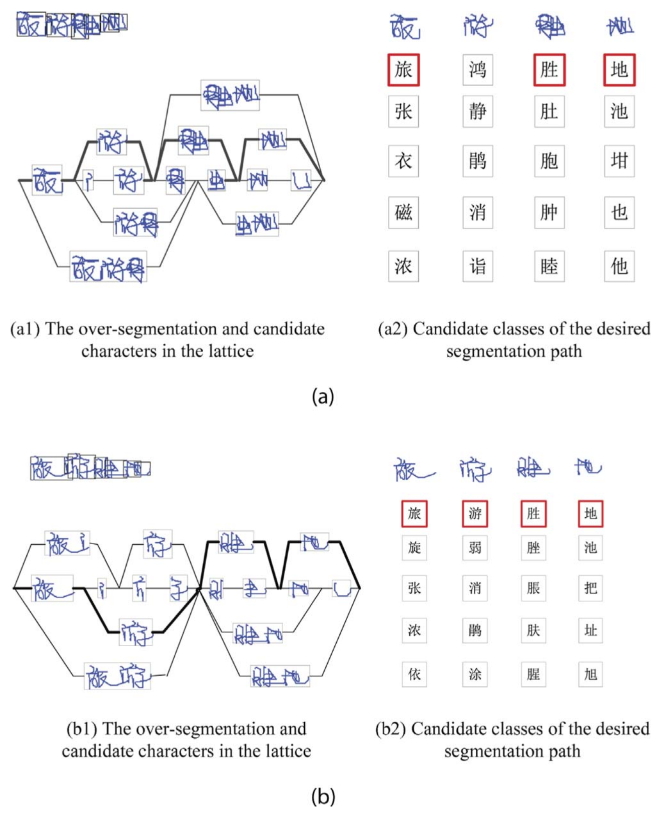

旅 鸿 胜 地
张 静 肚 池
衣 鹃 胞 坩
磁 消 肿 也
浓 诣 睦 他
(a1) The over-segmentation and candidate characters in the lattice
(a2) Candidate classes of the desired segmentation path
(a)
旅 游 胜 地
旋 弱 脞 池
张 消 胀 把
浓 鹃 肤 址
依 涂 腥 旭
(b1) The over-segmentation and candidate characters in the lattice
(b2) Candidate classes of the desired segmentation path
(b)

Fig. 3. (a) The second character in the desired path is not correctly recognized; (b) All the characters in the desired path are correctly recognized.

algorithm (Ploetz and Fink, 2009) with the character, filler and keyword hidden Markov models, the query can be located on the handwritten text lines without any segmentation (Fischer et al., 2012). The CTC token passing algorithm has been successfully used in a connectionist system (Graves et al., 2009) for handwriting recognition and modified to search for the query in the lattice (Frinken et al., 2012), which is the output of recurrent neural networks with long short-term memory (LSTM) cells.

3. System overview

For fast keyword spotting from a large collection of documents, the proposed system of online handwritten Chinese document retrieval consists of two stages: indexing and keyword search, as shown in Fig. 1. The indexing is done offline to generate the pruned candidate lattice and compute character confidence measures (edge probabilities/ scores), while the keyword search is performed online to locate instances matched with the query.

To build the index file, the document is first segmented into text lines according to the time and space interval between consecutive strokes (Zhou et al., 2009). Then each line is over-segmented into primitive components (stroke blocks) by stroke grouping according to the off-stroke distance (Zhou et al., 2007). Candidate characters are generated by combining consecutive components and are assigned candidate classes by the character classifier. Candidate characters and classes are represented in a candidate lattice representing alternative segmentation-recognition hypotheses of text lines. To enhance the character recognition performance, the original lattice is first augmented to supplement candidate classes and then pruned by a modified forward-backward algorithm for less storage space and fast character scoring. After edge scores are calculated based on the semi-CRF model, we store the number of primitive components and the bound of each primitive component. At each segmentation point, we store the number of edges ending at this point, and for each edge, its primitive component number, class label and edge score.

In text search, the query word is matched with sequences of candidate characters (partial paths in the candidate lattice) with every primitive component as the start. The word similarity is obtained by combining character similarity scores. When the word similarity is greater than a threshold, the word instance is located in the document. Fig. 2 shows an example of text line segmentation. For two instances with the same ground truth, Fig. 3(a) and Fig. 3(b) show text line over-segmentation results, candidate characters in the lattice and candidate classes of the desired segmentation path. In Fig. 3(a), the second character on the desired path is not correctly recognized and cannot be located by the previous CSDS algorithm (Zhang et al., 2013a). To locate both instances in Fig. 3(a) and Fig. 3(b), we propose the proxy-character driven search algorithm (cf. Section 6) by using proxies of the query character to search in the index file.

4. Building the index file using Semi-CRFs

In this section, we first describe the candidate character augmentation technique and then briefly introduce the semi-CRF model for lattice pruning and edge (character-label pair) score computation. At last, we propose the model trained method, i.e., CE criterion which views keyword spotting as binary classification of candidate characters. The compact lattices together with edge scores (for text lines in a document) are concatenated and stored into the index file for retrieval.

4.1. Candidate character augmentation

To reduce the risk of missing the true class in lattice generation, we first supplement candidate classes using the candidate character augmentation techniques based on linguistic context (Wang et al., 2012, 2012):

(1) Predict the character with the maximum probability in forward character bigram:

$$c_i^* = \arg\max_{c_i} P(c_i|c_{i-1}). \tag{1}$$

(2) Predict the character with the maximum probability in backward character bigram:

$$c_i^* = \arg\max_{c_i} P(c_i|c_{i+1}) = \arg\max_{c_i} \frac{P(c_{i+1}|c_i)P(c_i)}{P(c_{i+1})}. \tag{2}$$

(3) Character sequences related to topics in the document are likely to appear several times, so we predict the character by both forward and backward bigram:

$$\{c_i\text{: } (c_{i-1}, c_i) \in 1best \text{ or } (c_i, c_{i+1}) \in 1best\}, \tag{3}$$

where $1best$ stands for the candidate path with the maximum score in the lattice.

4.2. Semi-CRFs in candidate lattice

In Zhou et al. (2013), the semi-CRF model is defined in the lattice to directly estimate the posteriori probability of a hypothesized segmentation-recognition path given the text string X:

$$P(S, Y|X) = \frac{1}{Z(X)} \prod_{c\in S} \Psi_c(X, Y_c), \tag{4}$$

where (S,Y) stands for a segmentation S (candidate character sequence) of X paired with the label sequence Y, and $\Psi_c(X, Y_c)$ is the potential function on maximal clique c (m consecutive characters in the lattice when the order of semi-CRFs is $m-1$, and Y_c is the labeling of c):

$$\Psi_c(X, Y_c) = \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(X_c, Y_c)\right\}, \tag{5}$$

$f_k(X_c, Y_c)$ is the k-th feature function defined on clique c, which models character recognition, geometric or linguistic context. $\Lambda = \{\lambda_k | k = 1, \ldots, K\}$ are weighting parameters to be learned. $Z(X)$ is the partition function defined as the summation over all the paths in the lattice:

$$Z(X) = \sum_{(S',Y')} \prod_{c\in S'} \Psi_c(X, Y'_c). \tag{6}$$

4.3. Feature functions

In semi-CRFs, feature functions are used to capture attributes at different levels of granularity of the same observations (Lafferty et al., 2001). For text line recognition, feature functions include scores of isolated character recognition, the language model (*tri*-gram) (Wang et al., 2012, 2012) and four geometric models (Wang et al., 2012, 2012).

For candidate character feature extraction, we use the popularly stroke direction histogram feature, implemented by the normalization-cooperated feature extraction (NCFE) method of Liu and Zhou (2006) with bi-moment normalization. The extracted 512D feature vector is further reduced to 160D by Fisher linear discriminant analysis (FLDA) (Fukunaga, 1990). The state-of-the-art modified quadratic discriminant function (MQDF) (Kimura et al., 1987) classifier is used for isolated character recognition. The MQDF is a modified version of quadratic discriminant function (QDF) which roots from the Bayesian classifier with multivariate Gaussian density assumption. By replacing minor eigenvalues of each class in QDF as a constant, the MQDF is resulted in and only principal eigenvectors (50 in our experiments) are used in the discriminant function. The MQDF classifier reduces the computation complexity and meanwhile benefit the generalization performance.

Table 1
Summary of all the feature functions used in Semi-CRFs.

Type	Features	classes	Classifier	CT
Character recognition	NCFE	7356	MQDF	Yes
Language model	Tri-gram	\	\	\
Unary class-dependent	Unary geometric	7356	QDF	Yes
Unary class-independent	Unary geometric	2	SVM	Yes
Binary class-dependent	Binary geometric	36	QDF	Yes
Binary class-independent	Binary geometric	2	SVM	Yes

To model geometric contexts, we extract class-dependent and class-independent unary/binary geometric features from bounding boxes of a candidate character, and from two adjacent character patterns, respectively (Yin et al., 2013). Based on the fact that many different characters have similar geometric features, we particularly cluster character classes into 6 super-classes using the EM algorithm and so, the number of bi-character classes is 6*6 for binary class-dependent geometric model. The scores for class-dependent and class-independent geometries are given by quadratic discriminant functions (QDFs) and linear SVMs, respectively. All the classifier outputs of character recognition and geometric models are transformed to probabilistic confidences (confidence transformation, CT) (Zhou et al., 2013) according to the Dempster-Shafer theory of evidence:

$$P(c_j|x) = \frac{\exp[-\alpha d_j(x) + \beta]}{1 + \sum_{i=1}^{M} \exp[-\alpha d_i(x) + \beta]}, \tag{7}$$

where M is the total number of defined classes, $d_j(x)$ is the dissimilarity score of character c_j output by the classifier, the α and β are confidence parameters. All the feature functions used in Semi-CRFs are briefly summarized in Table 1.

