A semi-incremental recognition method for online handwritten text

by

Cuong Tuan NGUYEN

A thesis submitted in conformity with the requirements
for the degree of

Doctor of Philosophy

in

Electronic and Information Engineering

Graduate School of Engineering
Tokyo University of Agriculture and Technology

Under supervision of Prof. Dr. Masaki NAKAGAWA

2016

A semi-incremental recognition method for online handwritten text

Cuong Tuan NGUYEN

Department of Computer and Information Sciences
Tokyo University of Agriculture and Technology

2016

Abstract

This thesis presents approaches for reducing waiting time and improve recognition accuracy in online handwritten text recognition. For explicit segmentation methods, the whole process of generating segmentation hypothesis, character/word recognition and searching for the best path of the candidate lattice incurs substantial waiting time as the length of the handwritten text increases. We employ local processing strategy which focuses on a recent sequence of strokes defined as "scope" to build and update a segmentation and recognition candidate lattice then advance the best-path search incrementally. Therefore, the method produces recognition results without noticeable waiting time while maintaining recognition rate. To reduce waiting time of segmentation free methods, we reduce the states in the decoding of handwritten text by modeling the decoding states as a state machine then applying reduction on it. Moreover, applying N-best states decoding method significantly reduces the waiting time and applying merging of recognition paths improves recognition accuracy of the system.

The methods are successfully applied in online handwritten recognition of Japanese and English text with explicit segmentation methods and the recognition of English text with segmentation free approach.

Acknowledgments

First of all, I would like to thank my supervisor: Prof. Masaki Nakagawa for his guidance and support. Next, I also would like to thank Dr. Bilan Zhu for providing me the recognition engine as well as technical support. Thanks my senior Dr. Truyen Van Phan for his guidance, my friends Anh Duc Le, Khanh Minh Phan, Hung Tuan Nguyen and all other members of Nakagawa laboratory for their assistances and help making a special place to work as well as enjoy the precious time here with me. Finally, I would like to thank my family for their constant encouragement and support.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

In recent years, due to the development of pen-based and touch-based devices such as tablets, digital pens (like the Anoto pen) and touch-based smart phones, on-line handwritten text recognition as an input method has been given considerable attention after a long period of research (Liu, Jaeger, and Nakagawa 2004; Plamondon and Srihari 2000; Graves et al. 2008). Since hand-held devices have relatively smaller CPU performance for less power consumption compared with desktop PCs and they are interactive devices, however, handwriting recognition on these devices must respond to user's input with high recognition rates but without incurring much CPU time.

As the use of context in the whole input sequence (e.g. geometric context, linguistic context) is important for online handwritten text recognition (Nakagawa, Zhu, and Onuma 2005; Graves et al. 2008), it is straightforward to recognize on-line handwritten text after the whole text is completed. Although, this strategy could achieve high recognition rates, waiting time of recognizing whole text takes time as the amount of characters increases. The problem of waiting time could be solved with the use of incremental recognition methods which recognize handwriting while users are writing. Incremental recognition does not incur long waiting time but it may degrade the recognition rate due to local processing of every stroke. Due to repeated processing after receiving every stroke, it also extend the total CPU time required for recognition.

Recognizing a whole input sequence without segmenting it into small parts takes advantage of recognition accuracy due to avoiding the errors in segmentation. The method, however, need to find the best solution path through a large amount of temporal input from the input sequence and the number of internal states that needs to be processed grows fast unless having a good constraint. As the number of internal states grows, processing time and memory consumption also increases, makes the recognition process infeasible to deal with long input sequence. Heuristic reduction of internal states tends to be too local and leads to drop recognition accuracy for dealing with the long input sequence which requires a more global solution.

1.1 Contributions

In this study, we focus on when incremental recognition processes are triggered. If a system triggers them whenever a new stroke is given, we classify it as pure incremental recognition. So far, all the published incremental recognition systems are classified in this group. However, we may trigger the processes by a little larger unit, i.e., several strokes so that we can exploit a little larger context. We classify this strategy as semi-incremental recognition. This thesis presents a semi-incremental recognition method of on-line handwritten text, which is useful for both the busy and the lazy recognition interfaces. Whenever the number of newly written strokes reaches the fixed number named the window size, the new strokes are added to the previous strokes, character patterns are segmented, candidate character patterns are recognized, a lattice representing segmentation and recognition candidates is updated, and search is processed, while writing continues. This process is repeated on recent strokes rather than on full text, so that text recognition result is shown immediately after writing is finished without noticeable waiting time while keeping a high recognition rate.

Although batch recognition achieves a high recognition rate with low total CPU time, it costs large waiting time as the amount of characters increases. On the contrary, pure incremental recognition incurs little waiting time but the recognition rate may drop due to local processing of every stroke and the total CPU time is extended due to repeated processing after receiving every stroke. Semi-incremental recognition with appropriate value of the window size may maintain high recognition rate as batch recognition, incur little waiting time and decrease the total CPU time compared with the pure incremental recognition.

In this work, we also present the improvement of segmentation which leads to the improvement of recognition accuracy and speed for online handwriting recognition system using recurrent neural networks (RNN).

For English text recognition, we also study the segmentation-free recognition method using the state of the art Long Short term Memory (LSTM), a type of RNN. We present a Finite State Machine based decoding method for LSTM, which does not only reduce the waiting time of recognition but also improves the recognition accuracy of the recognition system.

1.2 Overview of Thesis

This thesis presents about applying semi-incremental recognition method for online handwriting Japanese text and English text recognition, improvements for segmentation of English text recognition, decoding speed for segmentation-free English text recognition. Chapter 2 makes an overview about online handwriting recognition system and the recognition methods. Chapter 3 and Chapter 4 present the detail of semi-incremental recognition method and applying it to Japanese text recognition and English text recognition. Chapter 5 and Chapter 6 present the improvements of segmentation and recognition methods. Chapter 7 gives the conclusion and future works.

Chapter 2

Online handwritten text recognition

2.1 Overview of on-line text recognition system

In online handwriting recognition problem, trajectories of pen tip movement are recorded and analyzed to identify the linguistic information expressed (Liu, Jaeger, and Nakagawa 2004). Online handwriting recognition deals with the spatio-temporal representation of the input, whereas the offline case involves analysis of the spatio-luminance of an image (Plamondon and Srihari 2000).

Online handwriting text recognition deals with the problem of recognizing handwritten text including many text lines. For this problem, there are two approaches: explicit segmentation and implicit segmentation. In explicit segmentation method, the handwritten text is divided into text lines, then each text line is divided into smaller units (words or characters). The separated units are recognized and being combined to produce the text recognition result. In another hand, implicit segmentation approach does not require further segmentation.

2.2 Explicit segmentation approach

2.2.1 Soft decision

In recognition of on-line handwritten text with the explicit segmentation approach, there are two main tasks. First, an input sequence of strokes is segmented into smaller units as lines, words and characters. Second, the segmented units are recognized and the best path is searched to maximize the total score of segmentation and recognition.

Character segmentation is done based on geometric layout features. Due to the instability and ambiguity of these features in actual handwriting, however, it is difficult to determine segmentation without using recognition cues and linguistic context. Therefore, soft-decision is employed for segmentation and recognition. Then, the best path search is applied to perform segmentation and recognition. Namely, the following process is applied. Handwritten text is

segmented into text lines and each text line is over-segmented into primitive segments such that each segment is composed of a single character or a part of a character. A segment or a sequence of a few consecutive segments is assumed as a candidate character pattern, which is recognized by a character recognizer with a list of candidate categories and scores. Multiple ways of segmentation into candidate character patterns and multiple ways of recognition into character classes are represented in a segmentation-recognition candidate lattice (src-lattice in short) (Zhu et al. 2010). Text recognition result is produced by searching into the lattice for the path with the highest total score of geometric context, linguistic context and character recognition scores. Explicit segmentation approach with soft decision method is the state of the art in recognition of online handwritten Japanese text (Zhu et al. 2010), online handwritten Chinese text (Wang, Liu, and Zhou 2012). The method is also applied for online handwritten English text recognition (C. T. Nguyen, Zhu, and Nakagawa 2014).

2.2.2 Combination of online and offline recognizer

There are mainly two types of methods for recognizing a character or word pattern. An on-line method treats each pattern as a temporal feature sequence of pen movements while an off-line method processes each pattern as a two-dimensional image. An on-line method is robust against stroke connection and deformation but sensitive to stroke order variations or stroke duplications, while an off-line method is insensitive to the latter but weak for the former. The combination of the on-line and off-line recognition methods improves the recognition accuracy because they compensate for their disadvantages reciprocally (Oda et al. 2006; Zhu, Gao, and Nakagawa 2011; Liwicki et al. 2011).

The combination has also been made in the level of features. On-line recognition methods, which incorporate off-line features, and off-line methods, which include on-line features, also solve the problem of using on-line or off-line features alone, as shown in previous studies ((Zhu et al. 2013) and (Jaeger et al. 2001), respectively).

Although a combination of recognition methods or features improves the recognition rate, it requires more computation, incurring longer waiting time when it is used for batch recognition, especially for Japanese and Chinese with a large set of character categories.

For Japanese character recognition, Zhu et al. used the combination of an online recognition method based on Markov Random Fields (MRF) model with an offline recognition method using pseudo 2D bi-moment normalization (P2DBMN) and modified quadratic discriminant function (MQDF) (Zhu, Gao, and Nakagawa 2011).

The MRF online recognizer firstly extracts the feature points along the pen-tip trace from pen-down to pen-up of handwritten strokes. Then, it uses the feature point coordinates as unary features and the differences in coordinates between the neighboring feature points as binary features (Zhu and Nakagawa 2014). Each character class is modeled as a MRF model.

The MQDF recognizer extracts directional features from the normalized offline image of handwritten strokes. The normalization method of P2DBMN make recognition system more robust to difference handwriting styles or devices. Then, each character class is modeled using the parameters of mean and covariance matrix calculate from the training set. Recognition score is obtained by calculate the discriminant score with the learned parameters (Zhu, Gao, and Nakagawa 2011).

For recognizing on-line handwriting cursive word with segmentation-free approach, MRFs are constructed by concatenating character MRFs according to a trie lexicon of words during recognition. The system expands the search space using a character-synchronous beam search strategy to search the segmentation and recognition paths. With restricting of the search paths from the trie lexicon of words and preceding paths, as well as the lengths of feature points during path search. The evaluation function employing the recognition scores of both MRF character recognizer and P2DBMN-MQDF character recognizer yields a significant improvement in the recognition accuracy of English handwriting word recognition as compared with using only the recognition score of MRF recognizer (Zhu et al. 2013).

2.3 Segmentation free approach

2.3.1 Sequence to sequence learning

Segmentation free approach apply a connectionist system to handle the whole sequence of input directly. Time delayed neural network (TDNN) (Schenkel, Guyon, and Henderson 1995),

Hidden Markov Model (HMM) (Liwicki and Bunke 2006) and recently Long Short term Memory (LSTM) (Graves et al. 2009) are used for recognizing handwritten text. The method train the connectionist system to learn the temporal classification at each time step directly (TDNN, LSTM) or via the state transition (HMM). Learning process employs alignment methods to align the temporal output for each input time step as the use of forward backward algorithm for training HMM or Connectionist Temporal Classification (CTC) forward backward algorithm (Graves et al. 2006) for training LSTM.

2.3.2 Decoding

Decoding is a process of finding the most probable path from an input sequence. For state machine models, e.g. HMMs, it is straight forward to use Viterbi algorithm for decoding, it retains the best path reached to each state for each time step. Then the retained paths of the states are used for calculating in the next time step, the process is repeated until reaching the end of the time step (Viterbi 1967). The state machine models implicit the lexicon constraint while creating the word model or string model from character model. For decoding in RNN and LSTM, lexicon-free and lexicon-driven method can be applied. Lexicon-free decoding only use recognition probability to determine the most probable path, as it can be done trivially by selecting the most probable class label at each time step (best path decoding) or by calculating the total probability for each label sequence (prefix search decoding). Lexicon-driven applies constraint on output vocabulary as well as language model context for determine the best path, it can be done by a token passing algorithm. A token passing algorithm run on CTC called CTC token passing algorithm decodes the whole string with constraint of lexicon and linguistic context (Graves et al. 2009).

