2009 10th International Conference on Document Analysis and Recognition

Effect of Improved Path Evaluation for On-line Handwritten Japanese Text Recognition

Bilan ZHU †, Xiang-Dong ZHOU ‡, Cheng-Lin LIU ‡, and Masaki NAKAGAWA †

† Tokyo University of Agriculture and Technology, Tokyo 184-8588, Japan

‡ Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

† {zhubilan, nakagawa}@cc.tuat.ac.jp, ‡ {xdzhou, liucl}@nlpr.ia.ac.cn

Abstract

This paper describes a method of on-line handwritten Japanese text recognition by improved path evaluation. Based on a theoretical ground, the method evaluates the likelihood of candidate segmentation paths by combining scores of character pattern size, inner gap, character recognition, single and pair character position, candidate segmentation point and linguistic context, with the weight parameters optimized by a genetic algorithm. The path score is insensitive to the number of candidate patterns and the optimal path can be found by Viterbi search. Experimental results demonstrate the superiority of the proposed method.

1. Introduction

Due to the development of pen input devices such as tablet PCs, electronic whiteboards, PDAs, digital pens (like the Anoto pen) and so on, the increasing of continuous writing with less constraints entails handwritten text recognition rather than character recognition. Handwritten text recognition is usually performed by integrated character segmentation and recognition (character string recognition) because characters cannot be reliably segmented before they are recognized due to the irregularity of character size and spacing. On evaluating the possible paths of segmentation-recognition by combining the scores of character recognition, geometric features and linguistic context, the best path is searched to give the result of character segmentation and recognition [1][2].

In on-line Japanese text recognition, a stochastic model was proposed to evaluate the likelihood of segmentation-recognition paths [1]. The likelihood, depending on the number of segmented characters (segmentation length), favors short paths and so, tends to over-merge characters. Zhou et al. use a path score normalized by the segmentation length to overcome the over-merge [2]. Nevertheless, this normalized score favors longer strings, and so, tends to over-split characters. In hidden Markov model (HMM)-based text recognition, the path score depends on the fixed length of observation sequence, but the character shapes cannot be grasped well. Chen et al. proposed a variable duration HMM for handwritten word recognition, where the probability of a hypothesized character is weighted by the number of primitives composing it [3]. Yu et al. similarly use the number of primitives of character pattern for weighting the character recognition score in path evaluation [4]. However, they did not explain the theoretical ground of path evaluation and did not optimize the weight parameters.

In this paper, we propose a method of on-line handwritten Japanese text recognition by improving the path evaluation quality. Based on a theoretical ground, we evaluate the likelihood of candidate paths by combining scores of character pattern size, inner gap, character recognition, single-character position, pair-character position, candidate segmentation point and linguistic context. The weight parameters are optimized using a genetic algorithm. Since the path score remains cumulative with respect to the character string, the optimal path can be found by Viterbi search. In experiments in handwritten Japanese sentence recognition, the proposed method is shown to outperform previous ones.

2. Processing Flow

An on-line character string (a sequence of strokes) is processed in the following three steps.

(1) Over-segmentation. The strokes in a string are grouped into blocks (primitive segments) according to some the off-stroke (pen lift between two adjacent strokes) distance and the overlap of bounding boxes of adjacent strokes. Each primitive segment is assumed to be a character or a part of a character. The off-stroke between adjacent blocks is called a candidate segmentation point, which can be a true segmentation point (SP) or a non-segmentation point (NSP).

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.215

(2) Candidate lattice construction. One or more consecutive primitive segments form a candidate character pattern, and each pattern is associated with several candidate classes with scores assigned by character classification. The combination of all candidate patterns and character classes is represented by a candidate lattice (Fig. 1), where each node denotes a segmentation point and each arc denotes a character class assigned to a candidate pattern.

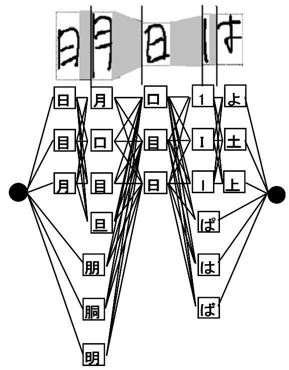

Fig. 1. Candidate lattice.