4.4. Calculation of edge scores

To calculate edge scores, we should first estimate marginal probabilities on each lattice edge (the probability that an edge is on the desired segmentation-recognition path) using the forward and backward algorithm, and edge scores are just the logarithm of marginal probabilities.

Let $m \geq 2$ be the maximal clique size (unless otherwise stated, the default m is 3). Assume that candidate segmentation points are indexed from 0 to T in the lattice. We denote a sub-segmentation path (character sequence) by $\mathbf{t}_{a:b}$, with $t_a, t_{a+1}, \ldots, t_b$ being $b-a+1$ ordered candidate segmentation points, and the labeling of $\mathbf{t}_{a:b}$ is denoted by $\mathbf{y}_{a+1:b}$, with $y_{a+1}, y_{a+2}, \ldots, y_b$ being $b-a$ labels for each character of $\mathbf{t}_{a:b}$. Let $(S_{t_a}^{t_b}, Y_{t_a}^{t_b})$ be an arbitrary sub-path (character sequence and labeling) from candidate segmentation point t_a to t_b. The forward variables $\{\alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n})\}$ are calculated on each $(\mathbf{t}_{1:n}, \mathbf{y}_{2:n})$ in the lattice, where $t_n = 1, \ldots, T$. For each $(\mathbf{t}_{1:n}, \mathbf{y}_{2:n})$, $\alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n})$ is a summation of potential products (sum-product) over all the partial paths (character sequence and labeling) starting from segmentation point 0 and ending at $(\mathbf{t}_{1:n}, \mathbf{y}_{2:n})$. The forward variables and the partition function $Z(X)$ can be calculated by the forward algorithm:

(1) Initialization

$$\alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n}) = \sum_{j=2}^{n} \Psi_{\mathbf{t}_{1:j}}(X, \mathbf{y}_{2:j}), \text{ for } t_1 = 0, \quad 2 \leq n \leq m, \tag{8}$$

(2) Recursion

$$\alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n}) = \sum_{\substack{(S_0^{t_n}, Y_0^{t_n}):\\ (\mathbf{t}_{1:n}, \mathbf{y}_{2:n}) \in (S_0^{t_n}, Y_0^{t_n})}} \prod_{c \in S_0^{t_n}} \Psi_c(X, Y_c)$$

$$= \sum_{t_0, y_1} \Psi_{\mathbf{t}_{0:n}}(X, \mathbf{y}_{1:n}) \alpha_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1}), \text{ for } t_1 \neq 0, \quad n = m, \tag{9}$$

(3) Termination

$$Z(X) = \sum_{\mathbf{t}_{1:n}, \mathbf{y}_{2:n}} \alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n}), \quad \text{for } t_n = T. \tag{10}$$

In the above algorithm, $\Psi_{\mathbf{t}_{0:n}}(X, \mathbf{y}_{1:n})$ denotes the potential function on maximal clique-labeling pair $(\mathbf{t}_{0:n}, \mathbf{y}_{1:n})$. $\{\alpha_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n})\}$ are calculated sequentially from $t_n = 1$ to $t_n = T$.

Similarly, we can deduce the backward variables $\{\beta_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1})\}$ defined on each $(\mathbf{t}_{0:n-1}, \mathbf{y}_{1:n-1})$ in the lattice, where $t_0 = 0, \ldots, T-1$. When $t_{n-1} = T$, $2 \le n \le m$, otherwise $n = m$. $\beta_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1})$ is a summation of potential products over all the partial paths starting from $(\mathbf{t}_{0:n-1}, \mathbf{y}_{1:n-1})$ and ending at segmentation point T.

(1) Initialization

$$\beta_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1}) = 1, \quad \text{for } t_{n-1} = T, \quad 2 \le n \le m, \tag{11}$$

(2) Recursion

$$\beta_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1}) = \sum_{\substack{(S_{t_0}^T, Y_{t_0}^T):\\ (\mathbf{t}_{0:n-1}, \mathbf{y}_{1:n-1}) \in (S_{t_0}^T, Y_{t_0}^T)}} \prod_{c \in S_{t_{n-1}}^T} \Psi_c(X, Y_c)$$

$$= \sum_{t_n, y_n} \Psi_{\mathbf{t}_{0:n}}(X, \mathbf{y}_{1:n}) \beta_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n}), \text{ for } t_{n-1} \neq T, \quad n = m, \tag{12}$$

$\{\beta_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1})\}$ is calculated in reverse order from $t_0 = T-1$ to $t_0 = 0$.

From the forward and backward variables, we can calculate the marginal probability on any subpath $(\mathbf{t}_{0:k}, y_{1:k})$:

$$P(\mathbf{t}_{0:k}, y_{1:k}|X) = \sum_{(S,Y):(\mathbf{t}_{0:k}, y_{1:k}) \in (S,Y)} P(S, Y|X)$$

$$= \frac{1}{Z(X)} \sum_{\substack{\mathbf{t}_{2-m:1},\\ \mathbf{y}_{3-m:1}}} \sum_{\substack{\mathbf{t}_{k-1:k+m-2},\\ \mathbf{y}_{k:k+m-2}}} \alpha_{\mathbf{t}_{2-m:1}}(\mathbf{y}_{3-m:1})$$

$$\times \beta_{\mathbf{t}_{k-1:k+m-2}}(\mathbf{y}_{k:k+m-2}) \prod_{c \in \mathbf{t}_{1:k+m-2}} \Psi_c(X, Y_c), \text{ for } m \ge 2, \tag{13}$$

where $\{\alpha_{\mathbf{t}_{2-m:1}}(\mathbf{y}_{3-m:1})\}$ denotes all the forward variables ending at $(\mathbf{t}_{0:1}, y_1)$, and $\{\beta_{\mathbf{t}_{k-1:k+m-2}}(\mathbf{y}_{k-1:k+m-2})\}$ denotes all the backward variables starting from $(\mathbf{t}_{k-1:k}, y_k)$. By setting k=1, we can calculate the marginal probability on the edge $(\mathbf{t}_{0:1}, y_1)$:

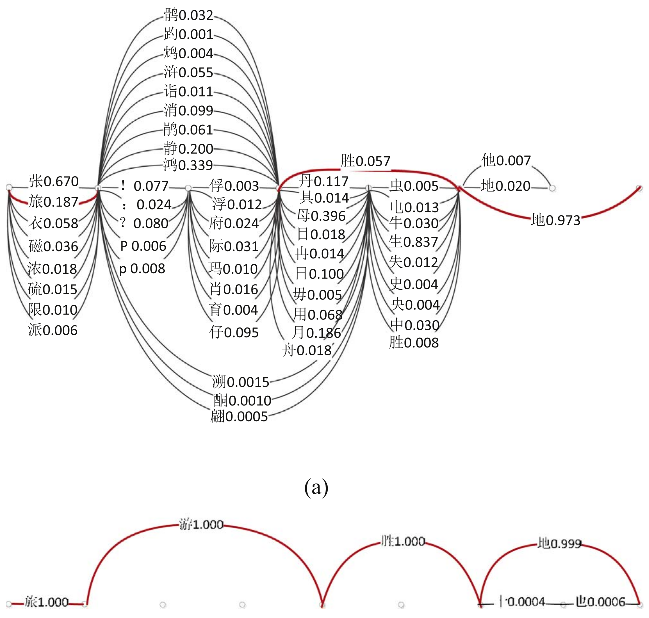
$$P(\mathbf{t}_{0:1}, y_1|X) = \sum_{(S,Y):(\mathbf{t}_{0:1}, y_1) \in (S,Y)} P(S, Y|X)$$

$$= \frac{1}{Z(X)} \sum_{\substack{\mathbf{t}_{2-m:1},\\ \mathbf{y}_{3-m:1}}} \sum_{\substack{\mathbf{t}_{0:m-1},\\ \mathbf{y}_{1:m-1}}} \alpha_{\mathbf{t}_{2-m:1}}(\mathbf{y}_{3-m:1})$$

$$\times \beta_{\mathbf{t}_{0:m-1}}(\mathbf{y}_{1:m-1}) \prod_{c \in \mathbf{t}_{1:m-1}} \Psi_c(X, Y_c), \text{ for } m \ge 2, \tag{14}$$

4.5. Lattice pruning

To build a compact index file, the edge scores are computed on a pruned lattice. The difficulty of lattice pruning lies in that the edges are not independent, i.e., to remove an edge will break the paths through it. To avoid breaking the high-probability paths, we consider a forward-backward lattice pruning method.



(b)

Fig. 4. The pruned lattices and edge probabilities (the red label indicates the correctly recognized characters).

Replacing the summation by maximization in Eqs. (9) and (12) will yield another type of forward and backward variables $(\{\hat{\alpha}_{\mathbf{t}_{1:n}}(\mathbf{y}_{2:n})|t_n = 1, \ldots, T\}$ and $\{\hat{\beta}_{\mathbf{t}_{0:n-1}}(\mathbf{y}_{1:n-1})|t_0 = 0, \ldots, T-1\}$, respectively), with which we can calculate the posterior probability of the best path traversing an edge by replacing the summation with maximization in Eq. (14). In this paper, the initial dense lattice will be pruned by a first-order semi-CRFs ($m = 2$), thus

$$P(S^*, Y^*, \mathbf{t}_{0:1} \in S^*, y_1 \in Y^*|X) = \max_{\substack{(S,Y):\\ (\mathbf{t}_{0:1}, y_1) \in (S,Y)}} P(S, Y|X)$$

$$= \frac{1}{Z(X)} \hat{\alpha}_{\mathbf{t}_{0:1}}(y_1) \hat{\beta}_{\mathbf{t}_{0:1}}(y_1). \tag{15}$$

In Eq. (15), $Z(X)$ is a constant with trained parameters. The value $\hat{\alpha}_{\mathbf{t}_{0:1}}(y_1)\hat{\beta}_{\mathbf{t}_{0:1}}(y_1)$ will be used as a score of $(\mathbf{t}_{0:1}, y_1)$ for pruning the lattice. Note that $\hat{\alpha}_{\mathbf{t}_{0:1}}(y_1)\hat{\beta}_{\mathbf{t}_{0:1}}(y_1)$ is also the score of the best path traversing $(\mathbf{t}_{0:1}, y_1)$.