Chapter 3

Semi-incremental recognition method for Japanese text

This chapter presents a semi-incremental recognition method for on-line handwritten Japanese text and its evaluation. As text becomes longer, recognition time and waiting time become large if it is recognized after it is written (batch recognition). Thus, incremental methods have been proposed with recognition triggered by every stroke but the recognition rates are damaged and more CPU time is incurred. We propose semi-incremental recognition and employ a local processing strategy by focusing on a recent sequence of strokes defined as "scope" rather than every new stroke. For the latest scope, we build and update a segmentation and recognition candidate lattice and advance the best-path search incrementally. We utilize the result of the best-path search in the previous scope to exclude unnecessary segmentation candidates. This reduces the number of candidate character recognition with the result of reduced processing time. We also reuse the segmentation and recognition candidate lattice in the previous scope for the latest scope. Moreover, triggering recognition processes every several strokes saves CPU time. Experiments made on TUAT-Kondate database show the effectiveness of the proposed semi-incremental recognition method not only in reduced processing time and waiting time, but also in recognition accuracy.

The chapter is organized as follows: Section 3.1 is Introduction, Section 3.2 gives an overview of the baseline batch recognition method. Section 3.3 describes the semi-incremental recognition method. Section 3.4 presents experimental results of the semi-incremental recognition method. Section 3.5 draws our conclusion.

3.1 Introduction

As introduced in Chapter 1, the development of pen-based and touch-based devices makes research of on-line handwritten text recognition getting more attention recently. Along with the motivation of increasing recognition accuracy, the requirement of recognition speed and CPU burden in hand-held devices also induces the research for reduce waiting time and CPU time for recognition process.

For on-line handwritten text recognition, utilizing full context of the whole input sequence is important. Nakagawa et al. (Nakagawa, Zhu, and Onuma 2005) shows the effect of integrating linguistic context, character structure context along with recognition score to improve on-line handwritten Japanese text recognition. Graves et al. (Graves et al. 2009) shows the state-of-the-art on-line English handwriting recognition with the bi-directional recurrent neural networks which integrate context from both forward and backward directions.

To implement these methods, it is straightforward to recognize on-line handwritten text after the whole text is completed. We call this strategy as batch recognition (Zhu et al. 2010). Batch recognition is appropriate for the user interfaces where users are writing while thinking. In this case, users do not need recognition result when writing and they only need recognized text when they suspend writing. We call this user interface as lazy recognition interface (Nakagawa et al. 1993) while we call on-the-fly recognition user interfaces after every character or stroke is written as busy recognition interfaces. Here, a stroke is a sequence of finger-tip or pen-tip coordinates from finger/pen-down to finger/pen-up.

Although the batch recognition strategy can easily use the full context to achieve high recognition rates, waiting time to recognize the whole text input sequence at once takes time as the amount of characters increases. To reduce waiting time, incremental recognition method as in (Tanaka, Akiyama, and Ishigaki 2002) for Japanese text and (Wang, Liu, and Zhou 2012) for Chinese text, triggers recognition process after every new stroke is written. Each incremental recognition process a small portion of input stroke sequence and complete within small amount of time. Therefore, the result is feedback to the user with small amount of waiting time. Incremental recognition does not incur long waiting time after the user has finished writing, however, it may degrade the recognition rate due to local processing of every stroke. In fact, the recognition rate by incremental recognition decreased by about 0.3 point as compared with batch recognition (Tanaka, Akiyama, and Ishigaki 2002). Incremental recognition also extend the total CPU time due to repeated processing after receiving every stroke. Wang et al. report additional CPU time for each incremental recognition. Not only repetitive processes of triggering recognition are incurred, but also incomplete patterns are sought to be recognized at every stroke. Therefore, it takes a substantial total CPU time for recognizing a long input stroke sequence.

There are also two alternatives in the user interface of handwritten text recognition: busy or on-the-fly recognition and lazy or delayed recognition (Nakagawa et al. 1993) . A busy recognition interface shows the recognition result while a user is writing. It produces immediate feedback to the user but the user might be bothered by confirmation or correction of recognition. A predictive input interface (Matic, Platt, and Wang 2002), which predicts a character or word from a few beginning strokes, may be categorized as a busy recognition interface. On the other hand, a lazy recognition interface delays the output of the recognition result until needed. It is suitable for a user who is writing while thinking. The user does not need a recognition result when writing and only needs recognized text after he/she stops writing.

A lazy recognition interface can be implemented straightforwardly with the batch recognition method. Due to the problem of waiting time, however, the incremental recognition method in the background when a user is writing should be sought even for a lazy recognition interface when the problem of waiting time is serious.

As stated above, it is effective to use the full context for text recognition, i.e., from forward and backward directions. The backward context, which is the context caused by the succeeding strokes easy to obtain in the batch recognition method but not with the incremental recognition method, since succeeding strokes are unavailable. To use backward context, we should provide a way in which backward context affects the recognition of previous strokes.

In this work, we aim to solve the drawbacks and exploit the advantages of the batch recognition method and the incremental recognition method. We focus on maintaining the global context in incremental recognition and triggering recognition for a little larger unit called “scope”. We named this solution the semi-incremental recognition method while calling the method of triggering recognition at every stroke a pure incremental method. So far, all current incremental recognition systems are classified as pure. This chapter presents a semi-incremental recognition method of on-line handwritten Japanese text, which is useful for both the busy and the lazy recognition interfaces. Whenever the number of newly written strokes reaches the fixed number named the window size, the new strokes are added to the previous strokes, character patterns are segmented, candidate character patterns are recognized, a lattice representing segmentation and recognition candidates is updated, and search is processed, while writing continues. This process

is repeated on recent strokes rather than on full text, so that text recognition result is shown immediately after writing is finished without noticeable waiting time while keeping a high recognition rate.

Although batch recognition achieves a high recognition rate with low total CPU time, it costs large waiting time as the amount of characters increases. On the contrary, pure incremental recognition incurs little waiting time but the recognition rate may drop due to local processing of every stroke and the total CPU time is extended due to repeated processing after receiving every stroke. Semi-incremental recognition with appropriate value of the window size may maintain high recognition rate as batch recognition, incurs little waiting time and decreases the total CPU time compared with the pure incremental recognition.

3.2 Batch recognition method

This section describes the batch recognition method for on-line handwritten Japanese text. It processes the whole on-line handwritten text at once after all the strokes are added, it applies segmentation then constructs a candidate segmentation lattice, recognizes each candidate segment of strokes to build a segmentation recognition candidate lattice and finally employs context information to find the best recognition of handwritten text. Figure 3.1 shows the flow of the batch recognition method.

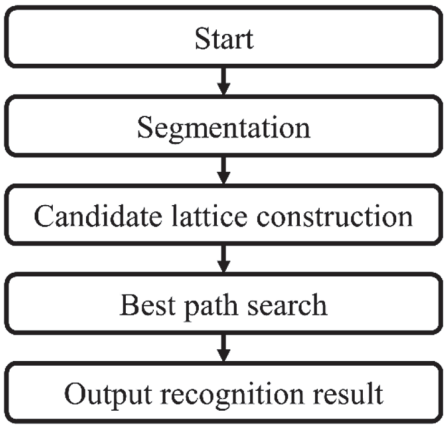


Figure 3.1 Flow of batch recognition

3.2.1 Segmentation

Using the technique presented in (Zhou, Wang, and Liu 2009), we first separate an input sequence of strokes into text lines. Then, we segment each text line into candidate character patterns as shown in Fig. 3.2. For over-segmentation, we apply the support vector machine (SVM) to classify each off-stroke into three classes, segmentation point (*SP*), non-segmentation point (*NSP*) and undecided point (*UP*) according to geometric features (Zhu et al. 2010). A segmentation point *SP* separates two characters at the off-stroke while a non-segmentation point *NSP* indicates the off-stroke is within a character. Off-strokes with low confidence are classified as *UP*. An off-stroke between two text lines is treated as *SP*. A sub-sequence of strokes delimited by *SP* or *UP* off-strokes is called a primitive segment. A primitive segment and consecutive primitive segments beside *UP* form candidate character patterns. Concatenation of consequent primitive segments is limited by their total lengths.

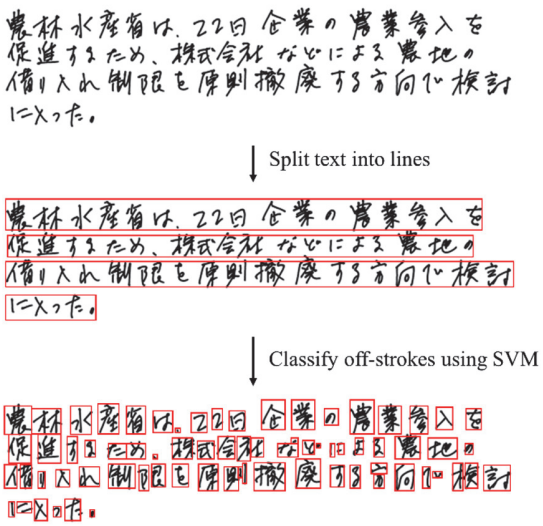


Figure 3.2 Segmentation process

Table 3.1 and Table 3.2 show terms to derive geometric features and geometric features derived, respectively. Since the entire input sequence of strokes is available, batch recognition can use both the contexts in forward and backward directions for segmentation. The features for determining segmentation at the current off-stroke are not only extracted from its immediate

preceding stroke and succeeding stroke but also from all the preceding strokes and succeeding strokes, which are highlighted in Tab. 3.2. A segmentation feature analysis in (Zhu and Nakagawa 2008) shows that the feature extracted from both the forward and backward contexts are useful for character segmentation.

Table 3.1 Terms of features

Feature	Definition
S_p	Immediate preceding stroke
S_s	Immediate succeeding stroke
B_p	Bounding box of S_p
B_s	Bounding box of S_s
B_{p_all}	Bounding box of all preceding strokes
B_{s_all}	Bounding box of all succeeding strokes
D_{b_x}	Distance between B_p and B_s in x-axis
D_{b_y}	Distance between B_p and B_s in y-axis
D_{B_x}	Distance between B_{p_all} and B_{p_all} in x-axis
O_b	Overlap area between B_p and B_s
O_B	Overlap area between B_{p_all} and B_{s_all}
D_{c_y}	Distance between centers of B_s and B_p in y-axis
D_{c_a}	Absolute distance of centers of B_p and B_s
acs	Average character size of text line
D_{t_y}	Distance between top of B_{p_all} and top of B_s in y-axis
P	Pattern of all strokes
P_{sub}	Sub-pattern of S_p and S_s

Table 3.2 Geometric features for character segmentation

F1	Passing time for off-stroke
F2	D_{B_x} / *acs*
F3	D_{b_x} / width of B_p
F4	D_{b_x} / width of B_s
F5	D_{b_x} / *acs*
F6	D_{b_y} / height of B_p
F7	D_{b_y} / height of B_s
F8	D_{b_y} / *acs*
F9	O_B / *(acs)*^{2}
F10	O_b / *(*width of B_s * height of B_s*)*
F11	O_b / *(acs)*^{2}
F12	D_{c_y} / *acs*
F13	D_{c_a} / *acs*
F14	D_{t_y} / *acs*
F15	Length of off-stroke / *acs*
F16	Sine value of off-stroke
F17	Cosine value of off-stroke
F18	F2 / maximum F2 in text

3.2.2 Candidate lattice construction

Employing the combination of on-line and off-line recognition methods for character recognition (Zhu, Gao, and Nakagawa 2011), each candidate character pattern is associated with a number of candidate classes with confidence scores. All the possible segmentations and recognition candidate classes are represented by a src-lattice as shown in Fig. 3.3, where each node denotes a candidate segmentation point and each arc denotes a character class assigned to a candidate character pattern.