(3) Segmentation and recognition. The segmentation paths in the candidate lattice are evaluated by combining the scores of candidate character patterns and between-character compatibilities, and the optimal path is searched to give the result of character segmentation and recognition.

3. Recognition Model

In the candidate lattice, it is not appropriate to score the paths using posterior probabilities of characters, because different paths may have different numbers of characters. We herein present an evaluation model that combines multiple features and is theoretically independent of the length of segmentation paths.

3.1 Path Evaluation

Representing a character string pattern as a sequence of primitive segments: $\mathbf{X}=s_1...s_m$, which is partitioned into character patterns $\mathbf{Z}=z_1...z_n$, where each candidate pattern contains k_i primitive segments: $z_i = s_{j_i} \cdots s_{j_i+k_i-1}$. The segmented character patterns are assigned classes $\mathbf{C}=C_1...C_n$. To evaluate the score of string $\mathbf{X}$ in respect to string class $\mathbf{C}$, we extract features for scoring the primitive segments (or candidate patterns) and between-segment (or between-character) compatibilities:

- Bounding box feature b_i
- Inner gap feature q_i
- Shape feature s_i or z_i
- Unary position feature p^u_i of single segment (or character)
- Binary position feature p^b_i between adjacent segments (or characters)
- Between-segment gap feature g_i, which is to be classified as SP or NSP.

Denoting by $\mathbf{b}$, $\mathbf{q}$, $\mathbf{X}$, $\mathbf{p}^{\mathrm{u}}$, $\mathbf{p}^{\mathrm{b}}$, $\mathbf{g}$ for the sequences of features of primitive segments, the posterior probability of string class is given by:

$$P(\mathbf{C} \mid \mathbf{X}) = P(\mathbf{C} \mid \mathbf{b},\mathbf{q},\mathbf{X},\mathbf{p}^{\mathrm{u}},\mathbf{p}^{\mathrm{b}},\mathbf{g}) = \frac{P(\mathbf{b},\mathbf{q},\mathbf{X},\mathbf{p}^{\mathrm{u}},\mathbf{p}^{\mathrm{b}},\mathbf{g} \mid \mathbf{C})P(\mathbf{C})}{P(\mathbf{b},\mathbf{q},\mathbf{X},\mathbf{p}^{\mathrm{u}},\mathbf{p}^{\mathrm{b}},\mathbf{g})} \quad (1)$$

Omitting the string class-independent denominator and reasonably assuming independence between different features, the string class can be equivalently evaluated by

$$f(\mathbf{X},\mathbf{C}) = \log\left(P(\mathbf{b},\mathbf{q},\mathbf{X},\mathbf{p}^{\mathbf{u}},\mathbf{p}^{\mathbf{b}},\mathbf{g} \mid \mathbf{C})P(\mathbf{C})\right) = \log P(\mathbf{C}) + \sum_{i=1}^{m}\left[\begin{array}{l}\log P(b_i \mid c_i) + \log P(q_i \mid c_i) + \log P(s_i \mid c_i) \\ + \log P(p_i^u \mid c_i) + \log P(p_i^b \mid c_{i-1},c_i) + \log P(g_i \mid t_i)\end{array}\right] \quad (2)$$

where c_i denotes a character class or a hypothetical category of primitive segment (we call it hyper-category), and t_i denotes SP or NSP.

$P(\mathbf{C})$ is represented by the tri-gram of hyper-categories for primitive segments, and is approximated by the tri-gram of character categories:

$$\begin{aligned}\log P(\mathbf{C}) &= \sum_{i=1}^{m} \log P(c_i \mid c_{i-2}c_{i-1}) \\ &= \sum_{i=1}^{n}\left[\log P(c_{j_i} \mid c_{j_i-2}c_{j_i-1}) + \sum_{j=j_i+1}^{j_i+k_i-1} \log P(c_j \mid c_{j-2}c_{j-1})\right] \\ &\approx \sum_{i=1}^{n}\left[\lambda_{11}\log P(C_i \mid C_{i-2}C_{i-1}) + \lambda_{12}\sum_{j=j_i+1}^{j_i+k_i-1} P(C_i \mid C_{i-2}C_{i-1}) + \lambda_1\right] \\ &= \sum_{i=1}^{n}\left\{[\lambda_{11} + \lambda_{12}(k_i-1)]\log P(C_i \mid C_{i-2}C_{i-1}) + \lambda_1\right\}\end{aligned} \quad (3)$$

where λ_{11} and λ_{12} are weighting parameters, and λ_1 is a bias for balancing the number of characters. We approximate the transition probability of start segment of a character pattern and that of non-start segment using different weights for their varying effects.