Considering that a high-score path is unlikely to go through a low-score edge, removing a low-score edge may impossibly break a high-score path. Denote the best path score in the dense lattice by Q_{max}, an edge is retained if the following condition is held:

$$\log Q_{max} - \log(\hat{\alpha}_{\mathbf{t}_{0:1}}(y_1)\hat{\beta}_{\mathbf{t}_{0:1}}(y_1)) \le \gamma_p, \tag{16}$$

where $\gamma_p > 0$ is the pruning threshold. With this method, paths with higher scores are retained, while those with lower scores are discarded.

Fig. 4(a) and Fig. 4(b) respectively show the two pruned lattices and edge probabilities resulting from original lattices in Fig. 3(a) and Fig. 3(b). The pruning strategy results in a compact index file and speeds up the edge scoring, but it also cuts some correctly recognized edges. So we use the proxy-character driven search algorithm described in Section 5 to locate the mis-matched instances.

4.6. Model training with CE criterion

The semi-CRF model trained with MAP (maximum a posteriori) has been demonstrated to give high classification accuracies on Chinese document recognition (Zhou et al., 2013). However, MAP criterion aims to optimize the model for multi-classification while keyword spotting is binary classification of candidate characters in the lattice.

In our keyword spotting, the CE loss is proposed for semi-CRF model training to separate the target character from the others. We use edges in the lattices of training text lines as the training set, defined as:

$$\{(\mathbf{t}_{0:1}, y_1)|(\mathbf{t}_{0:1}, y_1) \in \mathsf{F}\}, \tag{17}$$

where $(\mathbf{t}_{0:1}, y_1)$ is an edge in the entire or pruned lattice F (to alleviate the computational burden in model training, the original lattice is first pruned with a pre-trained first-order semi-CRF (Zhou et al., 2013)). The multi-class label y_1 is transformed to binary label $L_1 = \delta(y_1, g_1)$, where g_1 is the ground truth of $\mathbf{t}_{0:1}$ and function $\delta(y_1, g_1) = 1$ if $y_1 = g_1$ or $\delta(y_1, g_1) = 0$. Then given N training samples $\{X_i|i = 1, 2...N\}$ and corresponding lattices $\{\mathsf{F}_i|i = 1, 2...N\}$, the CE criterion is to minimize the expected cost:

$$\begin{aligned} MinCE = &-\sum_{i=1}^{N} \sum_{(\mathbf{t}_{0:1},y_1)\in \mathsf{F}_i} \{L_1 logP((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) \\ &+ (1-L_1)log(1-P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda))\} \\ = &-\sum_{i=1}^{N} \sum_{(\mathbf{t}_{0:1},y_1)\in \mathsf{F}_i} F_i(\mathbf{t}_{0:1}, y_1), \end{aligned} \tag{18}$$

where $P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ denotes the marginal probability on edge $(\mathbf{t}_{0:1}, y_1)$ in the lattice, and function $F_i(\mathbf{t}_{0:1}, y_1)$ is defined as:

$$F_i(\mathbf{t}_{0:1}, y_1) = L_1 \log P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) + (1-L_1)log(1-P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)). \tag{19}$$

We minimize the empirical loss in Eq. (18) by stochastic gradient descent to estimate parameters Λ. On each training edge $(\mathbf{t}_{0:1}, y_1)$ in the lattice F_i of X_i, the parameters can be updated as:

$$\lambda_k \leftarrow \lambda_k + \alpha \frac{\partial F_i(\mathbf{t}_{0:1}, y_1)}{\partial \lambda_k}, \tag{20}$$

and the derivative of $F_i(\mathbf{t}_{0:1}, y_1)$ on λ_k can be computed as:

$$\frac{\partial F_i(\mathbf{t}_{0:1}, y_1)}{\partial \lambda_k} = \left(\frac{L_1}{P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)} - \frac{1-L_1}{1-P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)}\right) \frac{\partial P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)}{\partial \lambda_k}. \tag{21}$$

To compute the derivatives of the marginal probability in Eq. (21), we rewrite the marginal probability in Eq. (14) as:

$$P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) = \left\{ \sum_{(S,Y)\in \mathsf{F}_i \Lambda (\mathbf{t}_{0:1},y_1)\in(S,Y)} \prod_{c\in S} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(X_c^i, Y_c)\right)\right\} / Z(X_i, \Lambda). \tag{22}$$

Then the partial derivatives of $P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ can be computed as:

$$\begin{aligned} \frac{\partial P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)}{\partial \lambda_k} = &\sum_{(S,Y)\in \mathsf{F}_i \Lambda (\mathbf{t}_{0:1},y_1)\in(S,Y)} \sum_{c\in S} f_k(X_c^i, Y_c)P(S, Y|X_i, \Lambda) \\ &- \frac{P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)}{Z(X_i, \Lambda)} \frac{\partial Z(X_i, \Lambda)}{\partial \lambda_k}, \end{aligned} \tag{23}$$

where

$$\begin{aligned} &\sum_{(S,Y)\in \mathsf{F}_i \Lambda (\mathbf{t}_{0:1},y_1)\in(S,Y)} \sum_{c\in S} f_k(X_c^i, Y_c)P(S, Y|X_i, \Lambda) \\ &= \sum_{(c,Y_c)\in \mathsf{F}_i} f_k(X_c^i, Y_c) \sum_{(S,Y)\in \mathsf{F}_i \Lambda (c,Y_c)\in(S,Y) \Lambda (\mathbf{t}_{0:1},y_1)\in(S,Y)} P(S, Y|X_i, \Lambda) \\ &= \sum_{(c,Y_c)\in \mathsf{F}_i} f_k(X_c^i, Y_c)P((c, Y_c), (\mathbf{t}_{0:1}, y_1)|X_i, \Lambda). \end{aligned} \tag{24}$$

According to Eq. (6), we can calculate the partial derivatives of $Z(X_i, \Lambda)$ in Eq. (23) as:

$$\begin{aligned} \frac{\partial Z(X_i, \Lambda)}{\partial \lambda_k} &= Z(X_i, \Lambda) \sum_{(S,Y)\in \mathsf{F}_i} \sum_{c\in S} f_k(X_c^i, Y_c)P(S, Y|X_i, \Lambda) \\ &= Z(X_i, \Lambda) \sum_{(c,Y_c)\in \mathsf{F}_i} f_k(X_c^i, Y_c)P((c, Y_c)|X_i, \Lambda). \end{aligned} \tag{25}$$

Substituting Eqs. (24) and (25) into Eq. (23), we can get the derivatives of the marginal probability $P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ as:

$$\begin{aligned} \frac{\partial P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)}{\partial \lambda_k} = &\sum_{(c,Y_c)\in \mathsf{F}_i} f_k(X_c^i, Y_c)\cdot(P((c, Y_c), (\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) \\ &- P((c, Y_c)|X_i, \Lambda)P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)). \end{aligned} \tag{26}$$

Denoting the size and segmentation point of clique (c, Y_c) as s and t_c, then the joint probability $P((c, Y_c), (\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ in Eq. (26) can be rewritten as $P((\mathbf{t}_{c:c+s}, y_{c:c+s}), (\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ and computed by:

$$P((\mathbf{t}_{c:c+s}, y_{c:c+s}), (\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) = \begin{cases} P((\mathbf{t}_{c:c+s}, y_{c:c+s})|X_i, \Lambda) & for\ t_{c+s} < t_0 or t_1 < t_c, \\ \quad P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda) & \\ P((\mathbf{t}_{c:c+s}, y_{c:c+s})|X_i, \Lambda) & for\ (\mathbf{t}_{0:1}, y_1) \in (\mathbf{t}_{c:c+s}, y_{c:c+s}), \\ P((\mathbf{t}_{c-1:c+s}, y_{c-1:c+s})|X_i, \Lambda) & for\ t_1 = t_c \\ P((\mathbf{t}_{c:c+s+1}, y_{c:c+s+1})|X_i, \Lambda) & for\ t_{c+s} = t_0 \\ 0 & others, \end{cases} \tag{27}$$

where the sub-path $(\mathbf{t}_{c-1:c+s}, y_{c-1:c+s})/(\mathbf{t}_{c:c+s+1}, y_{c:c+s+1}))$ is the concatenation of the edge $(\mathbf{t}_{0:1}, y_1)$ and the clique $(\mathbf{t}_{c:c+s}, y_{c:c+s})$ when $t_1 = t_c/t_{c+s} = t_0$. In Eq. (27), we can see that when $t_{c+s} < t_0 or t_1 < t_c$, the partial derivatives of $P((\mathbf{t}_{0:1}, y_1)|X_i, \Lambda)$ is zero and so the summation space in Eq. (26) is reduced.