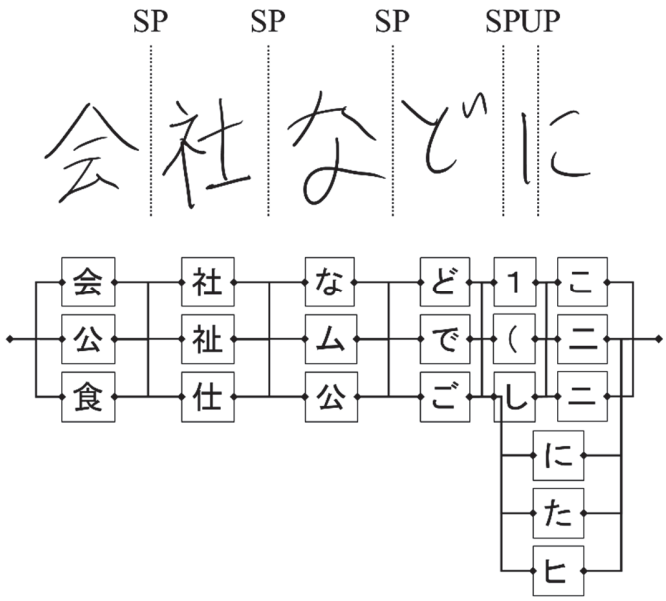


Figure 3.3 Segmentation-recognition candidate lattice

For implementation, we employ candidate character blocks and each of them represents a set of all the candidate character patterns separated by two adjacent SP off-strokes. Figure 3.4 shows them for the src-lattice with two SP off-strokes and three candidate character blocks.

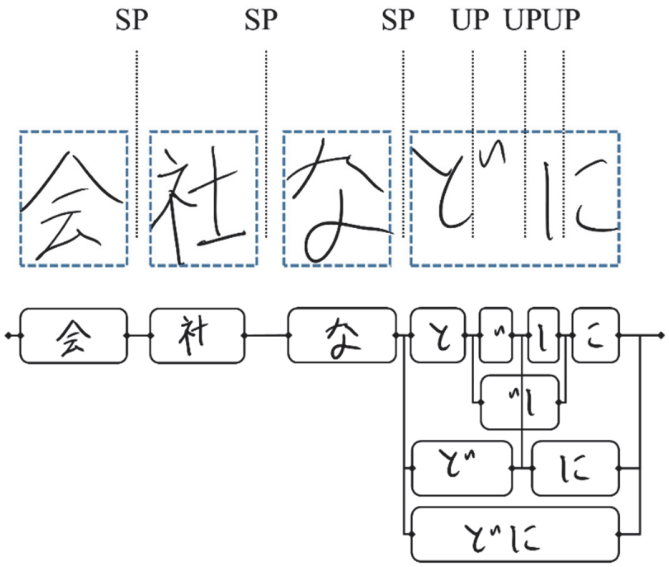


Figure 3.4 Candidate character blocks

3.2.3 Best-path search and recognition

From an src-lattice, paths are evaluated by combining the scores of character/word recognition, geometric features, and linguistic context (Zhu et al. 2010). We apply the Viterbi algorithm to search for the optimal path that has the highest evaluation score and obtain the text recognition result.

For evaluating a path through a sequence of *m* primitive segments $S = s_1, s_2, ..., s_m$ of an input sequence X, forming a sequence of *n* candidate character/word patterns $Z = z_1, z_2, ..., z_n$ which is assigned as $C = c_1, c_2, ..., c_n$, we have the posterior probability as follows:

$$\begin{aligned} P(C \mid X, S, Z) &= \frac{P(X, S, Z \mid C)P(C)}{P(X, S, Z)} \\ &= \frac{P(S \mid X, Z, C)P(X, Z \mid C)P(C)}{P(X, S, Z)} \end{aligned} \tag{3.1}$$

We omit the class-independent denominator to obtain the following formula:

$$P(C \mid X,S,Z) \propto P(S \mid X,Z,C)P(X,Z \mid C)P(C) \tag{3.2}$$

From the posterior probability, we obtain the evaluation function as:

$$f(X,S,Z,C) = \log P(S \mid X,Z,C) + \log P(X,Z \mid C) + \log P(C) \tag{3.3}$$

The probability $P(X,Z \mid C)$ is approximated using the following features extracted from candidate character/word patterns of an input sequence X :

- Geometric features: Width and height of bounding boxes $B = b_1, b_2, ..., b_n$, inner gap $Q = q_1, q_2, ..., q_n$, unary position $P^U = p_1^u, p_2^u, ..., p_n^u$ and binary position $P^B = p_1^b, p_2^b, ..., p_n^b$.

- Shape features $R = r_1, r_2, ..., r_n$

Since the probabilities of the features are conditionally independent, we obtain the following formula:

$$\begin{aligned} P(X,Z \mid C) &= P(B,Q,P^U,P^B,R \mid C) \\ &= P(B \mid C)P(Q \mid C)P(P^U \mid C)P(P^B \mid C)P(R \mid C) \end{aligned} \tag{3.4}$$

Assuming the independence of the features for each character/word pattern or for two consecutive character/word patterns, we obtain the formula:

$$P(X,G \mid C) = \prod_{i=1}^{n} \begin{bmatrix} P(b_i \mid c_i)P(q_i \mid c_i)P\left(p_i^u \mid c_i\right) \\ \times P\left(p_i^b \mid c_{i-1}c_i\right)P\left(r_i \mid c_i\right) \end{bmatrix} \tag{3.5}$$

where b_i, q_i, p_i^u, p_i^b are the values of geometric features derived from a candidate character/word pattern. The likelihood probabilities $P(b_i \mid c_i)$, $P(q_i \mid c_i)$, $P\left(p_i^u \mid c_i\right)$ and $P\left(p_i^b \mid c_{i-1}c_i\right)$ are obtained from the features using quadratic discriminant (QDF) classifiers. The likelihood probability $P\left(r_i \mid c_i\right)$ of a class c_i with respect to a candidate character/word pattern g_i is calculated using the recognizer presented in Sect. 2.2.2.

The linguistic context probability $P(C)$ is estimated using a tri-gram language model with back-off weight:

$$P(C) = \prod_{i=1}^{n} P(c_i \mid c_{i-2}c_{i-1}) \tag{3.6}$$

The segmentation probability $P(S \mid X, G, C)$ actually does not depend on character/word classes C and therefore it is approximated by the score from a segmentation classifier at each candidate segmentation point d_j (*SP* or *UP*) between two primitive segments s_j and s_{j+1}:

$$P(S \mid X, G, C) = \prod_{j=1}^{m-1} P(d_j \mid X, G) \tag{3.7}$$

Each candidate segmentation point d_j could be an off-stroke between character/word patterns or an off-stroke within a character/word pattern. We denote *T* the labeling function outputting the type of off-stroke (*B*: between, *W*: within) for a candidate segmentation point.

$$P(S \mid X, G, C) = \prod_{j=\overline{1,m-1};T(d_j)=B} P_{sp}(d_j) \times \prod_{j=\overline{1,m-1};T(d_j)=W} P_{nsp}(d_j) \tag{3.8}$$

with $P_{sp}(d_j)$ and $P_{nsp}(d_j)$ are the classification probabilities of an off-stroke being classified as *SP* and *NSP*, respectively.

The evaluation function is expressed as:

$$\begin{aligned} f(X, S, G, C) = & \sum_{i=1}^{n} \left\{ \sum_{h=1}^{6} \left[\lambda_{h1} + \lambda_{h2} (k_i - 1) \right] \log P_h \right\} \\ & + \lambda_{71} \sum_{j=\overline{1,m-1};T(d_j)=B} \log P_{sp}(d_j) \\ & + \lambda_{72} \sum_{j=\overline{1,m-1};T(d_j)=W} \log P_{nsp}(d_j) + n\lambda \end{aligned} \tag{3.9}$$

where $P_h\left(h = 1,...,6\right)$ denote the probabilities of $P(c_i \mid c_{i-2}c_{i-1})$, $P(b_i \mid c_i)$, $P(q_i \mid c_i)$, $P\left(p_i^u \mid c_i\right)$, $P\left(p_i^b \mid c_{i-1}c_i\right)$, and $P_r\left(r_i \mid c_i\right)$, respectively, k_i denotes the number of primitive segments contained in the candidate character pattern g_i. The weighting parameters $\lambda_{h1}, \lambda_{h2} (h = \overline{1,7})$ and λ are selected using genetic algorithm to optimize the text recognition performance on a training dataset.

3.3 Semi-incremental recognition method

The main objective to develop the semi-incremental recognition method is to perform possible computation as much as possible while a user is writing. Moreover, it should keep the recognition rate as high as possible compared with the batch recognition method. In the batch recognition, the majority of computing time is spent for the recognition of candidate character patterns. If those can be processed in the background of user's handwriting, text recognition result will be displayed without any noticeable waiting time.

Both of the methods in (Tanaka, Akiyama, and Ishigaki 2002) and (Wang, Liu, and Zhou 2012) proposed incremental segmentation and recognition of handwritten text. For incremental segmentation, they determine the segmentation of the latest stroke based on the previously segmented sequence of strokes. This is unrecoverable, however, if wrong segmentation is made without knowing strokes coming hereafter. To deal with the problem, we present the resuming strategy, which allows the change of segmentation and recognition of previously written strokes due to strokes coming in future.

3.3.1 Resuming strategy

The context at the current time step in incremental recognition presented so far is limited to forward direction, since the future input is not available. As batch recognition could employ the context in both forward and backward direction for segmentation and recognition, we need to develop a method which allows backward context in incremental recognition.

Semi-incremental recognition performs recognition process after receiving Ns newly written strokes, where Ns denotes the number of strokes to trigger incremental recognition. Ideally, we only have to process the newly received strokes and update the src-lattice. In fact, the backward context of the newly added strokes affects the segmentation and recognition of a small number of strokes previously received. Therefore, to allow the effect of the backward context, we extend the incremental processing to the section that includes these strokes and the newly received strokes. We call it "scope".

We advance segmentation and recognition as new strokes are input by resuming segmentation from a range of recent input strokes since their segmentation may change due to the newly added strokes. We proceed with the pointer being slightly behind the latest input stroke up to which the result of segmentation and character recognition is considered stable and fixed.

Newly added strokes may affect the segmentation and recognition in the current scope. Change of segmentation leads to change of character recognition. As each change in classification of an off-stroke induces change in recognition of primitive character segments connecting to that off-stroke, we must update the scope to include those primitive character segments.

As for the best-path search, it is made from the first stroke to the last stroke in the batch recognition while it can be made incrementally using scope. Therefore, if the scope is well defined, the semi-incremental recognition should produce almost the same recognition result without incurring much waiting time.

3.3.2 Processing flow

From the previously described strategy, Fig. 3.5 shows the processing flow of the semi-incremental recognition method.

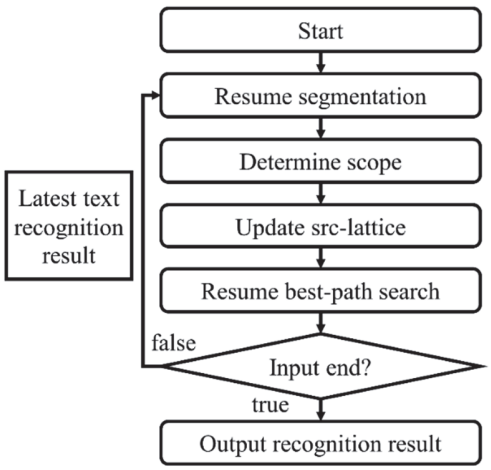


Figure 3.5 Flow of semi-incremental recognition

First, we receive new strokes. Secondly, we resume segmentation. Thirdly, we determine the scope. Fourthly, we update the src-lattice. Finally, we resume the best-path search from the beginning in this scope to get text recognition result. The result is used for next processing cycle.

3.3.3 Resumption of segmentation

First, we determine the pointer to resume segmentation named *Seg_rp*. It must be determined so that the segmentation and recognition of its preceding strokes is stable. In the result of text recognition up to the latest scope, i.e., the best-path up to the latest scope in the src-lattice, an off-stroke between two recognized characters can be considered as an SP with confirmation of the best path search. Since the effect of backward context of newly added strokes to the segmentation and recognition of preceding strokes reduces as they are far away from the latest stroke in backward direction, the segmentation at an off-stroke between two recognized characters far enough from the latest stroke could be fixed. We determine *Seg_rp* among those off-strokes based on the number of characters from each off-stroke to the last character in the recognition result. If this number equals to a predefined parameter named N_{seg}, that off-stroke will be determined as a new *Seg_rp*. N_{seg} is defined as a fixed number of characters required to determine a new *Seg_rp*.