Similarly, we use different weights for the start segment and non-start segment of the other features, obtaining the path score:

$$f(\mathbf{X},\mathbf{C}) = \sum_{i=1}^{n}\left\{\begin{array}{l}\sum_{h=1}^{6}[\lambda_{h1} + \lambda_{h2}(k_i-1)]\log P_h \\ \lambda_{71}\log P(g_{j_i} \mid SP) + \lambda_{72}\sum_{j=j_i+1}^{j_i+k_i-1}\log P(g_j \mid NSP)\end{array}\right\} + n\lambda \quad (4)$$

where P_h, h=1,…,6, stand for $P(C_i|C_{i-2}C_{i-1})$, $P(b_i|C_i)$, $P(q_i|C_i)$, $P(z_i|C_i)$, $P(p^u_i|C_i)$ and $P(p^b_i|C_{i-1}C_i)$, respectively.

The weighting parameters λ_{h1}, λ_{h2} (h=1~7) and λ are selected using a genetic algorithm to optimize the string recognition performance on a training dataset.

The path score in Eq. (4) is accumulated over the primitive segments, and hence, is insensitive to the number of segmented character patterns. Thus, the optimal path can be found by Viterbi search (dynamic programming).

The path evaluation scorer of [1] and that of [4] can be viewed as a special case of the proposed one in Eq. (4) by setting λ_{h1} =1, λ_{h2}=0 (h=1~7) and λ=0 for [1], and by setting $\lambda_{41}=\lambda_{42}$, $\lambda_{h2}=0$ (h=1~3, 5~7), and $\lambda=0$ for [4], respectively.

3.2 Evaluation of Terms

The tri-gram probability $P(C_i|C_{i-2},C_{i-1})$ is calculated on a text corpus. It is reduced to unigram or bi-gram when C_i is the first or second character of a sentence. The tri-gram is smoothed to overcome the imprecision of training with insufficient text [5]:

$$P'(C_i\,|\,C_{i-2},C_{i-1})=\beta_1 P(C_i\,|\,C_{i-2},C_{i-1})+\beta_2 P(C_i\,|\,C_{i-1})+\beta_3 P(C_i)+\beta_4 \quad (5)$$

where the weights (subject to $\beta_1+\beta_2+\beta_3+\beta_4=1$) are obtained by using a different text corpus.

The values of geometric features b_i, q_i, p^u_i and p^b_i are normalized with respect to the average character size *acs* for scale invariance. Several geometric features are shown in Fig. 2.

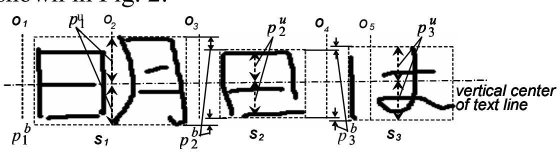

Fig. 2. Some geometric features.

The feature vector b_i comprises the height and width of each character pattern bounding box.

The feature vector q_i comprises six values as shown in Fig. 3. The first three values represent the horizontal gaps of three vertical slits (partitioned from vertical projection), and the last three ones represent the vertical gaps of three horizontal slits (from horizontal projection).

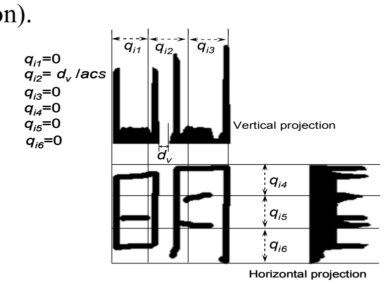

Fig. 3. Features of character pattern inner gap.