5. Proxy-character driven search algorithm

Previously, we use a character-synchronous dynamic search algorithm for locating keywords in the lattice (Zhang et al., 2013a). The search algorithm contains two key steps: one is the candidate character scoring and the other is the word search. If the query is matched with the candidate character/word, the similarity score is the logarithm of the edge probability; otherwise, is $-\infty$. So, the query mis-matched with the candidate character/word cannot be located and the recall rate is limited. In Fig. 4(a), the candidate character between segmentation points (1,3) is not correctly recognized, and so, this instance cannot be located by the previous CSDS algorithm. Fig. 5 shows the spotting result in the document and we can see only the instance in Fig. 4(b) is located. In this section, to locate incorrectly recognized candidate characters, we propose a new character-synchronous dynamic search algorithm by using proxy-characters of the query to search in the index file.

5.1. Use of proxy-character

Proxy characters are generated on training character samples. The set of training samples is denoted as $\{x_{in_i}|i = 1,...,Cand n_i = 1,...,N_i\}$, where C is the total class number of Chinese characters and N_i is the

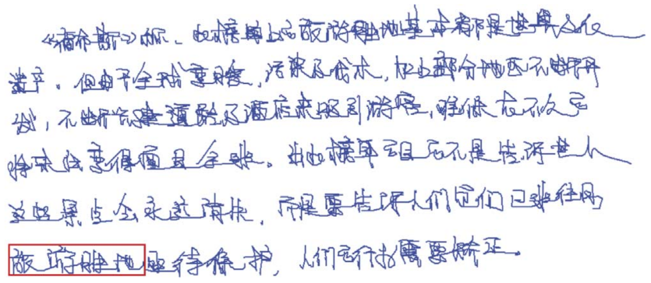


Fig. 5. The locating result by the previous CSDS algorithm: the instance in the first line is not found.

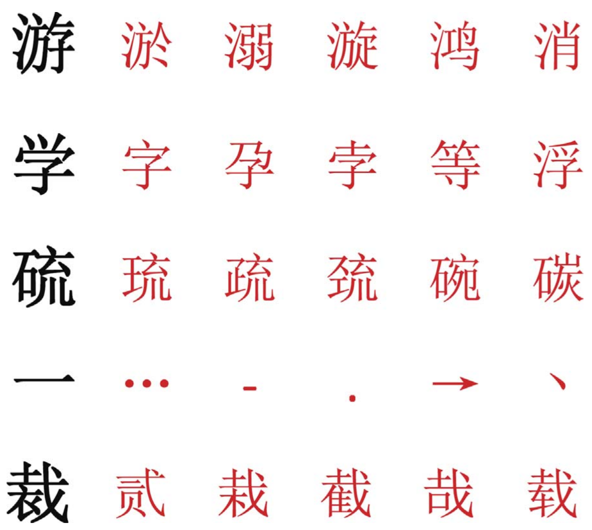


Fig. 6. An example of character proxies: the red small characters are part of proxies.

total number of training samples for character c_i. During the proxy-character generation, each sample x_{in_i} is classified into N_1 (set to 20 in the experiments) candidate classes Ω_{in_i}. If character $c_j \in \Omega_{in_i}$, do $N_{ij} \leftarrow N_{ij} + 1$ where N_{ij} (initialized as 0) is denoted as the total number of times that character c_j $(j \neq i)$ appears in the candidate classes of character c_i. After the classification of all the training samples, for each character c_i, we rank $\{N_{ij}|j = 1, \ldots, Candj \neq i\}$ and the top N_2 (set to 20 in our experiments) confusing characters are generated as the proxy characters of c_i. Fig. 6 shows an example of character proxies which can be used to search in the index file. We can see that in Fig. 3(a), the second character on the desired path is not correctly recognized but can be located with proxy-characters to search in the lattice.

Our use of proxy characters is inspired by proxy keywords in speech search, which entails some in-vocabulary keywords that are acoustically similar as the proxies of OOV (out of vocabulary) query (Chen et al., 2013). Compared with proxy keywords in speech search, our use of proxy-characters has several different issues:

(1) We use proxies to spot the in-vocabulary but incorrectly recognized characters while proxy keywords in speech are used to search OVV words.
(2) We use proxies of characters not words to drive the character-synchronous dynamic search.
(3) We entail characters that are similar in feature space as proxies while speech search augments proxy keywords that are acoustically similar.
(4) Our proxies are generated on training character samples while speech proxies are generated in the lattice.
(5) A candidate character may be recognized as different proxies of the ground truth, so we sum the probabilities of all the proxies in the candidate classes to compute matching score.
(6) To reduce the time cost in keyword search, we use proxy-characters only when the candidate character cannot be matched with the query and only one character in the query is searched with the proxies.

5.2. Word spotting by proposed search

In the dynamic search, the query word $\mathbf{y}_{1:n} = y_1, y_2, \ldots, y_n$ is matched with sequences of candidate characters (partial paths in the candidate lattice) with every component as the start. The similarity between the

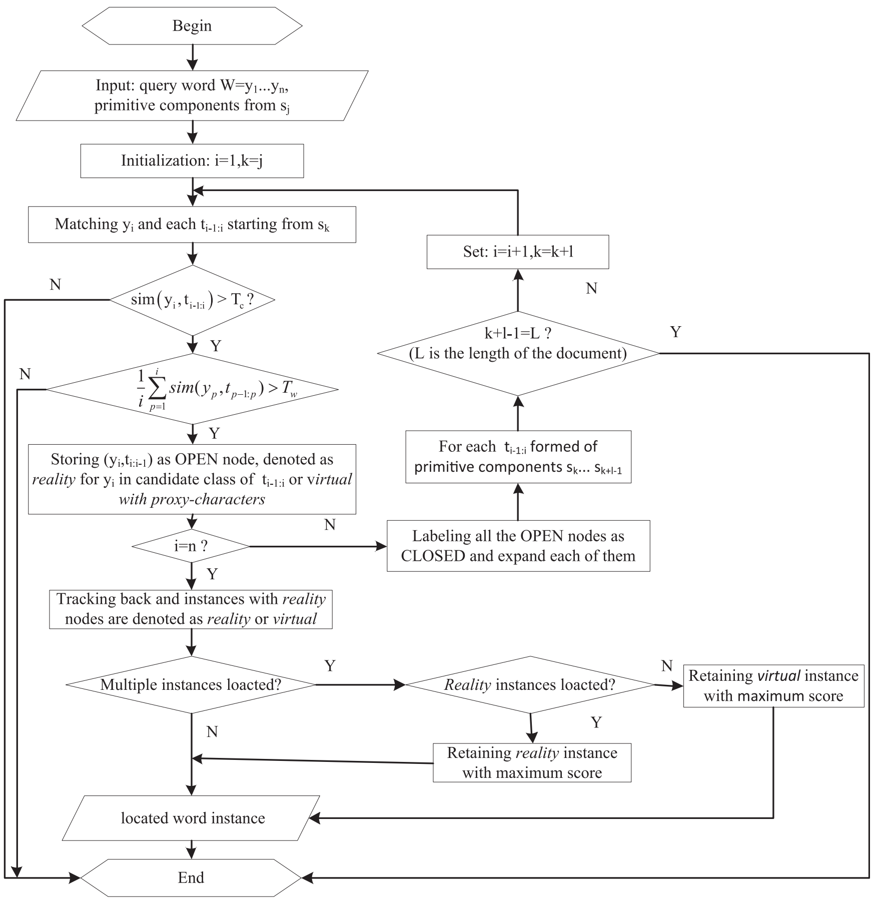

Begin
Input: query word W=y1...yn, primitive components from sj
Initialization: i=1,k=j
Matching yi and each ti-1:i starting from sk
sim(yi, ti-1:i) > Tc ?
N
Y
(1/i) Σ_{p=1}^{i} sim(yp, tp-1:p) > Tw
N
Y
Storing (yi,ti:i-1) as OPEN node, denoted as reality for yi in candidate class of ti-1:i or virtual with proxy-characters
i=n ?
N
Y
Tracking back and instances with reality nodes are denoted as reality or virtual
Multiple instances loacted?
Y
N
Reality instances loacted?
N
Y
Retaining virtual instance with maximum score
Retaining reality instance with maximum score
located word instance
End
Labeling all the OPEN nodes as CLOSED and expand each of them
For each ti-1:i formed of primitive components sk... sk+l-1
k+l-1=L ? (L is the length of the document)
Y
N
Set: i=i+1,k=k+1

Fig. 7. The flow chart of the proposed keyword search.

query and a sequence of n candidate characters $\mathbf{t}_{0:n}$, with $t_0, t_1, \ldots, t_n$ being $n+1$ ordered candidate segmentation points, is obtained by combining the similarities of candidate characters:

$$SIM(\mathbf{y}_{1:n}, \mathbf{t}_{0:n}) = \frac{1}{n}\sum_{i=1}^{n} sim(y_i, \mathbf{t}_{i-1:i}), \tag{28}$$

where $sim(y_i, \mathbf{t}_{i-1:i})$ denotes the similarity between candidate character $\mathbf{t}_{i-1:i}$ and class y_i, which is defined as the logarithm of marginal probability $P((t_{i-1:i}, y_i)|X)$ on edge $(\mathbf{t}_{i-1:i}, y_i)$; otherwise, is $-\infty$ for mis-matched characters in our previous works (a, 2013). In this paper, we use proxies of miss-matched character for keyword searching and the similarity can be computed as:

$$sim(y_i, t_{i-1:i}) = \begin{cases} \log P((t_{i-1:i}, y_i)|X) & for\ y_i \in \Omega_{t_{i-1:i}}, \\ \sum_{p_i \in \Omega_{t_{i-1:i}}} \log P((t_{i-1:i}, p_i)|X) & for\ y_i \in \overline{\Omega}_{t_{i-1:i}}, \end{cases} \tag{29}$$

where p_i is the proxy of character y_i and $\Omega_{t_{i-1:i}}$ is the candidate class set of $t_{i-1:i}$ in the lattice. When the word similarity score is greater than the given threshold, the word instance is located in the document. To accelerate text search, we use two thresholds to prune matched candidate characters: a character threshold T_c to prune those characters with $sim(y_i, \mathbf{t}_{i-1:i})$ and a word threshold T_w to prune those words with $SIM(\mathbf{y}_{1:n}, \mathbf{t}_{0:n})$.