3.3.4 Fixation of SP off-strokes from UP off-strokes

Determination of off-strokes to *SP* off-strokes has large effect to the recognition rate and performance of the system. Although *SP* off-strokes are detected by SVM in the segmentation process, the performance of SVM for detecting *SP* off-strokes is still limited. Due to the uncertainty of segmentation, a large number of outputs from SVM are marked as *UP*. Each *UP* roughly doubles the number of candidate character patterns for which character recognition is applied. To overcome this problem, we also use the result of text recognition up to the latest scope to determine *UP* off-strokes (*UP*s in short) to *SP* off-strokes (*SP*s in short). We call this process *UP* fixation. *UPs* between recognized characters, before the latest N_{seg_det} characters in the recognition result are determined as *SPs*. Here, N_{seg_det} denotes a predefined constant for the minimum number of characters that follow an *UP* off-stroke to make it a stable *SP* off-stroke. Generally, N_{seg_det} is smaller than or equal to N_{seg}.

3.3.5 Determination of scope

To determine the scope, we use the result from the segmentation process. The segmentations of the strokes before and after the system has received new strokes are compared with each other. If classification-changed off-strokes are detected, we consider the strokes before the earliest classification-changed off-strokes are stably classified while the strokes after that are not classified stably. Otherwise, the off-stroke before the newly added strokes is considered as the earliest classification-changed off-stroke. This earliest classification-changed off-stroke may occur within some candidate character block or between two candidate character blocks. We define the scope as the sequence of strokes starting from the first stroke of the candidate character block containing or just preceding the earliest classification-changed off-stroke to the last stroke.

3.3.6 Bounded waiting time

Since we resume segmentation from *Seg_rp*, segmentation of the off-strokes before *Seg_rp* remains unchanged. Thus, we only need to compare the results of segmentation before and after new strokes are added from *Seg_rp* to the latest stroke.

Since classification changes of off-strokes occur between *Seg_rp* and the latest stroke, the scope extends in backward direction at most one candidate character block from *Seg_rp*. Thus, by setting fixed N_{seg}, the maximum number of characters in the scope is bounded by (N_{seg} + 1) plus the number of new characters in the newly added strokes. This is the main factor to bound the waiting time in each processing. Moreover, changing more *UPs* to *SPs* in lattice blocks also reduces the time cost to rebuild the src-lattice due to shortened block sizes.

3.3.7 An example of the processes

Figure 3.6 shows an example to determine the scope**.** Assume the latest scope with segmentation and text recognition result in Fig. 3.6(a). Then, the new strokes marked red are added. We update *Seg_rp* and apply segmentation from the updated *Seg_rp* (Fig. 3.6(b)). Next we change *UPs* to *SPs* by *UP* fixation and find the earliest classification-changed off-stroke (Fig. 3.6(c)). Finally, we locate the character block including or just preceding this off-stroke and update the scope (Fig. 3.6(d)).

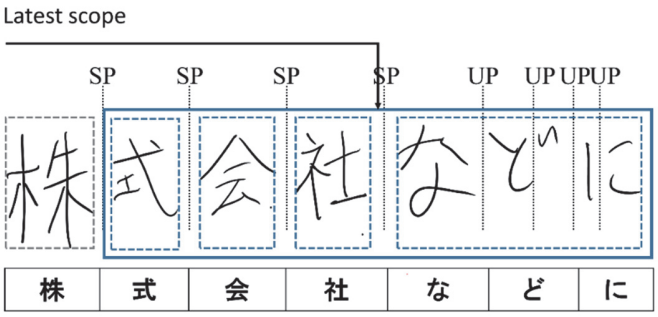


Figure 3.6 a). Latest scope with segmentation and text recognition results

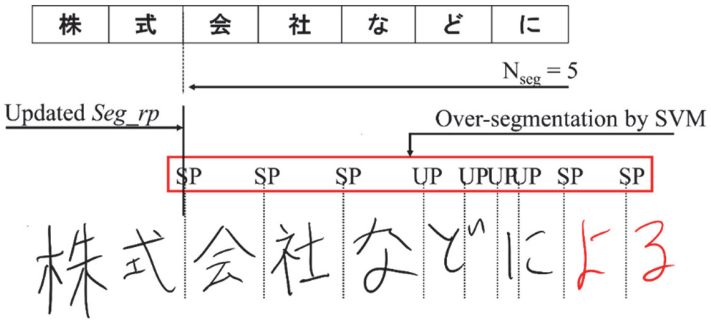


Figure 3.6 b). Receiving new strokes, updating *Seg_rp* and applying segmentation

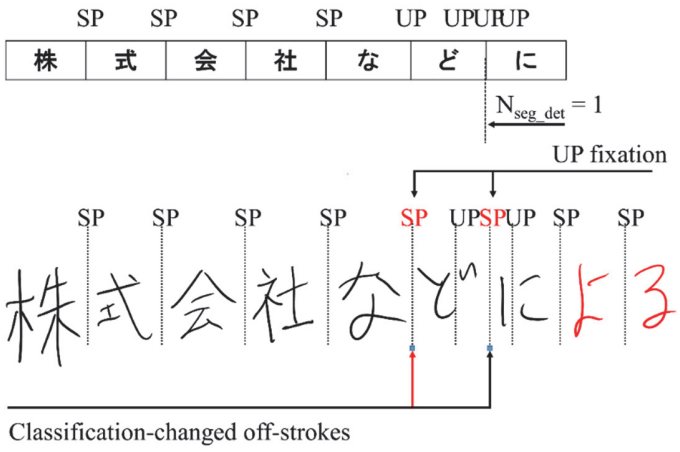


Figure 3.6 c). Determining *UP*s to *SP*s and finding classification-changed off-strokes

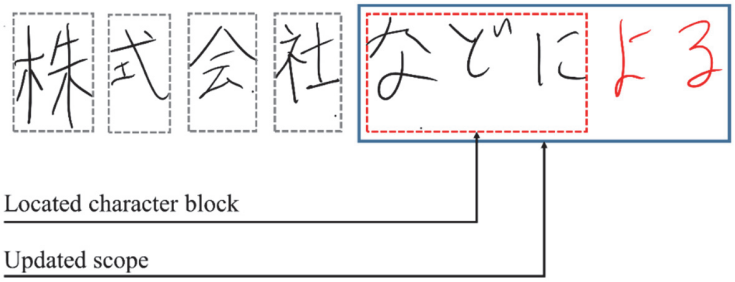


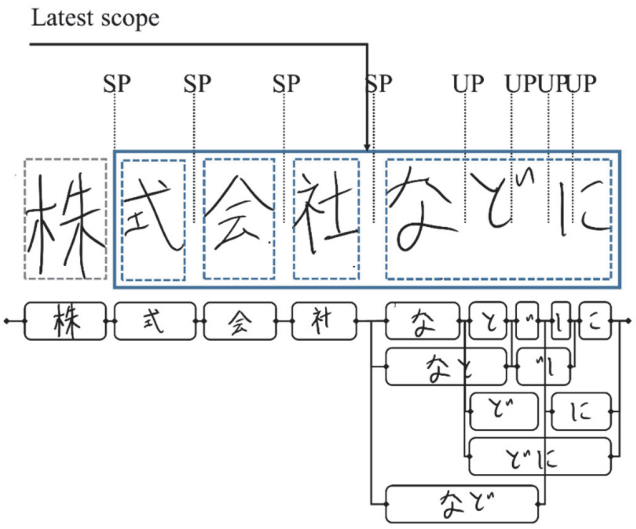
Figure 3.6 (d). Locating the character block and updating the scope

Figure 3.6 An example of determining scope

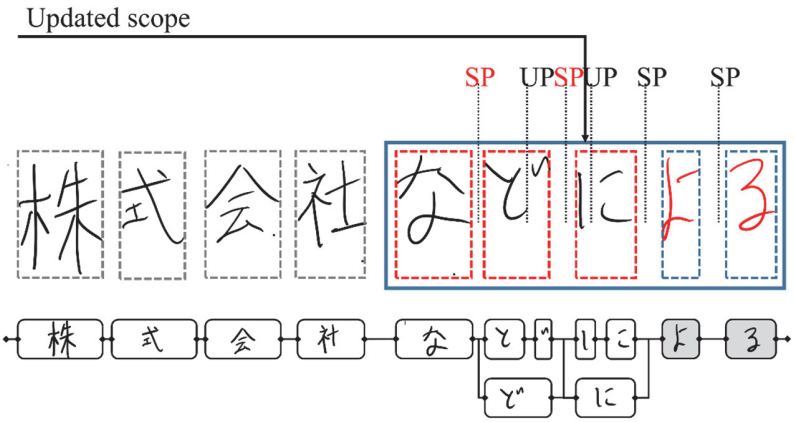
3.3.8 Update of src-lattice

Since each incremental recognition processes a scope of recent strokes, there are overlap between previous scope and current scope. In this overlap region, there are unchanged candidate character patterns which can be reused for updating the current scope. To maximize the reuse of the src-lattice in the previous scope, we use the following method. It takes advantage of previously built lattice candidates in the previous scope. From the beginning of the scope, the method finds *SPs* and splits candidate character blocks by these *SPs*. Each *SP* off-stroke divides a candidate character block into two parts: the preceding part and the succeeding part beside this *SP* off-stroke. The src-lattice in these lattice blocks will be checked if a candidate character pattern already exists in the previous scope. When exists, we get it from the previous scope, otherwise we rebuild it.

Figure 3.7(a) represents the lattice blocks of the previous scope, when new strokes are added as shown in Fig. 3.7(b), classification of the off-stroke between the two first characters in the updated scope is changed to *SP*. From this *SP* off-stroke, the previously built candidate character block is divided into three candidate character blocks and the candidate character patterns of the previous scope is reused for the updated scope. Then, only two candidate character patterns (shown in gray) are rebuilt due to the new strokes.



(a) Previous scope



(b) Updated scope

Figure 3.7 Reuse of candidate character patterns

3.3.9 Skipping partial patterns

Recognition of partial character patterns can be postponed until the complete character patterns are received. Therefore, we skip recognizing them to reduce CPU time. We treat candidate

character patterns containing the last primitive segment as partial candidate character patterns (PPs) until a new primitive segment is detected or the recognition is requested. We call this process PP skip.

3.3.10 Handling of delayed strokes

The batch recognition system (Zhu et al. 2010) is not designed to handle delayed strokes, since the segmentation of strokes is based on writing order. To make the segmentation of a text line including them correctly, however, we first detect delay strokes and ignore them in the segmentation process. Then, we determine a segmented block for each delayed stroke to merge the delayed stroke into it. Finally, we rebuild the src-lattice.

Delayed strokes are detected using the previous recognition result. Firstly, we retrieve the bounding box for each recognized character from the segmentation-recognition result of the previous scope. Then, we determine each newly added stroke as a delayed stroke if it is close to the previous bounding boxes rather than the latest bounding box.

When delayed strokes occur, we rebuild the src-lattice in two steps: first we build the src-lattice without delayed strokes, second we put delayed strokes into appropriate primitive segments and rebuild the candidate character patterns containing the delayed strokes.

3.3.11 Resuming best-path search and recognition

From the first character lattice block in current scope, we resume the best-path search and get text recognition result. Resuming the best-path search at each incremental processing sets bounds to the processing time and waiting time. This solves the drawback of the method in (Tanaka, Akiyama, and Ishigaki 2002), in which the processing time for the best-path search is prolonged as the number of characters increases.

3.4 Experiments

3.4.1 Measures for evaluation

First, over-segmentation is applied then segmentation is determined along with character recognition and best-path search. The over-segmentation process classifies each off-stroke as an *SP*, *NSP*, or *UP* off-stroke. An *UP* off-stroke can then be further classified as an *SP* or *NSP* in the text recognition process. Let *#SP*, *#NSP*, *#UP* are the numbers of returned *SP*s, *NSP*s and *UP*s, respectively. $\#SP_c$ is the number of correctly classified *SP*s among the returned SP_s. $\#SP_t$ is the number of true *SP*s defined in the ground truth.