The feature vector p^u_i comprises the vertical lengths from the center line to the top and bottom of the bounding box. The feature vector p^b_i has two elements measured from the bounding boxes of two adjacent character patterns: the vertical distances between the upper bounds and between the lower bounds. $P(p^b_1|C_1,C_0)$ is set as 1. To reduce the cardinality of $P(p^b_i|C_{i-1}C_i)$, we cluster the character classes into six super-classes according to the mean vector of the unary position features of each class on training samples. $P(p^b_i|C_{i-1}C_i)$ is then replaced by $P(p^b_i|C'_{i-1},C'_i)$, where C'_{i-1},C'_i are the super-classes.

The geometric feature vectors b_i, q_i, p^u_i and p^b_i are transformed to log-likelihood scores (to be used in Eq. (4)) using quadratic discriminant function (QDF) classifiers.

The character shape score $P(z_i|C_i)$ is given by a character recognizer.

The feature vector g_i comprises multiple features measuring the relationship between two primitive segments adjacent to a candidate segmentation point [8]. We approximate $P(g_i|SP)$ and $P(g_i|NSP)$ using a SVM classifier. The SVM output is warped to obtain probabilities $P(o_i|SP)$ and $P(o_i|NSP)$, where o_i is the output of the SVM for g_i. The warping function is obtained from the distribution of SVM outputs on a validation dataset. $P(o_1|SP)$ is set as 1.

To warp the SVM outputs, we first obtain the histograms of outputs $P(o_i|SP)$ and $P(o_i|NSP)$, then take the cumulative probabilities $P'(o_i|SP)$ and $P'(o_i|NSP)$:

$$P'(o_i\,|\,SP)=\sum_{l=-\infty}^{O_i}P(l\,|\,SP) \qquad (6)$$

$$P'(o_i\,|\,NSP)=\sum_{l=O_i}^{\infty}P(l\,|\,NSP)$$

$P'(o_i|SP)$ and $P'(o_i|NSP)$ are then fitted by two sigmoidal functions, with the parameters estimated by minimizing squared errors.

4. Parameter Optimization

We train the weighting parameters λ_{h1}, λ_{h2} (h=1~7) and λ by a genetic algorithm using training data of character string patterns. To do this, we treat each one of λ_{h1}, λ_{h2} (h=1~7) and λ as an element of a chromosome.

(1) Initialization: Initialize N chromosomes with random values from 0 to 1, average fitness of the N chromosomes f_{old} as 0 and time t as 1.

(2) Crossover: Select two chromosomes at random from N chromosomes. Cross the elements between two random positions to produce two new chromosomes. Repeat until obtaining M new chromosomes.

(3) Mutation: Change each element of N+M chromosomes with a random value from -1 to 1 at a probability P_{mut}.

(4) Fitness evaluation: Evaluate fitness in terms of the recognition rate on training data with the weight values encoded in each chromosome.

(5) Selection: Decide the roulette probability of each chromosome according to its fitness. First select two chromosomes with the highest fitness, and then select chromosomes using the roulette until obtaining N new chromosomes. Replace the old N chromosomes with the new ones.

(6) Iteration: Obtain the average fitness of the new N chromosomes f_{new}. If (f_{new} - f_{old} < threshold) occurs n_{stop} times or $t > T$, return the chromosome of the highest fitness. Otherwise, set f_{new} to f_{old}, increment t, and go to step 2.

We set N as 50, M as 100, P_{mut} as 0.03, n_{stop} is as 25 and T as 10,000.

For evaluating the fitness of a chromosome, each training string pattern is searched the optimal path evaluated using the weight values in the chromosome. To save computation, we first set each weight value as 1, and select the top 100 recognition candidates for each training string. We then train the weight parameters by genetic algorithm using the selected 100 recognition candidates of each training string pattern. After some iterations, we use the updated weight values to re-select top 100 recognition candidates for each training string pattern. We repeat recognition candidate selection three times.

5. Experiments

For evaluating the proposed character string recognition model, we trained the character recognizer and geometric scoring functions using a Japanese on-line handwriting database Nakayosi [6][7]. The character recognizer combines off-line and on-line recognition methods by normalizing the recognition scores to conditional probabilities $P(s_i|C_i)$ [6]. For the geometric scores, four quadratic discriminant function (QDF) classifiers are trained for $P(b_i|C_i)$, $P(q_i|C_i)$, $P(p^u_i|C_i)$ and $P(p^b_i|C_{i-1},C_i)$, respectively.