For a specific start component s_j, the flow chart of proposed search process is described in Fig. 7, wherein a pair of matched candidate-query characters without proxies is stored as a reality node in the state space or *virtual* node for the pair with proxy-characters. Compared with the previous CSDS algorithm, the main modification of the proposed search lies in character matching and located instance pruning when multiple instances are returned during tracking back. At character matching step, the *virtual* node is stored as an OPEN node to search proxies for mis-matched characters. To keep the precision rate not too low and reduce the search time, we constrain that there is at most one *virtual* node on each search path. At pruning step, it is more likely for *reality* instances to be correctly recognized than *virtual* instances, so we retain the *reality* one of maximum score and prune the other. If all the matched instances are *virtual*, we retain the one whose *virtual* node score is maximum. The pruning metric is similar to the previous algorithm in Zhang et al. (2013a).

Fig. 8 shows an example of the proposed search process. There are two located instances (W_1,W_2) and both of them *virtual*. For instances W_1 and W_2, their *virtual* nodes are the same. So we retain the W_2 whose path score is higher. Fig. 9 shows an example of the proposed search results. Compared with the result in Fig. 5, the proposed search can locate all the instances and so the recall rate is high.

6. Experiments

We evaluated the performance of the proposed keyword spotting method on a database of online Chinese handwriting: CASIA-OLHWDB (Liu et al., 2011). This database is divided into six data sets, three for isolated characters and three for handwritten texts. There are 3,912,017 isolated character samples and 5,092 handwritten pages (52,220 text lines) in total. Both the isolated data and handwritten text data have been divided into standard training set (816 writers) and testing set (204 writers). The training set contains 3,129,496 isolated character samples of 7356 classes and 4072 pages of handwritten texts (41,710 text lines, including 1,082,220 characters of 2,650 classes). The presented system was evaluated on the test set of texts, which contains 10,510 text lines from 1,020 text pages, including 269,674 characters of 2631 classes. Fig. 10(a) shows some samples of isolated characters, and Fig. 10(b) shows a handwritten text page with multiple lines of characters.

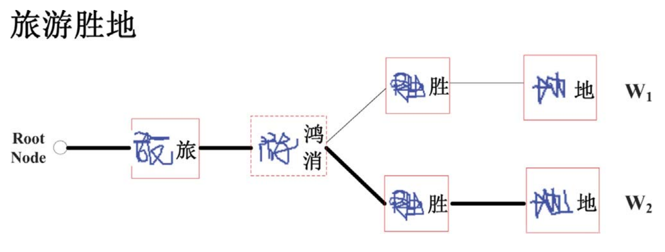


Fig. 8. An example of the proposed proxy-character driven search: two proxy characters are used in depth2.

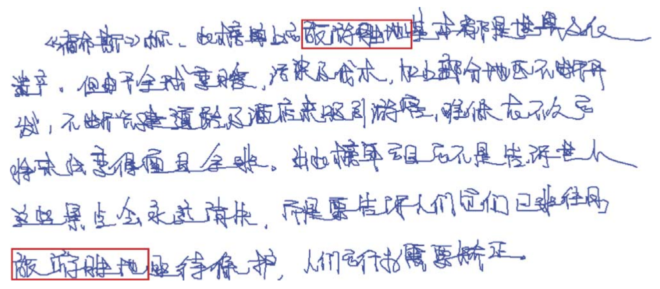


Fig. 9. The locating results by the proposed proxy-character driven search: both of the two instances are correctly located.

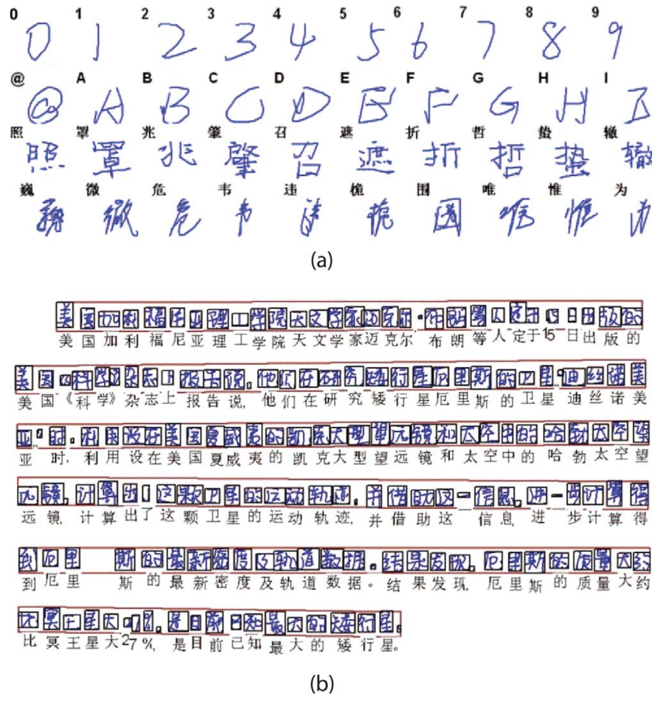

0 1 2 3 4 5 6 7 8 9
@ A B C D E F G H I
照 罩 兆 肇 召 遮 折 哲 蛰 辙
巍 微 危 韦 违 桅 围 唯 惟 为
(a)
美国加利福尼亚理工学院天文学家迈克尔·布朗等人定于15日出版的
美国《科学》杂志上报告说，他们在研究矮行星厄里斯的卫星迪丝诺美
亚时，利用设在美国夏威夷的凯克大型望远镜和太空中的哈勃太空望
远镜，计算出了这颗卫星的运动轨迹，并借助这一信息进一步计算得
到厄里斯的最新密度及轨道数据。结果发现，厄里斯的质量大约
比冥王星大27%，是目前已知最大的矮行星。
(b)

Fig. 10. (a) Example of isolated character samples; (b) A handwritten text page. Each character sample is attached with a class label.

6.1. Experimental setting

The training isolated characters and the characters segmented from the training text pages were used for parameter estimation of the character classifier, confidence estimation and proxy-character generation. The classifier parameters were learned using 4/5 of the data and the remaining 1/5 were used for confidence parameter estimation. The geometric features were extracted from the training text lines and used to learn the respective geometric models. The language model is the same as in Wang et al. (2012, 2012) and the semi-CRF parameters were learned on the training text lines.

For evaluating the retrieval performance on the test dataset, we use the high-frequency words in the lexicon of the Sogou labs (SogouLab,) as query words. The top 60,000 frequently used words, including

Table 2
Effects of lattice pruning.

γ_p	1	5	10	15	20	30
EER (%)	null	5.42	5.29	5.28	5.28	5.28
AUC (*0.01)	91.45	93.88	94.34	94.40	94.40	94.40
TR (%)	93.65	95.05	95.49	95.64	95.73	95.80
LED	1.05	1.45	2.94	6.07	10.86	23.83
Index-Size (Mb)	5.96	6.49	8.51	12.73	19.19	36.68

39,057 two-character words, 9975 three-character words and 9451 four-character words, were tested in our experiments. The keyword spotting performance is measured using the recall-precision curve, EER (Equal Error Rate, $(1 - recall)*100\%$ when recall equals to precision) and AUC (Area Under the Curve).

Our experiments were implemented on a PC with Intel(R) Core(TM) i5-3470 CPU, 3.20 GHz processor and 4 GB RAM, and the algorithms were programmed using C++.

6.2. Effects of lattice pruning

For simplification, we perform the character detection to evaluate the effects of lattice pruning. Table 2 lists the effects of different lattice pruning thresholds (γ_p). The total recall (TR) rate is defined as the number of true instances in the lattice divided by the number of instances in the transcript (ground truth), which is a upper bound of the recall rate. Lattice edge density (LED) is defined as the total number of edges divided by the total number of characters in the transcript, which is used to measure the complexity of the index file. From Table 2 we can see that, by decreasing γ_p, lattice pruning can effectively reduce LED and consequently the size of index file, while EER and AUC change just slightly when γ_p is not too small. The default threshold $\gamma_p = 10$ performs sufficiently well in respect of EER and AUC. Enlarging the threshold, though increases the total recall rate, does not improve the two metrics. Fig. 11 shows the recall-precision curves of character detection with different lattice pruning thresholds and similar results can be found: the metrics will not improve significantly when the threshold γ_p is above 10. So in the following experiments, we use threshold $\gamma_p = 10$ for the lattice pruning to balance the perform metric and the disc cost.

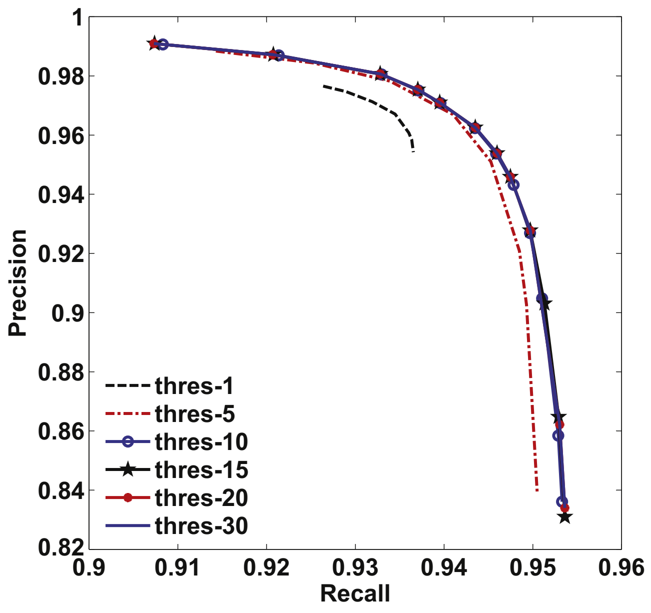


Fig. 11. Recall-precision curves of character detection with different lattice pruning thresholds.