The performance of over-segmentation is evaluated with the following measures.

Precision (*p*):

$$p = \frac{\#SP_c}{\#SP} \tag{3.10}$$

Recall (*r*):

$$r = \frac{\#SP_c + \#UP}{\#SP_t} \tag{3.11}$$

Inclusion of *#UP* in the dividend is typical for over-segmentation since *UP*s maintain the possibility that they will be classified correctly.

F-measure (*f*) is calculated as follows:

$$f = \frac{2 * p * r}{(p + r)} \tag{3.12}$$

Although *UP*s maintain the possibility that they will be classified correctly, thus increase recall, leaving many *UP*s instead of *SP*s or *NSP*s, however, incurs more waiting time as analyzed in Sect. 3.4. Therefore, we evaluate the detection rate (*d*) of over-segmentation as ability of determining more *SP*s instead of *UP*s by the following formula:

$$d = \frac{\#SP}{(\#SP + \#UP)} \tag{3.13}$$

As final segmentation is determined from the result of the best-path search, we get *SP*s as off-strokes between two recognized characters and the remaining are *NSP*s. Let $\#SP_f$, $\#SP_{fc}$ and $\#SP_{ft}$ are the number of returned *SP*s in final segmentation, the number of correctly classified *SP*s among those returned *SP*s and the number of true *SP*s in the ground truth, respectively. The F-measure of final segmentation denoted as the segmentation measure is evaluated as follows:

$$F = \frac{2 * P * R}{(P + R)} \tag{3.14}$$

where *P* and *R* are the precision and recall of final segmentation, respectively. They are defined as follows:

$$P = \frac{\#SP_{fc}}{\#SP_f} \tag{3.15}$$

$$R = \frac{\#SP_{fc}}{\#SP_{ft}} \tag{3.16}$$

3.4.2 Setup for experiments

We trained the character recognizer and geometric scoring functions using Japanese on-line handwriting database Nakayosi (Nakagawa and Matsumoto 2004). We employed a trigram table extracted from the year 1993 volume of the Asahi newspaper and the year 2002 volume of the Nikkei newspaper to model linguistic context. For training the weight parameters of the evaluation function (1) and evaluating the performance of text recognition, we used horizontally written text line patterns extracted from the TUAT-Kondate database collected from 100 people (Matsushita and Nakagawa 2014). We separated the text lines into 4 sets by writers and then used 3 sets (10,174 text lines written by 75 people) for training and 1 set (3,511 text lines written by 25 people) for testing. We changed the role four times and took the average. We used this separation to assure writer independence and conducted cross validation to evaluate the effect unbiased to data sets.

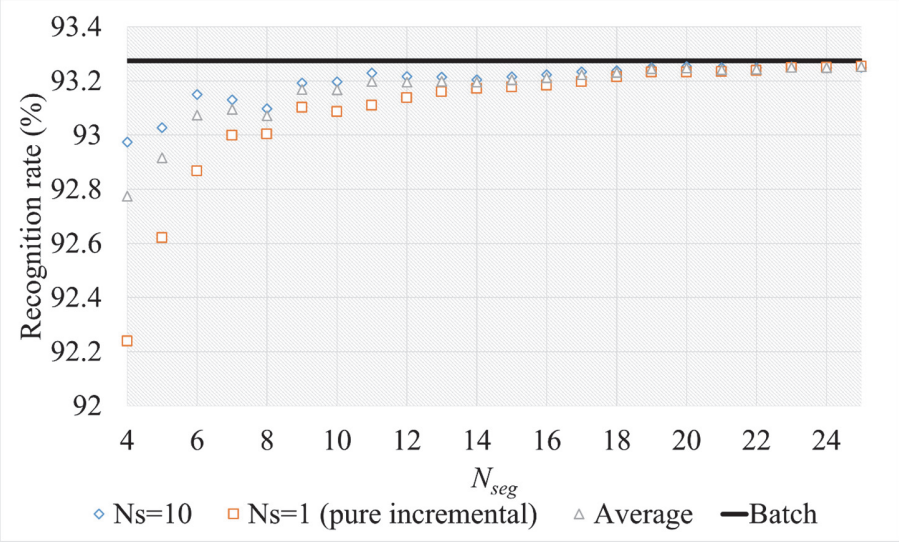


Figure 3.8 Recognition rate with respect to N_{seg}

The parameters of the evaluation function in Eq. (3.9) have been trained using each training set, but N_s and N_{seg} are not trained since N_s and N_{seg} are control variables rather than parameters.

We implemented a handwritten Japanese text recognition system using our semi-incremental recognition method. We used the batch recognition system for Japanese (Zhu et al. 2010) without modification. We ran all the systems on an Intel(R) Core(TM) i7 CPU 870@ 2.6Ghz with 4-GB memory.

3.4.3 Character recognition rate

Figure 3.8 shows the character recognition rate (i.e., the number of correctly recognized characters over that of all the characters in handwritten text) by the semi-incremental recognition method, including the case of pure-incremental recognition (N_s = 1) in comparison with the batch recognition method. To evaluate the effect of resuming segmentation to the character recognition rate, we made experiments with respect to N_s (from 1 to 10) and N_{seg} (from 4 to 25). For each N_{seg}, the character recognition rate increases as N_s increases from 1 to 10. With larger N_{seg}, the average character recognition rate increases, since the segmentation and recognition are resumed from a more stable *Seg_rp*. As N_{seg} approaches 20 and 25, the recognition rate reaches the performance

of batch recognition without large dependence on N_s (rates with N_s = 10 and those with N_s = 1 are nearly the same). The maximum recognition rate is 93.26% with N_{seg} = 20, which is nearly the same as that of the batch recognition method, i.e., 93.27%. The case of pure-incremental recognition (N_s = 1) degrades the recognition rate seriously with small values for N_{seg} (0.8 point when N_{seg}=4), but decreases the degradation as N_{seg} is set larger (0.01 point with N_{seg} >= 20). For N_{seg} > 20, the recognition rate does not increases but the waiting time increases as discussed in Sect. 4.4.

3.4.4 Waiting time

The evaluation was done on five different pages of handwritten text captured from touch screen devices with the number of strokes for each page being 347, 398, 590, 262, or 554, respectively. We evaluate the waiting time by the semi-incremental recognition method from two points, dependency on N_{seg} and that on N_s.

As for the first point, Figure 3.9 shows the average waiting time of semi-incremental recognition with respect to N_{seg} when N_s=1 and N_s=10. The waiting time increases as N_{seg} increases regardless of N_s. As mentioned above, the recognition rate saturates for N_{seg} > 20. Therefore, hereafter we consider N_{seg} up to 20.

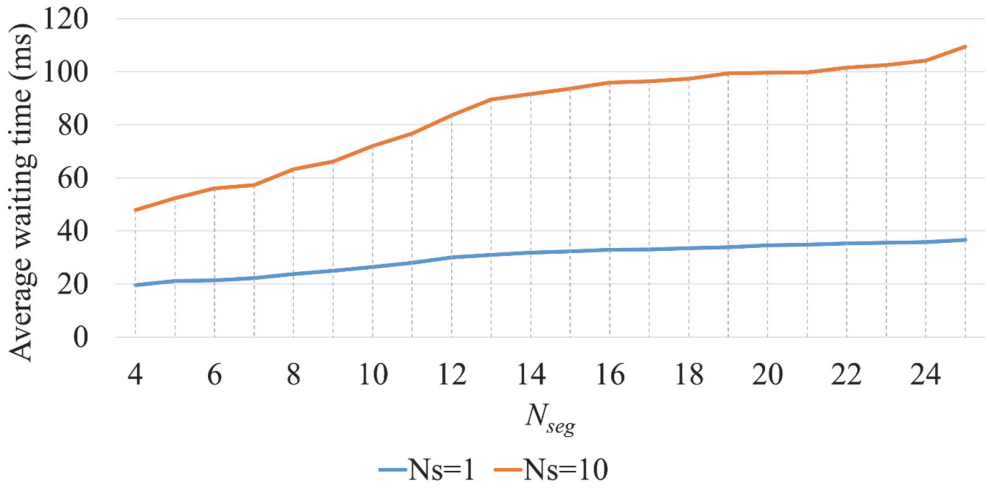


Figure 3.9 Waiting time with respect to N_{seg}

As for the second point, we measured the average waiting time with respect to N_s, while N_{seg} was fixed to 20 including the case of pure-incremental recognition (N_s = 1). We evaluated the effectiveness of reusing the src-lattice as well as applying UP fixation and PP skip.

Figure 3.10 shows the average waiting time of recognizing the five pages of Japanese text in comparison with the batch recognition method which takes 1,479ms in average for the waiting time. By reusing the src-lattice, the average waiting time is significantly reduced from roughly 150ms to 30ms. Applying UP fixation and PP skip further reduces the waiting time. The waiting time by the semi-incremental method with applying all three methods is less than 50 milliseconds, which is small enough to be unnoticeable by users. Pure-incremental recognition (N_s = 1) incurs the smallest waiting time since there is only one new stroke at each incremental processing.

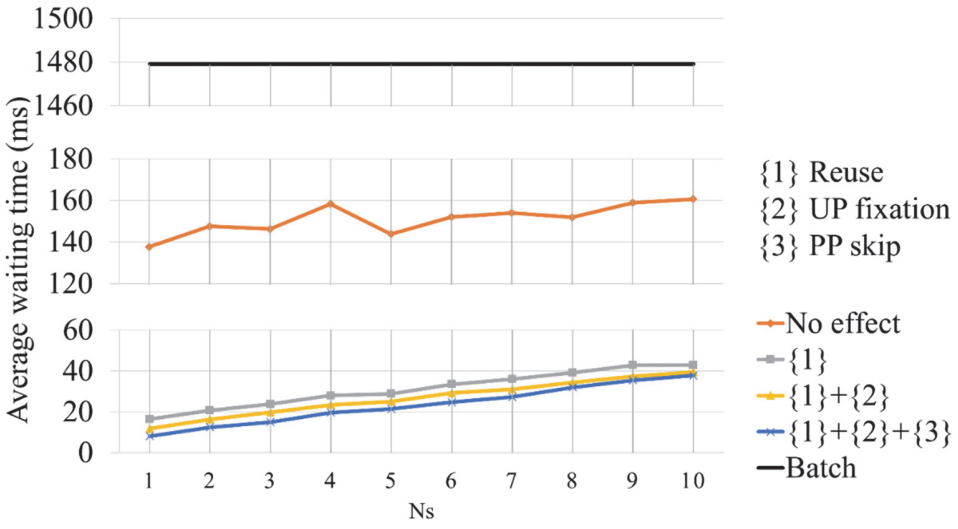


Figure 3.10 Average waiting time by semi-incremental recognition

Although the waiting time by the batch recognition method increases as text becomes longer, that by the semi-incremental recognition method is bounded, regardless of the length of the text. Figure 3.11 shows the waiting time of recognizing the first sample page (with 347 strokes) by the semi-incremental recognition method (N_s =1, 5, 10 while N_{seg} is fixed to 20) and the batch recognition method, while the number of strokes increases from 1 to 100.The case of N_s = 1 denotes pure-incremental recognition. As the number of strokes increases, the waiting time of the batch

recognition method gradually increases to 310ms, while those of semi-incremental and pure-incremental recognition are bounded to less than 150ms.