For scoring linguistic context, we prepared an initial tri-gram table from the year 1993 volume of the ASAHI newspaper and the year 2002 volume of the NIKKEI newspaper. We estimated the smoothing parameters $\beta_1,\beta_2,\beta_3,\beta_4$ using the Nakayosi database. The data size of the tri-gram was reduced to 6MB by suppressing non-occurring terms, neglecting a small number of occurrences, and quantizing the logarithm values of tri-gram probabilities.

Table 1. Statistics of training/test text lines.

	#Text lines	#Character patterns	#Character classes	#Characters per line
Training	10,174	104,093	1,106	10.23
Testing	3,511	35,686	790	16.89

For training the weight parameters and evaluating the performance of character string recognition, we extracted horizontally written text lines from the database *HANDS-Kondate_t_bf-2001-11* collected from 100 people. We used 75 persons' text lines for training the SVM classifier for the candidate segmentation point probability and the weighting parameters of path evaluation score. After training, we used the text lines of the remaining 25 persons for testing. The statistics of the training and test are listed in Table 1.

We compare the performance of the proposed method and the ones presented in [1] and [2]. For fair comparison, all the three methods use the same tri-gram for language context and same classifiers for character recognition and geometric context. The weighting parameters were optimized using the genetic algorithm for each method. The three methods combines the same seven terms in Eq. (4) for path evaluation, but the method of [1] (Method 1) does not use the term related to k_i (number of primitive segments composing a character pattern), the method of [2] (Method 2) normalizes the path score of Method 1 using the number of segmented characters. The experiments were implemented on a Pentium (R) 4 2.80 GHz CPU with 512 MB memory.

For Method 2, we use beam search for finding the optimal paths in the candidate lattice, because the path score is not cumulative with the character sequence. For the proposed method and the Method 1, the optimal paths are found by the Viterbi search. For all the three methods, the candidate lattice retains 10 candidate classes for each character pattern.

We use a character segmentation measure f (F-measure of segmentation point detection), the character recognition rate C_r, and average string recognition time $T_{av\_rec\_tl}$ to evaluate the performance of text line recognition.

Table 2 shows the string recognition results of the three methods. For reference, the trained weight values of Eq. (4) are as follows: $(\lambda_{11},\lambda_{12},\lambda_{21},\lambda_{22},\lambda_{31},\lambda_{32},\lambda_{41},\lambda_{42},\lambda_{51},\lambda_{52},\lambda_{61},\lambda_{62},\lambda_{71},\lambda_{72},\lambda)=$ (0.351,0.000,0.265,0.001,0.199,0.000,1.000,0.641,0.009,0.000,0.100,0.000,0.323,0.120, 0.100). This indicates that except the character recognition score $P(z_i|C_i)$ and the non-segmentation point score $P(g_i|\text{NSP})$, the other geometric features and linguistic context are almost independent of the primitive number of character pattern.

Table 2. Results of text line recognition.

Performance \ Method		Method proposed	Method 1	Method 2
Training dataset	f	0.9941	0.9906	0.9850
	Cr	92.65%	91.68%	91.12%
	$T_{av\_rec\_tl}$	1.31 (s)	1.32 (s)	1.32 (s)
Testing dataset	f	0.9934	0.9903	0.9827
	Cr	92.80%	91.94%	91.07%
	$T_{av\_rec\_tl}$	1.32(s)	1.33 (s)	1.33 (s)

From the results, we can see that our model improves the character recognition and segmentation accuracy. For Method 1 depending on the number of segmented characters, a longer character sequence tends to have smaller evaluation score than a shorter one. For Method 2 the normalized score biases in favor of longer strings, and so, a character with multiple components tends to be split into multiple characters. Our model resolves these problems, and the path score is insensitive to the number of segmented characters.

The three methods take almost the same processing time.

Fig.4 shows some examples of misrecognition and missegmentations, where each upper line is the written text and the lower line is the recognition result.

(1) Problem of character recognition: Fig.4 (a), (b), (c), (d) and (e) show recognition errors due to character recognition, where the correct answers are not within the top 10 candidate classes output by the character recognizer for each character pattern. To solve this, we need to improve the character recognition accuracy.