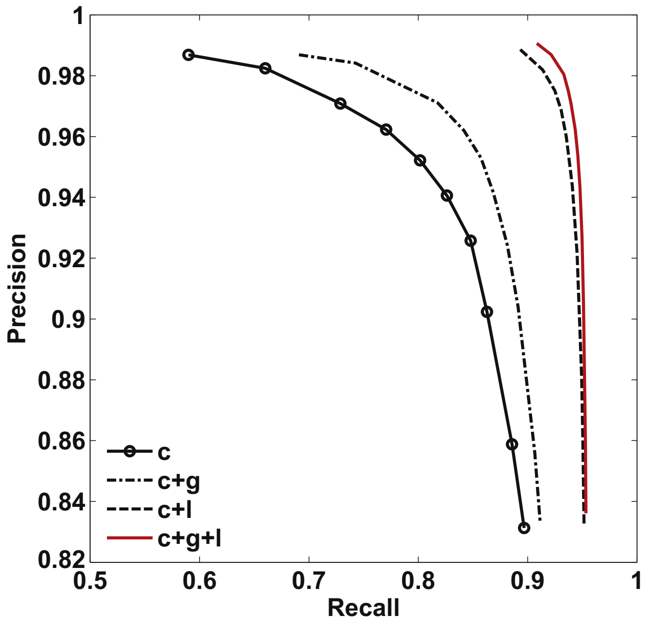


Fig. 12. Recall-precision curves of spotting results combining different contexts.

6.3. Effects of contexts

To evaluate the effects of different contexts, we perform the keyword spotting experiments with all the three kinds of keywords by our previous semi-CRF model and search algorithm in Zhang et al. (2013). Fig. 12 compares the recall-precision curves of different contexts, where “c”, “g” and “l” denote the feature functions for character recognition, geometric contexts and linguistic context, respectively. From Fig. 12, we can see that compared to using the character recognition model (“c”) only, the performance is remarkably improved by combining geometric contexts (“c+g”). The incorporation of linguistic context (“c+l”) is much more effective than the geometric contexts. The best result is given by combining both types of contexts (“c+g+l”). Because the language model was trained on a large corpus and can help distinguish between confusing similar characters, the system can always produce very high recall rate with linguistic context (“c+l” and “c+g+l”). Table 3 lists the keyword spotting results of different contexts, and we can directly find the best results combining all the contexts. So the following experiments are performed with the three features combined in semi-CRFs.

6.4. Effects of candidate character augmentation

The above experiments evaluate the effects of lattice pruning and the different features combined in semi-CRFs, but the candidate character augmentation technique has not been used. We then augment candidate characters of the original lattice, prune the lattice with the default threshold and perform the keyword spotting experiments with all the three features in the semi-CRFs model. The recall-precision curves are shown in Fig. 13, where CCA denotes spotting with candidate character augmentation, and Non-CCA denotes spotting without candidate character augmentation. Table 4 lists the EER and AUC with and without CCA. We can see that the candidate character

Table 3
Keyword spotting results combining different contexts.

Contexts	c	c+l	c+g	c+g+l
EER (%)	12.36	5.87	10.57	5.29
AUC (*0.01)	87.51	93.93	89.36	94.34

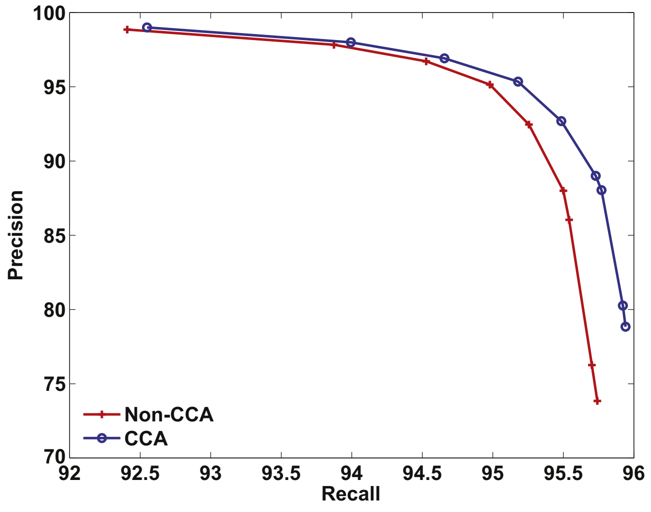


Fig. 13. Recall-Precision curves of spotting results with and without candidate character augmentation.

Table 4
Keyword spotting results with and without candidate character augmentation.

Methods	non-CCA	CCA
EER (%)	5.01	4.80
AUC (*0.01)	94.55	94.88

augmentation technique can slightly improve the spotting performance.

Based on the above experimental results, we use the exact configuration, i.e., the candidate character augmentation technique, lattice pruning and all the context features for the CE based semi-CRFs, and also the proxy-character driven search to perform experimental comparisons with reference methods.

6.5. Comparison with reference methods

We compare the proposed keyword spotting performance with several reference methods: the previous system based on semi-CRFs trained with MAP criterion (Zhang et al., 2013), keyword spotting by CSDS search without proxy characters (Zhang et al., 2014, 2013a), character scoring based on the N-best list (Zhang et al., 2012) and keyword search on text line recognition (transcription) (Zhou et al., 2013).

6.5.1. Comparison with MAP based semi-CRFs

We compare the proposed CE based semi-CRFds with our previous method (Zhang et al., 2013) using MAP criterion derived from the handwriting recognition (Zhou et al., 2013). Given N training samples: $\{(X^i, S^i, Y^i)|i = 1,\ldots,N\}$ (strings with segmentation points and character classes labeled), following the standard MAP estimation, the weighting parameters Λ can be learned by minimizing the negative log-likelihood loss:

$$L_{NLL}(\Lambda) = -\sum_{i=1}^{N} \log P(S^i, Y^i|X^i; \Lambda). \tag{30}$$

With MAP criterion, the parameters of semi-CRFs are also optimized using stochastic gradient descent. The recall-precision curves are shown in Fig. 14. The EER and ACU results are shown in Table 5. We can see that the CE training criterion can slightly improve the spotting performance.

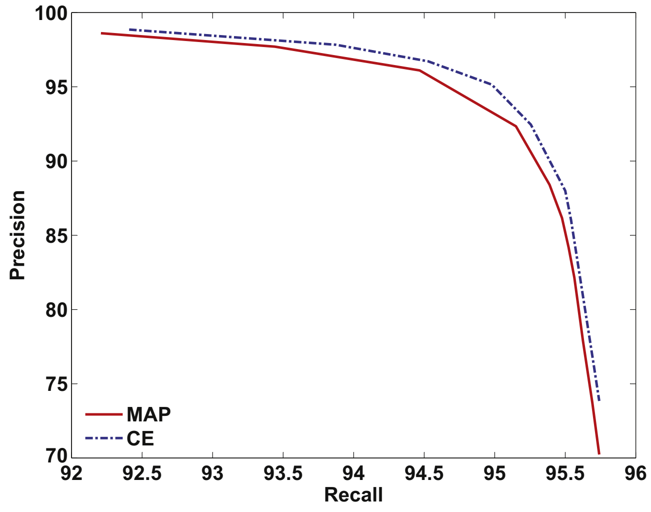


Fig. 14. Recall-precision curves of spotting results with semi-CRFs trained by the MAP and CE criterions.

Table 5
Keyword spotting results with semi-CRFs trained by the MAP and CE criterions.

Methods	MAP	CE
EER (%)	5.29	5.01
AUC (*0.01)	94.34	94.55

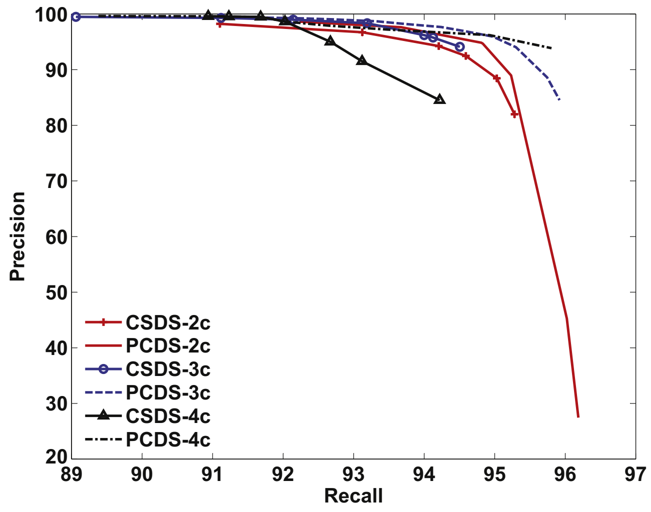


Fig. 15. Recall-precision curves of spotting results with the previous CSDS and the proposed PCDS search respectively.

Table 6
Effects of the proposed PCDS search compared with the previous CSDS algorithm.