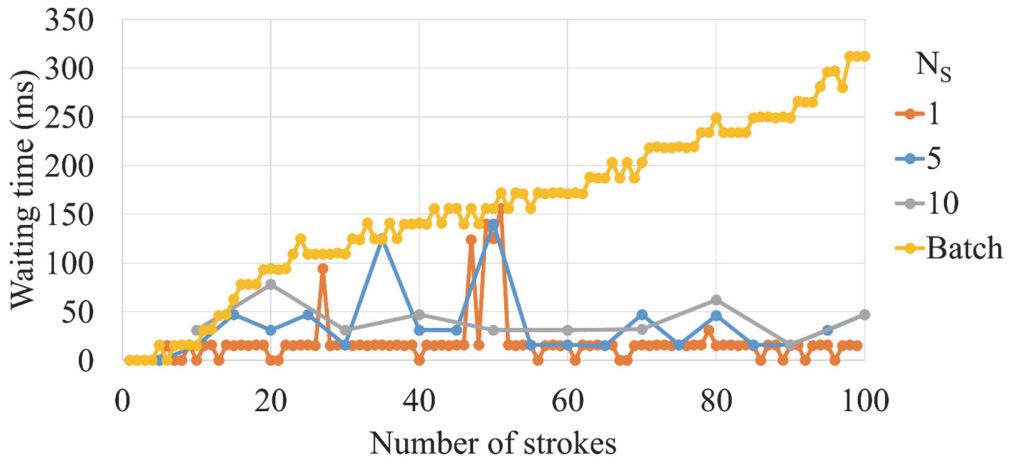


Figure 3.11 Waiting time of recognizing strokes as they increases

3.4.5 CPU time

To evaluate processing time, average CPU time per stroke is shown in Tab. 3. Compared with pure incremental recognition (N_s = 1), the semi-incremental recognition method with N_s > 2 save up to 53% of CPU time. Although the semi-incremental recognition method incurs more CPU time than the batch recognition method, it requires less CPU time as N_s increases. As for N_s >= 7, CPU time of the semi-incremental method approaches to CPU time of the batch recognition method.

Table 3.3 Processing time per stroke (ms)

Semi-incremental - N_s										Batch
1 (pure-incremental)	2	3	4	5	6	7	8	9	10	3.04
8.07	6.23	5.01	4.91	4.28	4.12	3.89	4.00	3.94	3.78	

3.4.6 Effect of resuming segmentation and UP fixation

While the batch recognition method applies segmentation for the entire handwritten text, the semi-incremental recognition method resumes segmentation from the *Seg_rp*. The first experiment was to confirm that the resumption of segmentation does not degrade the segmentation

performance if we set appropriate values for the parameters. Moreover, the performance of the semi-incremental method depends on whether we apply UP fixation or not.

Table 3.4 lists the F-measures of over-segmentation. The semi-incremental method without applying UP fixation performs the same as the batch recognition method. Applying UP fixation, however, decreases the F-measure by a small amount.

Table 3.4 F-measures of over-segmentation (%)

Semi-incremental recognition		Batch recognition
Without UP fixation	UP fixation	
99.79 ± 0.02	99.41 ± 0.05	99.80

On the other hand, UP fixation improves the detection rate of over-segmentation since more *SPs* are correctly determined. By applying UP fixation, the detection rate is improved from 2.92% to 37.49% as shown in Fig. 3.12. Moreover, applying UP fixation prevents the decrease in the detection rate when dealing with a long scope (i.e., when N_{seg} is large), although the detection rate decreases when $N_{seg} < 14$. In batch recognition, over-segmentation by using an SVM produces many *UP*s instead of *SP*s, the detection rate is as low as 3.00%. Therefore, applying UP fixation greatly improves the detection rate.

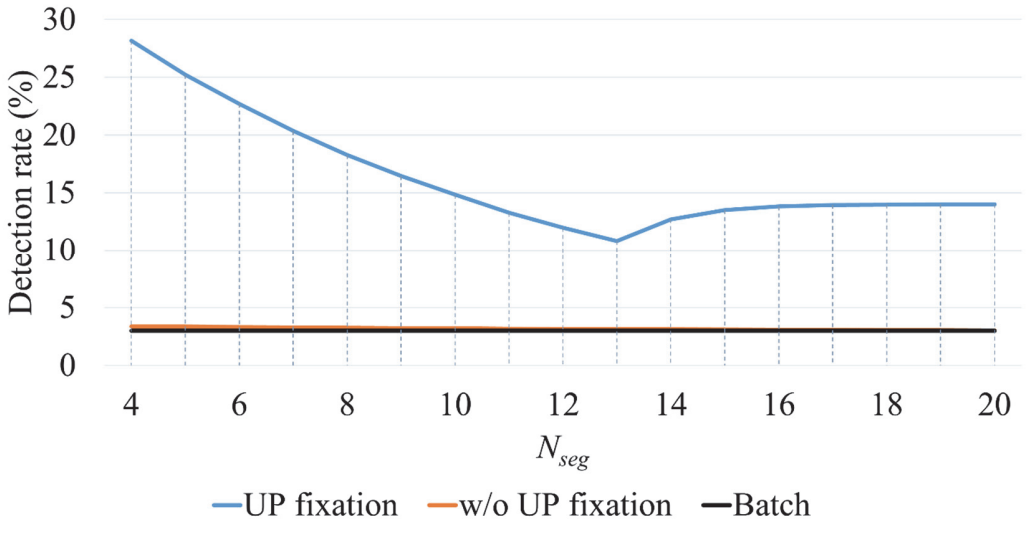


Figure 3.12 Detection rate of over-segmentation

We then evaluated the segmentation measure of final segmentation defined by Eq. (3.6). Figure 3.13 shows the segmentation measure with and without applying UP fixation with respect to N_{seg}. We set N_s from 1 to 10 and take the average segmentation measure. Compared with the batch recognition method, which yields the segmentation measure of 99.09%, the semi-incremental method with UP fixation outperforms the batch recognition method when N_{seg} >= 14. Comparison of the semi-incremental method with and without applying UP fixation shows maximum improvement of 0.13 point at N_{seg} = 19. The improvement increases as N_{seg} is set larger, since the number of correctly determined *SP*s using UP fixation increases in a longer scope. Moreover, applying UP fixation maintains a high segmentation measure in dealing with a long scope.

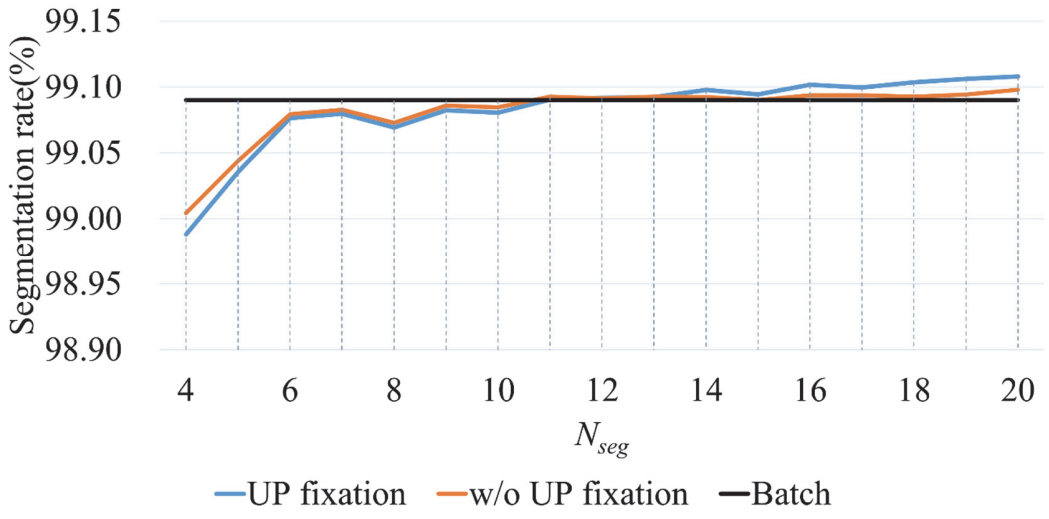


Figure 3.13 Final segmentation measure.

3.5 Conclusion

We presented a semi-incremental recognition method for on-line handwritten Japanese text. By resuming the segmentation and recognition in a local scope, the method reduces the waiting time to be small enough to be unnoticeable by users. Moreover, determining *SP* off-strokes based on recognition result shortens block lengths and bounds the waiting time. Skipping the recognition of partial patterns and reusing recognized character patterns in the src-lattice are also shown to be effective in reducing the waiting time.

The control variables N_s and N_{seg} should be set according to the environments. As far as they are set as shown in the experiments, the semi-incremental recognition method is clearly superior to the batch recognition method in the waiting time while maintaining the recognition rate. It also excels pure incremental recognition in the character recognition rate and the total CPU time.

Chapter 4

Semi-incremental online handwriting recognition method for English text

This chapter presents a semi-incremental online handwriting recognition method for English text. It extends the local context for recognition to a range of recent strokes called ‘scope’, triggers recognition at several recent strokes, updates the candidate word lattice and searches over the lattice for the best result incrementally. To reduce the waiting time, our method reuses previously recognized candidate word patterns, if they exist in the previous stage of the lattice. It also fixes undecided segmentation points if they are stable between word patterns. Moreover, it skips recognition of partial candidate word patterns. The semi-incremental method includes the case of triggering recognition at every new stroke with the above-mentioned techniques. We evaluate our method using the IAM-OnDB database and show that it generates results without noticeable delay while keeping high recognition rate. It also decreases CPU time.

The chapter is organized as follows. Section 4.1 is the Introduction. Section 4.2 describes the method for English with detailed descriptions of generalization and modification. Section 4.4 presents our experiments and the results. The conclusions are presented in Sect. 4.5.

4.1 Introduction

As previously introduced, the development of pen-based and touch-based devices makes research of on-line handwritten text recognition getting more attention recently. Along with the motivation of increasing recognition accuracy, the requirement of recognition speed and CPU burden in hand-held devices also induces the research for reduce waiting time and CPU time for recognition process.

In order to satisfy these requirements, we proposed a semi-incremental recognition method and applied it for Japanese text recognition (C. T. Nguyen, Zhu, and Nakagawa 2013). The batch-recognition method, which recognizes online handwritten text after it is written, produces high recognition rate owing to the full context, but incurs large waiting time. Pure incremental

recognition, triggering recognition at every stroke, where a stroke is a sequence of pen-tip coordinates from pen-down to pen-up, decreases the waiting time but increases the CPU time and lowers the recognition rate (Wang, Liu, and Zhou 2012; Tanaka 2002). The semi-incremental method triggers recognition at every few recent strokes, including the case of every new stroke so that it produces recognition output without noticeable delay as compared with batch recognition. It saves CPU time in comparison with the pure incremental recognition due to processing a sequence of few strokes rather than every new stroke. Moreover, we propose a local but larger context to maintain high recognition rate, and introduce three techniques to save CPU time and decrease waiting time even for pure incremental recognition.

After preliminary reports for Japanese (C. T. Nguyen, Zhu, and Nakagawa 2013) and for English (C. T. Nguyen, Zhu, and Nakagawa 2014), we revised segmentation and introduced time synchronous update of the candidate lattice for handwritten text, and skipping recognition of partial patterns (C.-T. Nguyen, Zhu, and Nakagawa 2016). This chapter focuses on a generalization of this method to cope with online handwritten English text, and presents a comprehensive evaluation using the IAM-OnDB database.

4.2 Recognition system overview

The recognition system for English first introduced in (C. T. Nguyen, Zhu, and Nakagawa 2014), we follow the approach of explicit segmentation with soft decision method as presented in Section 2.2.1. The flow of recognition system is shows in Fig. 4.1. From an input sequences of strokes, segmentation step produces segmentation candidates for segmenting words. Next, the word segmentation candidates are used for creating word candidate lattice, which includes word recognition as well as context information in each path of words. Finally, those paths are evaluated to find the best path, which represents the recognition result.