(2) Problems of path evaluation and over-segmentation: Fig.4 (f), (g), (h) and (i) show recognition errors due to path evaluation and over-segmentation. Correct character answers are within the top 10 candidate classes but the path evaluation fails to find the correct one. To solve this, we should improve the accuracy of the linguistic context score and the geometric features scores, and that of over-segmentation.

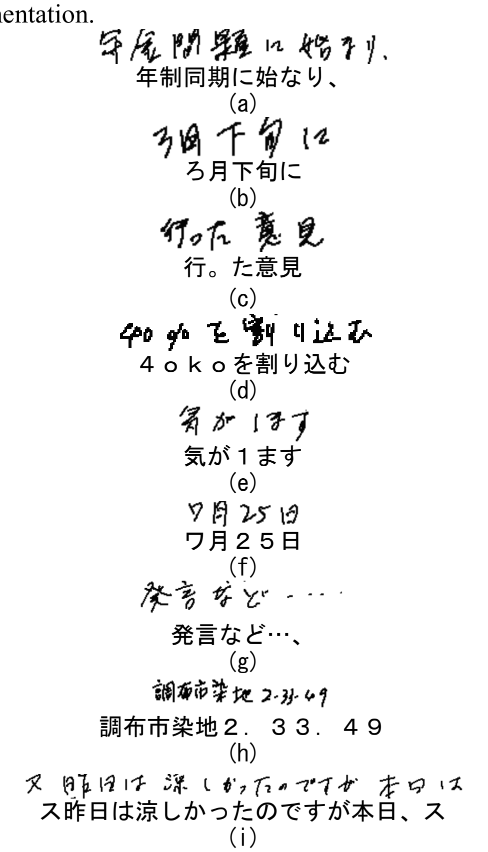

Fig. 4. Recognition errors. (i) has a segmentation error.

6. Conclusion

We presented a robust recognition model for on-line handwritten Japanese text. Our model evaluates the likelihood of candidate segmentation paths by combining multiple features such as geometric features, character recognition and linguistic context, being independent of the length of segmentation paths. The experiments show its superior performance. To further improve the performance, we will refine the over-segmentation module and better utilize the geometric context.

Acknowledgments

This research is partially supported by Grant-in-Aid for Scientific Research under the contract number (B)17300031 and the MEXT Research and Education Fund for Promoting Research on Symbiotic Information Technology.

References

[1] M. Nakagawa, B. Zhu and M. Onuma: A model of on-line handwritten Japanese text recognition free from line direction and writing format constraints, *IEICE Trans. Inf. & Syst.*, E88-D, 8, pp.1815-1822, 2005.

[2] X.-D. Zhou, J.-L. Yu, C.-L. Liu, T. Nagasaki and K. Marukawa: Online handwritten Japanese character string recognition incorporating geometric context, *Proc. 9th ICDAR, Curitiba,* Brazil, pp.48-52, 2007.

[3] M.-Y. Chen, A. Kundu and S. N. Srihari: Variable duration hidden Markov model and morphological segmentation for handwritten word recognition, *IEEE Trans. on Image Processing*, vol. 4, no. 12, pp.1675-1687, 1995.

[4] J.-L. Yu, X.-D. Zhou and C.-L. Liu: Search strategies in online handwritten character string recognition (in Chinese), *Proc. CCPR 2007 (Chinese Conference on Pattern Recognition),* China, pp.299-305, 2007.

[5] H. Mori and H. Aso: Postprocessing for Japanese document recognition based and 2nd order Markov model (in Japanese), *Technical Report of IPS J, NL*, 94, 63, pp.89-96, 1994.

[6] H. Oda, B. Zhu, J. Tokuno, M. Onuma, A. Kitadai and M. Nakagawa: A compact on-line and off-line combined recognizer, *Proc. IWFHR-10*, La Baule, France, pp.133-138, 2006.

[7] M. Nakagawa and K. Matsumoto: Collection of on-line handwritten Japanese character pattern databases and their analysis, *IJDAR*, vol.7, no.1, pp.69-81, 2004.

[8] B. Zhu and M. Nakagawa, Segmentation of on-line freely written Japanese text using SVM for improving text recognition, *IEICE Trans. Inf. & Sys.*, E91-D(1): 105-113, 2008.