Word Length	Previous CSDS method		Proposed PCDS search	
	EER (%)	AUC (*0.01)	EER (%)	AUC (*0.01)
2c	5.25	93.47	5.19	94.47
3c	5.57	93.97	4.62	95.32
4c	7.06	93.66	4.72	95.31

6.6. Comparison with traditional character-synchronous search

In this section, to evaluate the effects of the proposed proxy-character driven search for keywords of different length, we perform the keyword spotting experiments using queries with 2, 3, 4 characters

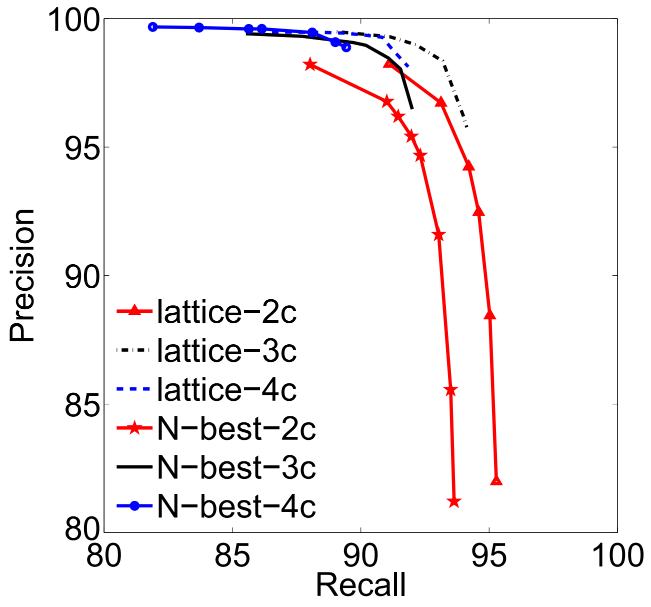


Fig. 16. Recall-Precision curves of keyword spotting by the N-based search and the proposed method.

(2c, 3c,4c) respectively. After pruning the augmented lattice with the default threshold and the computation of the edge scores with the three features in semi-CRFs, we perform the previous and the proposed search algorithms respectively. The recall-precision curves are shown in Fig. 15, where CSDS-2c denotes spotting with previous CSDS method, and PCDS-2c denotes spotting with proposed proxy-character driven search (PCDS) algorithm. The EER and AUC comparison can be found in Table 6. We can see that the PCDS search can improve the spotting performance especially for long words.

6.6.1. Comparison with N-best list based scoring

The former work (Zhang et al., 2012) estimates the character confidence based on the N-best recognition list obtained using the beam search algorithm from the original lattice. There is no essentially difference between the N-best list and the lattice. But the lattice includes more candidate paths than the N-best list, so the edge probability calculated in the lattice is better than the one on the N-best list. Besides, our semi-CRF model can better fusing the different contexts for evaluation of the path confidence/score. Fig. 16 compares the recall-precision curves of the proposed method and the N-best-based method. For fairly comparison, both methods use the previous CSDS algorithm, and N is set to 50 as in Zhang et al. (2012). From Fig. 16, we can see that the proposed method outperforms the N-best-based approach.

6.6.2. Comparison with transcription-based search

Because of the limited accuracy in handwriting recognition, most previous works on keyword spotting avoid using a text recognition system to transcribe the handwriting and search on the output text. However, some experimental studies (e.g., the one in (Frinken et al., 2012; Zhang et al., 2012)) show that transcription-based spotting can perform competitively. Here, on handwritten Chinese documents, we compared the proposed method with transcription-based search using a semi-CRFs based text line recognition method (Zhou et al., 2013). Since text line recognition gives unique text output, transcription-based word search gives a unique point of recall-precision rates (Fig. 17). In contrast, the proposed method provides flexible options of tradeoff between the two metrics. By properly sacrificing the precision rate, much higher recall rate can be achieved than the transcription-based method.

Table 7 shows the index size (the size of the index file for all test

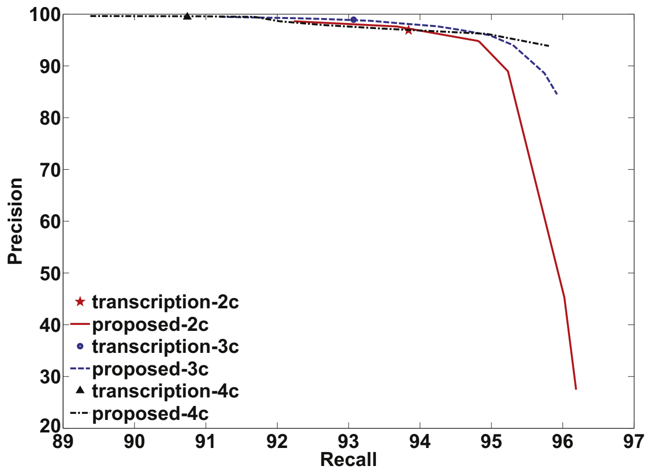


Fig. 17. Recall-Precision curves of keyword spotting by the transcription-based search and the proposed method.

Table 7
The index size (Mb) and the average keyword spotting time (ms) with different methods.

Method	Ink Size (Mb)	Index Size (Mb)	Search Time (ms)
Transcription		2.66	2.83
N-Best	47.71	6.32	6.47
Proposed		8.51	18.03

documents) and the average searching time (with threshold $-\infty$ so as to spot all the query words present in the lattice) for one word by the proposed method and reference methods. The data size of the test pages in stroke trajectory (4 bytes for a sampled point) is 47.71 Mb. We can see that though the proposed method consumes larger index storage and more search time than reference methods, the increased data size of index file is much smaller than the original handwriting data and the search time is acceptable.

6.7. Error analysis

The error reasons related to the semi-CRFs model are already analyzed in Zhou et al. (2013), so we only discuss about the proposed proxy-character driven search in this paper. If the candidate character is not correctly recognized and the proxy-characters are also not included in the candidate classes, the query will be mis-matched resulting in low recall. On the other hand, the candidate character is correctly recognized but some proxies of the wrong character are included in the candidate classes, the query will be incorrectly matched resulting in false positives.

Fig. 18 shows the example that the candidate character (3,5) is not correctly recognized and the candidate classes are not included in the proxies of the true character. Using the proposed proxy-character search, the candidate character at depth4 can not be matched with the forth character in the query. Such that the right word will be rejected. Fig. 19 shows the example that the candidate character (3,4) is correctly recognized in the candidate classes. Using the proposed proxy-character search, candidate character at depth4 can be matched with one proxy of the forth character in the query. Such that the wrong word will be accepted by a low threshold. To alleviate the wrong matching with proxy-characters, we use the proxies to match the candidate character with its maximum candidate score (logarithm of the edge probability) below a threshold (set to −0.3 in our experiments). For the candidate character with a large recognition score, it is possible that this candidate character is correctly recognized and so the proxy-characters are not needed.

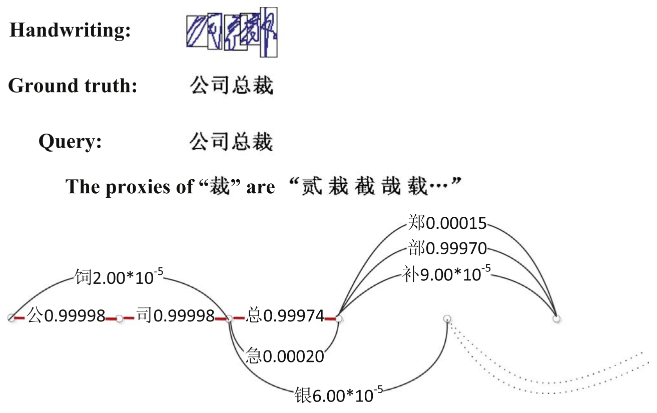


Fig. 18. The last character of the ground truth is not correctly recognized in the lattice and the proxy characters are also not included in the candidate classes.

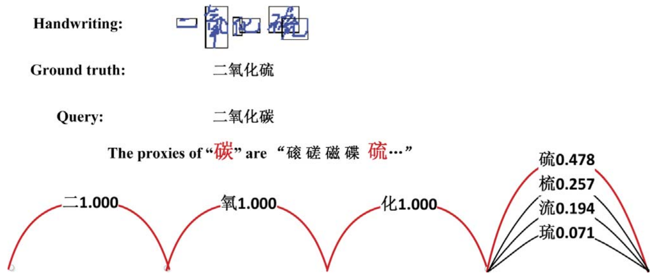


Fig. 19. The forth character of the query will be incorrectly matched with the character ($\mathbf{t}_{3:4}$) by the proposed search.

7. Conclusion

In this paper, we present an indexing method for keyword spotting in online handwritten Chinese documents based on semi-CRFs, which provide a theoretical framework for fusing the information of character recognition, geometric and linguistic contexts. By candidate character augmentation and lattice pruning, we obtain a compact index file for keyword spotting. In the pruned candidate segmentation-recognition lattice, the candidate character sores are estimated based on the semi-CRF model. The parameters of semi-CRFs are optimized using the cross-entropy (CE) criterion which can discriminate the candidate characters in the lattice. To locate mis-recognized characters, the confusing similar characters are used as proxies to search in the index file. Experimental results on the CASIA-OLHWDB database demonstrate the effectiveness of the proposed method, and justify the benefits of lattice pruning, the combination of different contexts, candidate character augmentation, CE training and the proxy-character driven search algorithm respectively. The spotting performance can be further improved by a systematical study on the re-scoring of incorrectly recognized candidate characters.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC) under grants no. 61403385 and no. 61273269.

References

Cao, H., Bhardwaj, A., Govindaraju, V., 2009. A probabilistic method for keyword retrieval in handwritten document images. Pattern Recognit. 42 (12), 3374–3382.

Chen, Guoguo, Yilmaz, Oguz, Trmal, Jan, 2013. Daniel Povey and Sanjeev Khudanpur, Using Proxies for OOV Keywords in the Keyword Search Task. In: Workshop on Automatic Speech Recognition and Understanding, pp. 416–421.

Cheng, C., Zhu, B., Nakagawa, M., 2013. Digital ink search based on character-recognition candidates compared with feature-matching-based approach. IEICE Trans. Inf. Syst. E96-D (3), 681–689.

Do, T.-M.-T., Artieres, T., 2006. Conditional random fields for online handwriting recognition. In: Proceedings IWFHR, pp. 197–202.