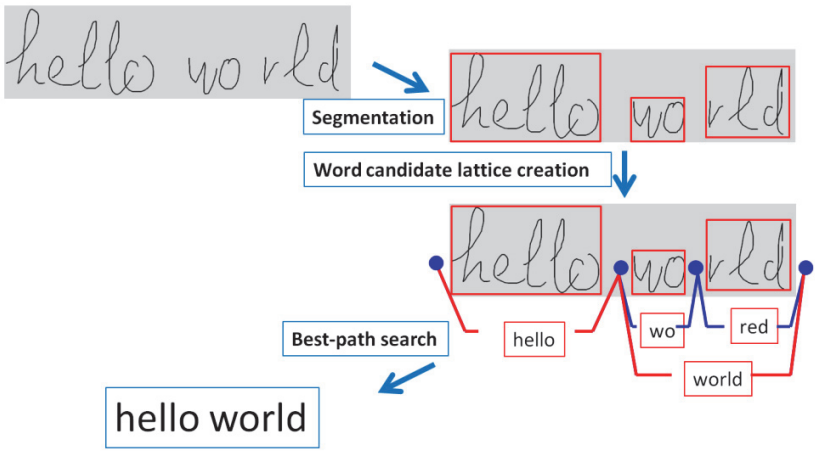


Figure 4.1 English text recognition system

4.2.1 Segmentation

For line segmentation, we apply a linear regression method to estimate the latest text line from the centroids of its strokes, and judge incrementally whether the latest stroke belongs to the latest text line or creates a new line. To make the text line estimation more precise, we use the weighted linear regression method, with the weight for each stroke centroid being the number of points in that stroke.

Here is the detail of the line segmentation method. First, we get the centroids of previous strokes, then use weighted linear regression to approximate the closest line of all those centroids. Next to determine if a stroke belongs to the current line or not, we calculate distance from this stroke and its neighbor strokes to the current line, a threshold is used.

The formulae of weighted linear regression method is shown in the following:

$$Y = \alpha + \beta X \tag{4.1}$$

Then, to determine β and α:

$$\beta = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}$$
$$\alpha = \overline{y} - \beta\overline{x} \tag{4.2}$$

With the following notations:

$$\bar{x} = \sum_i w_i x_i\,, \qquad \bar{y} = \sum_i w_i y_i\,, \qquad \overline{x^2} = \sum_i w_i {x_i}^2\,, \qquad \overline{xy} = \sum_i w_i x_i y_i$$

We apply segmentation in both forward and backward directions for the latest text line to make line segmentation robust, even when it is composed of a small number of strokes. If it is segmented by either forward or backward segmentation, it is segmented.

For word segmentation, we employ Bidirectional Long Short-Term Memory (BLSTM) network (Graves and Schmidhuber 2005) for word segmentation. BLSTM consists of recurrent neural networks, and works by accessing long-range context in both forward and backward directions to obtain segmentation probabilities. Based on off-stroke classification score obtained from BLSTM networks we determine an off-stroke as *SP*, *NSP* or *UP*.

We use a set of geometric features including: (1) gap between the preceding and succeeding strokes of the current off-stroke, (2) average stroke length on horizontal direction of the whole pattern, of sub-pattern containing adjacent strokes, (3) overlapping of distance between two adjacent strokes of the current off-stroke, (4) minimum point distance between the two adjacent strokes of the current off-stroke, (5) the direction of the centroids of the two adjacent strokes of the current off-stroke, (6) the ratio of the two bounding box of the preceding stroke and succeeding stroke of the current off-stroke.

We use a new type of feature: average stroke length on horizontal direction (ASL) feature (Fig. 4.2). This feature on an on-line pattern could be considered as the vertical histogram on an off-line pattern. In factor (2) we use the feature of both the whole pattern and the sub-pattern containing two adjacent strokes of the current off-stroke. For a sub-pattern containing a segmentation off-stroke, the ASL of it would be small due to the gap between the two adjacent strokes, therefore, ASL is a good feature to determine segmentation off-stroke. On the other hand, the ASL of the whole pattern provides a perspective of handwriting style, make the classifier robust with various handwriting styles.

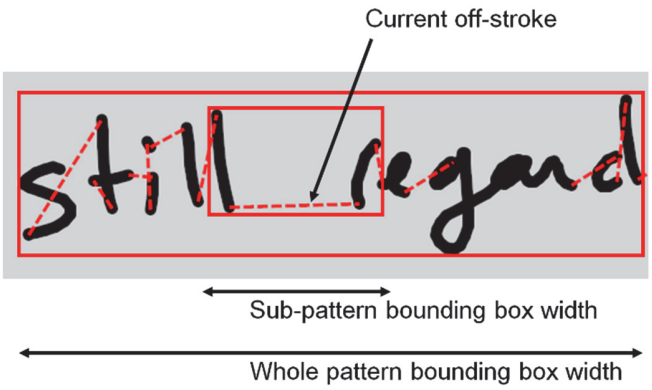


Figure 4.2 ASL feature.

4.2.2 English word recognition

Semi-incremental recognition presented here is not restricted to a particular recognizer. Any recognizer can be employed as long as it can recognize English words with confidence scores. Here, we employ an English word recognizer that combines Markov Random Fields (MRF) and Modified Quadratic Discriminant Functions (MQDF) to incorporate both online and offline information for character recognition (Zhu et al. 2013).

4.3 Semi-incremental recognition for English

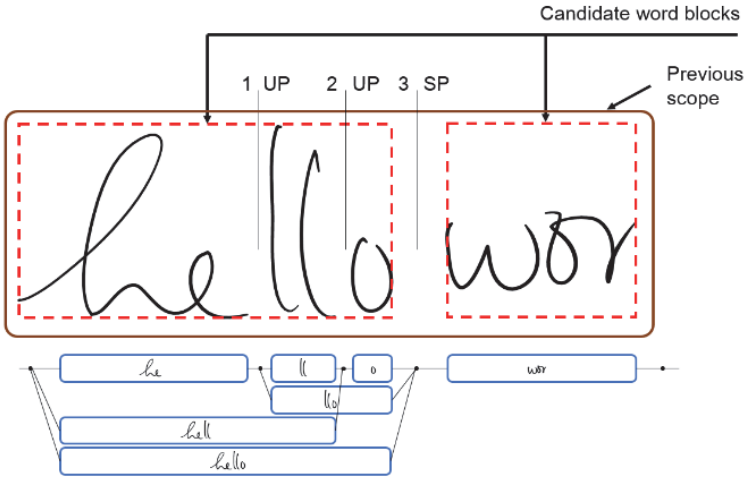
We basically follow the semi-incremental recognition method summarized in Chapter 3 to recognize online handwritten English text. Here, English words correspond to Japanese characters. In this section, we describe adaptation and modification of the method to English.

4.3.1 Determination of scope

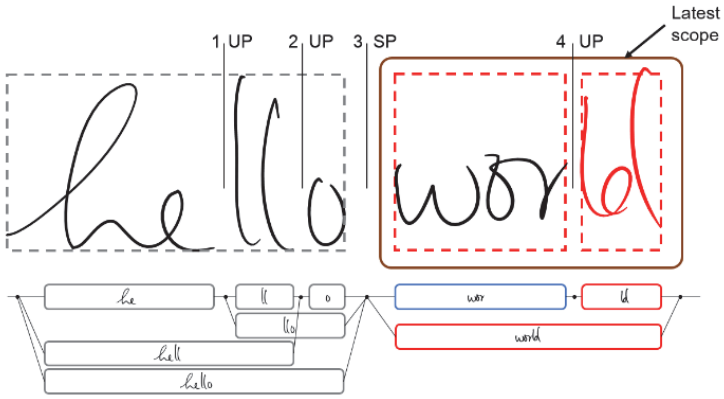
For English text recognition, the scope is determined in the same way as in Sect. 3.3.3, except that a candidate word block (a sequence of strokes between two consecutive *SP*s) is used instead of a candidate character block.

4.3.2 Time synchronous creation and update of src-lattice

In the current scope, we create the src-lattice incrementally in the time-synchronous order of the candidate segmentation points. For each candidate segmentation point, we build all of the candidate word patterns that end at that candidate segmentation point.



(a) Previous scope.



(b) Latest scope.

Figure 4.3 Reuse of candidate word patterns.

We reuse previously recognized candidate word patterns if they exist in the previous scope. Figure 4.3(a) shows candidate word patterns (bounded by blue rounded rectangles) in the src-lattice for the previous scope. When new strokes are added (shown in red) as in Fig. 4.3(b), we

update the src-lattice from the beginning of the latest scope, which triggers to build three candidate word patterns. Among them, two candidate word patterns bounded by red rounded rectangles have to be newly built. One candidate word pattern bounded by blue rounded rectangle are reused from the previous scope.

4.3.3 Time synchronous best-path search and recognition

The recognition result is produced incrementally from the latest src-lattice by the best-path search. For each time step, we apply the Viterbi search incrementally to find the best path reaching to that time step. Paths are evaluated by combining scores of word recognition, segmentation and linguistic context. For evaluating a path through a sequence of m primitive segments $S = s_1, s_2, ..., s_m$ of an input sequence X, forming a sequence of n candidate character/word patterns $Z = z_1, z_2, ..., z_n$ which is assigned as $C = c_1, c_2, ..., c_n$, we have the posterior probability as follows:

$$\begin{aligned} f(X,S,G,C) = & \sum_{i=1}^{n} \left\{ \begin{array}{l} \log P(r_i \mid c_i) \\ +\lambda_1 \log P\left(c_i \mid c_{i-2} c_{i-1}\right) \end{array} \right\} \\ & +\lambda_2 \left\{ \sum_{j=\overline{1,m-1};T(d_j)=B} \log P_{sp}(d_j) + \sum_{j=\overline{1,m-1};T(d_j)=W} \log P_{nsp}(d_j) \right\} \end{aligned} \tag{4.3}$$

with $P_{sp}(d_j)$ and $P_{nsp}(d_j)$ are the classification probabilities of an off-stroke being classified as *SP* and *NSP*, respectively. The labeling function T output the type of off-stroke (*B*: between, *W*: within) for a candidate segmentation point. $P\left(c_i \mid c_{i-2} c_{i-1}\right)$ is the tri-gram language probability. The likelihood probability $P\left(r_i \mid c_i\right)$ of a class c_i with respect to a candidate word pattern r_i is calculated using the word recognizer presented in Sect. 4.4.2. The parameters λ_1 and λ_2 are optimized by the Minimum Classification Error (MCE) algorithm (McDermott et al. 2007).

4.3.4 UP fixation and PP skip

We employ UP fixation and PP skip for online handwritten English text in the same way as for Japanese text, except that English words are employed rather than Japanese characters. In other

words, *UPs* between recognized characters before the latest N_{seg_det} words in the result of text recognition are fixed as *SPs* in UP fixation, and candidate word patterns containing the latest primitive segment are skipped without recognition until a new primitive segment is detected, or the recognition is requested in PP skip.

4.4 Experiments

This section presents experiments and evaluation of our proposed method for online handwritten English text.

4.4.1 Setup for experiments

To evaluate the recognition method, we used the IAM online database (IAM-OnDB) (Marti and Bunke 2003), which consists of pen trajectories collected from 221 different writers using an electronic whiteboard. We followed the handwritten text recognition task IAM-OnDB-t2, in which the database is divided into a training set, two validation sets, and a test set containing 5364, 1438, 1518, and 3859 written lines, respectively.

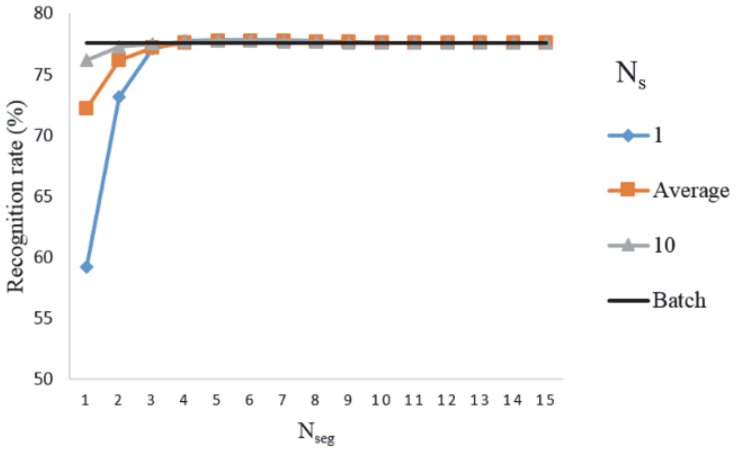
For our purpose, however, IAM-OnDB provides the ground truth only for the sentence level, and not for the word level. Therefore, we prepared the word-segmentation ground truth for IAM-OnDB by making a tool to detect segmentation candidates, and employing human inspection to verify the candidates (C. T. Nguyen, Zhu, and Nakagawa 2014).