Fischer, A., Keller, A., Frinken, V., Bunke, H., 2010. HMM-based word spotting in handwritten documents using subword models. In: Proceedings ICPR, pp. 3416–3419.

Fischer, A., Keller, A., Frinken, V., Bunke, H., 2012. Lexicon-free handwritten word spotting using character HMMs. Pattern Recognit. Lett. 33 (7), 934–942.

Frinken, V., Fischer, A., Manmatha, R., Bunke, H., 2012. A novel word spotting method based on recurrent neural networks. IEEE Trans. Pattern. Anal. Mach. Intell. 34 (2), 211–224.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition 2nd edn.. Academic Press.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J., 2009. A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern. Anal. Mach. Intell. 31 (5), 855–868.

Howe, N.-R., Feng, S., Manmatha, R., 2009. Finding words in alphabet soup: inference on freeform character recognition for historical scripts. Pattern Recognit. 42 (12), 3338–3347.

Huang, L., Yin, F., Chen, Q.-H., Liu, C.-L., 2013. Keyword spotting in unconstrained handwritten Chinese documents using contextual word model. Image Vis. Comput. 31 (12), 358–368.

Jain, A.-K., Namboodiri, A.-M., 2003. Indexing and retrieval of on-line handwritten documents. In: Proceedings ICDAR, pp. 655–659.

Jawahar, C.-V., Balasubramanian, A., Meshesha, M., Namboodiri, A.-M., 2009. Retrieval of online handwriting by synthesis and matching. Pattern Recognit. 42 (7), 1445–1457.

Kemp, T., Schaaf, T., 1997. Estimating confidence using word lattices. In: Proceedings ECSCT, pp. 827–830.

Kimura, F., Takashina, K., Tsuruoka, S., Miyake, Y., 1987. Modified quadratic discriminant functions and the application to chinese character recognition. IEEE Trans. Pattern. Anal. Mach. Intell. 9 (1), 149–153.

Kumar, G., Wshah, S., Govindaraju V., Ramachandrula, S., 2013. Segmentation-free keyword spotting framework using dynamic background model. In: Proceedings SPIE 8658, Document Recognition and Retrieval XX, 86580H (February 4).

Kuo, S.-S., Agazzi, O.-E., 1994. Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. IEEE Trans. Pattern. Anal. Mach. Intell. 16 (8), 842–848.

Lafferty, J., McCallum, A., Pereira, F.-C., 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings ICML, pp. 282–289.

Liu, C.-L., Zhou, X.-D., 2006. Online Japanese character recognition using trajectory-based normalization and direction feature extraction. In: Proceedings IWFHR, pp. 217–222.

Liu, C.-L., Koga, M., Fujisawa, H., 2002. Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading. IEEE Trans. Pattern. Anal. Mach. Intell. 24 (11), 1425–1437.

Liu, C.-L., Yin, F., Wang, D.-H., Wang, Q.-F., 2011. CASIA online and offline Chinese handwriting databases. In: Proceedings ICDAR, pp. 37–41.

Lopresti, D., Tomkins, A., 1994. On the searchability of electronic ink. In: Proceedings IWFHR, pp. 156–165.

Manmatha, R., Han, C., Riseman, E.-M., 1996. Word spotting: a new approach to indexing handwriting. In: Proceedings CVPR, pp. 631–637.

McCallum, A., Freitag, D., Pereira, F., 2000. Maximum entropy Markov models for information extraction and segmentation. In: Proceedings ICML, pp. 591–598.

Messina, R., Louradour, J., 2015. Segmentation-free handwritten Chinese text recognition with LSTM-RNN. In: Proceedings ICDAR, pp. 171–175.

Myers, C.-S., Rabiner, L.-R., Rosenberg, A.-E., 1981. Use of dynamic time warping for word spotting and connected word recognition. Bell Syst. Tech. Journ. 60 (3), 303–325.

Ney, H., Ortmanns, S., Lindam, I., 1997. Extensions to the word graph method for large vocabulary continuous speech recognition. In: Proceedings IEEE International Conference Acoustics, Speech, Signal Processing, pp. 1787–1790.

Oda, H., Kitadai, A., Onuma, M., Nakagawa, M., 2004. A search method for online handwritten text employing writing-box-free handwriting recognition. In: Proceedings IWFHR, pp. 157–162.

Ortmanns, S., Ney, H., Aubert, X., 1997. A word graph algorithm for large vocabulary continuous speech recognition. Comput. Speech Lang. 11 (1), 43–72.

Plamondon, R., Srihari, S.-N., 2000. On-line and off-line handwriting recognition: a comprehensive survey. IEEE Trans. Pattern. Anal. Mach. Intell. 22 (1), 63–84.

Ploetz, T., Fink, G.-A., 2009. Markov models for offline handwriting recognition: a survey. Int. J. Doc. Anal. Recognit. 12 (4), 269–298.

Quiniou, S., Anquetil, E., 2007. Use of a confusion network to detect and correct errors in an on-line handwritten sentence recognition system. In: Proceedings ICDAR, pp. 382–386.

Rabiner, L.-R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77 (2), 257–286.

Rodriguez-Serrano, J.-A., Perronnin, F., 2009. Handwritten word-spotting using hidden markov models and universal vocabularies. Pattern Recognit. 42 (9), 2106–2116.

Rodriguez-Serrano, J., Perronnin, F., 2012. A Model-based sequence similarity with application to handwritten word-spotting. IEEE Trans. Pattern. Anal. Mach. Intell. 34 (11), 2108–2120.

Rueber, B., 1997. Obtaining confidence measures from sentence probabilities. In: Proceedings ECSCT, pp. 739–742.

Sarawagi, S., Cohen, W.-W., 2004. Semi-markov conditional random fields for information extraction. In: Advances in Neural Information Processing Systems, pp. 1185–1192.

Sarkar, S., 2013. Word spotting in cursive handwritten documents using modified character shape codes. Advances in Computing and Information Technology, Springer Berlin Heidelberg, pp. 269–278.

Sixtus, A., Ortmanns, S., 1999. High quality word graphs using forward backward pruning. In: Proceedings IEEE International Conference Acoustics, Speech, Signal Processing, pp. 593–596.

SogouLab: ⟨http://www.sogou.com/labs/resources.html⟩.

Su, T.-H., Zhang, T.-W., Guan, D.-J., Huang, H.-J., 2009. Off-line recognition of realistic Chinese handwriting using segmentation-free strategy. Pattern Recognit. 42 (1), 167–182.

Van der Zant, T., Schomaker, L., Haak, K., 2008. Handwritten-word spotting using biologically inspired features. IEEE Trans. Pattern. Anal. Mach. Intell. 30 (11), 1945–1957.

Wang, D.-H., Liu, C.-L., Zhou, X.-D., 2012. An approach for real-time recognition of online Chinese handwritten sentences. Pattern Recognit. 45 (10), 3661–3675.

Wang, Q.-F., Yin, F., Liu, C.-L., 2012. Handwritten Chinese text recognition by integrating multiple contexts. IEEE Trans. Pattern. Anal. Mach. Intell. 34 (8), 1469–1481.

Wang, Q.-F., Yin, F., Liu, C.-L., 2014. Unsupervised language model adaptation for handwritten chinese text recognition. Pattern Recognit. 47 (3), 1202–1216.

Wessel, F., Schluter, R., Macherey, K., Ney, H., 2001. Confidence measures for large vocabulary continuous speech recognition. IEEE Trans. Speech Audio Process 9 (3), 288–298.

Yang, H.-D., Sclaroff, S., Lee, S.-W., 2009. Sign language spotting with a threshold model based on conditional random fields. IEEE Trans. Pattern. Anal. Mach. Intell. 31 (7), 1264–1277.

Yin, F., Wang, Q.-F., Liu, C.-L., 2013. Transcript mapping for handwritten chinese documents by integrating character recognition model and geometric context. Pattern Recognit. 46 (10), 2807–2818.

Zhang, H., Wang, D.-H., Liu, C.-L., 2012. A confidence-based method for keyword spotting in online Chinese handwritten documents. In: Proceedings ICPR, pp. 525–528.

Zhang, H., Wang, D.-H., Liu, C.-L., 2013a. Keyword spotting from online chinese handwritten documents using one-vs-all character classification model. Int. J. Pattern. Recogn. Artif. Intell. 27 (3).

Zhang, Heng, Zhou, Xiang-Dong, Liu, Cheng-Lin, 2013. Keyword spotting in online Chinese handwritten documents with candidate scoring based on semi-CRF model. In: Proceedings ICDAR, pp. 567–571.

Zhang, H., Wang, D.-H., Liu, C.-L., Bunke, H., 2013b. Keyword spotting from online Chinese handwritten documents using one-versus-all character classification model. Int. J. Pattern. Recogn. Artif. Intell. 27 (3).

Zhang, Heng, Wang, Da-Han, Liu, Cheng-Lin, 2014. Character confidence based on N-best list for keyword spotting in online chinese handwritten documents. Pattern Recognit. 47 (5), 1880–1890.

Zhou, X.-D., Yu, J.-L., Liu, C.-L., Nagasaki, T., Marukawa, K., 2007. Online handwritten Japanese character string recognition incorporating geometric context. In: Proceedings ICDAR, pp. 23–26.

Zhou, X.-D., Wang, D.-H., Liu, C.-L., 2009. A robust approach to text line grouping in online handwritten Japanese documents. Pattern Recognit. 42 (9), 2077–2088.

Zhou, X.-D., Wang, D.-H., Tian, F., Liu, C.-L., Nakagawa, M., 2013. Handwritten Chinese/Japanese text recognition using semi-Markov conditional random fields. IEEE Trans. Pattern. Anal. Mach. Intell. 35 (10), 2413–2426.