We trained a BLSTM classifier consisting of 20 LSTM blocks with one cell in each block for classifying the segmentation of off-strokes. We also trained the word recognizer combining MRF and MQDF by IAM-OnDB with word-level ground truth. For language modeling, we employed a trigram table extracted from the Lancaster-Oslo-Bergen (LOB) text corpus (Johansson 1978).

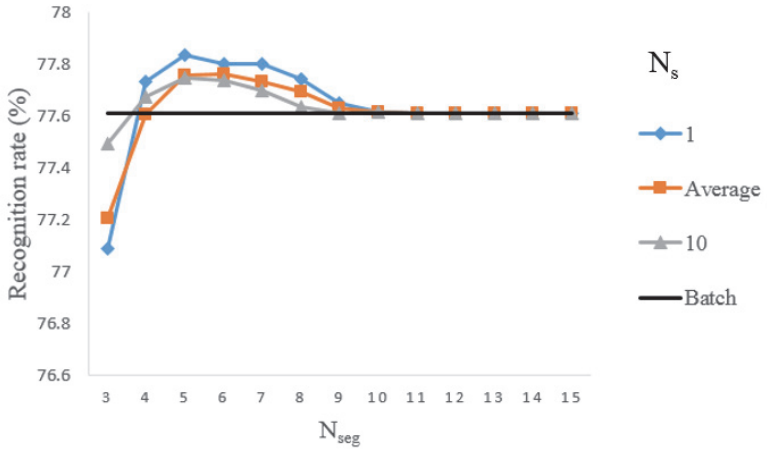
We compared the semi-incremental recognition system and the batch recognition system in (C. T. Nguyen, Zhu, and Nakagawa 2014), where BLSTM was used for over-segmentation instead of SVM. We ran all the experiments on an Intel® Core™ i7-4770 (3.4Ghz) with 8GB memory.

4.4.2 Word recognition rate

Figure 4.4 shows the word recognition rate (i.e. the percentage of correctly recognized words over all the handwritten words) by the semi-incremental method, including the case of pure-incremental recognition (N_s = 1) in comparison with the batch recognition method. In the experiments we varied N_s (from 1 to 10) and N_{seg} (from 1 to 15) to evaluate the effect of resuming segmentation to the word recognition rate. Figure 4.4(a) shows the recognition rate for 1 <= N_{seg} <= 15 while Fig. 4.4(b) shows the rate for 3 <= N_{seg} <= 15 with larger resolution.



(a) 1 <= N_{seg} <= 15.



(b) 3 <= N_{seg} <= 15.

Figure 4.4 Recognition rate with respect to N_{seg} .

As N_{seg} increases up to 5, the recognition rate of the semi-incremental recognition method rises without dependence on N_s (the maximum and minimum recognition rates are nearly the same), since the segmentation and recognition are resumed from a more stable *Seg_rp*. From $N_{seg} > 6$, however, the recognition rate drops and converges to that of batch recognition. The maximum recognition rate is 77.83% with $N_{seg} = 5$, which is a little better than 77.61% by the batch recognition method.

The pure incremental recognition ($N_s = 1$) performs slightly better than the semi-incremental recognition when $4 <= N_{seg} <= 8$. Note that this pure incremental recognition is different from the ones proposed in (Tanaka, Akiyama, and Ishigaki 2002; Wang, Liu, and Zhou 2012), since the scope is introduced and segmentation is resumed from several strokes behind the latest stroke with $N_{seg} > 1$ even if $N_s = 1$.

4.4.3 Waiting time

In this experiment, we evaluated the waiting time from two points, dependency on N_{seg} and on N_s. We measured it on thirteen sample text lines from IAM-OnDB with each containing 36 strokes on average.

As for the dependency on N_{seg}, Fig. 4.5 shows the average waiting time of semi-incremental recognition including the case of pure incremental recognition ($N_s = 1$) with respect to N_{seg} while $1 <= N_s <= 10$. The waiting time is slightly extended as N_{seg} increases regardless of value of N_s.

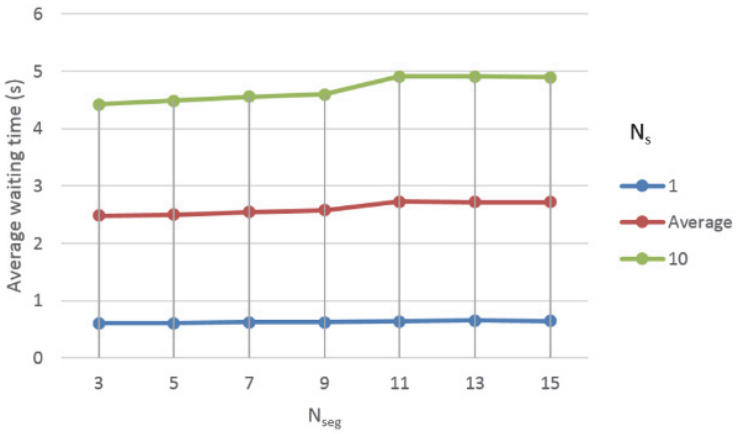


Figure 4.5 Waiting time with respect to N_{seg}.

As for the second point, Fig. 4.6 shows the average waiting time with respect to N_s, while N_{seg} is fixed to 5, which produces the best recognition rate. It is shown for the baseline without all of RP reuse, UP fixation and PP-skip, and for the system that adds them gradually. Semi-incremental recognition with RC reuse takes the average waiting time from 1.20s to 5.01s when N_s increases from 1 to 10, which is a significant reduction of the waiting time as compared with batch recognition taking 14.80s on average. UP fixation is more effective when N_s is large ($N_s > 7$). On the other hand, PP skip is somewhat more effective when N_s is smaller ($N_s < 5$). By applying all the three newly devised techniques, the waiting time of the semi-incremental recognition method is around 1s when $N_s < 3$, which is small enough to not distract the users. Pure incremental recognition ($N_s = 1$) incurs the smallest waiting time: applying all the three techniques shorten it from 1.98s to 0.58s.

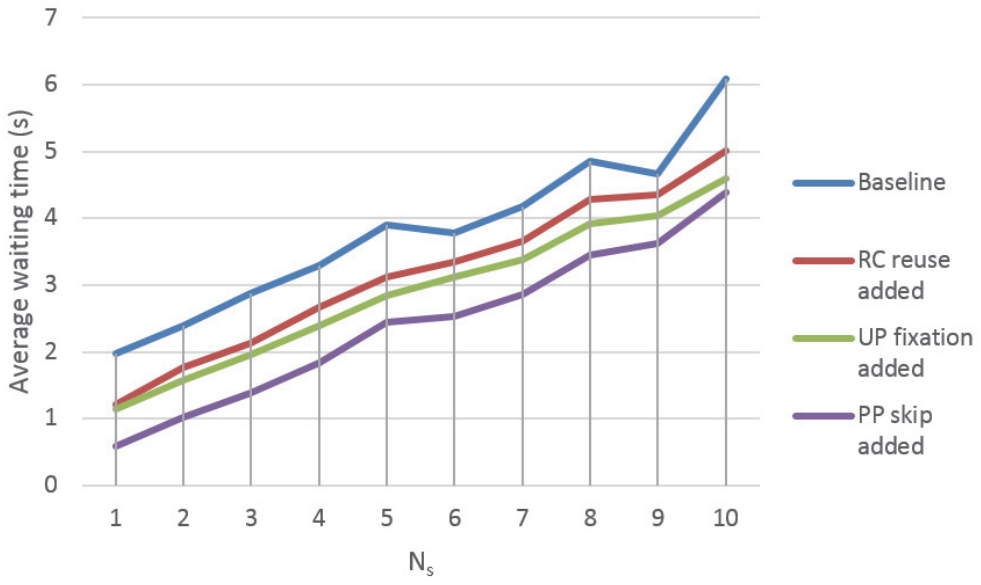


Figure 4.6 Waiting time with respect to N_s.

Figure 4.7 shows the waiting time of recognizing a sample text line (with 43 strokes) by semi-incremental method (N_s = 1, 5, 10, with N_{seg} fixed to 5) when the number of strokes increases from 1 to 43. Although the waiting time of batch recognition gradually increases to 14s as the text becomes longer, that of semi-incremental recognition is generally bounded within 2s regardless of the length of the text, but sometimes goes up to 8s. The anomalous cases occur for long handwritten words.

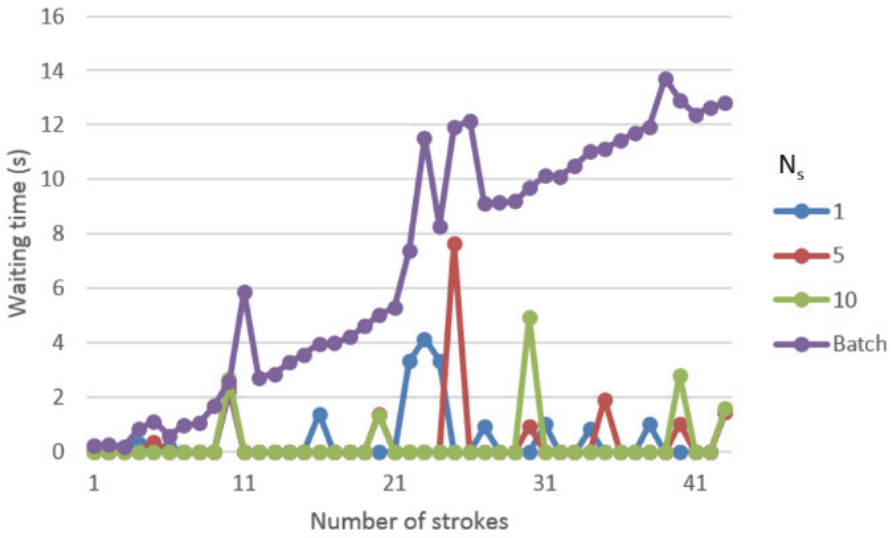


Figure 4.7 Waiting time of recognizing a sample text line

4.4.4 CPU time

Average CPU time per stroke is shown in Tab. 4.1. It is again shown for the baseline without the three techniques and for the system that adds them gradually. When all the three techniques are incorporated, semi-incremental recognition with N_s >= 2 saves up to 31.63% of the CPU time, i.e. from 0.59s of pure incremental recognition (N_s = 1) to 0.40s when N_s = 9. Pure incremental method triggers the recognition process at every input stroke received, therefore requires more CPU time. RP reuse reduces the CPU time from 1.98s to 1.20s, UP fixation reduces it to 1.14s and PP skip further reduces it to 0.59s with the total effect of 70.2%. Although the semi-incremental recognition method incurs more CPU time than the batch recognition method, it requires less CPU time as N_s increases. As for N_s >= 6, the CPU time of the semi-incremental method approaches to that of the batch recognition method.

Table 4.1 CPU time evaluation (sec)

Method / CPU time	Semi-incremental - Ns									Batch
	1 (pure-incremental)	2	3	4	5	6	7	8	9	10	
baseline	1.98	1.20	0.96	0.82	0.78	0.63	0.60	0.61	0.52	0.61	0.40
RP reuse added	1.20	0.88	0.71	0.67	0.62	0.56	0.52	0.53	0.48	0.50	
UP fixation added	1.14	0.79	0.66	0.60	0.57	0.52	0.48	0.49	0.45	0.46	
PP skip added	0.59	0.51	0.46	0.46	0.49	0.42	0.41	0.43	0.40	0.44	

4.5 Conclusion

We applied the semi-incremental online handwriting recognition method for English text and evaluated it using the IAM-OnDB database. The method shortened waiting time to unnoticeable level, decreased CPU time to almost half without degrading recognition rate. The scope and our three newly-devised techniques decrease waiting time and CPU time not only for the semi-incremental recognition but also for the pure-incremental recognition.

There are still problems when long English words are encountered. To increase the recognition rate and shorten the waiting time for these cases is left for future research.

Chapter 5

Improving Segmentation of Online English Handwritten Text Using Recurrent Neural Networks

Segmentation of online handwritten text recognition is better to employ the dependency on context of strokes written before and after it. This paper shows an application of Bidirectional Long Short-term Memory recurrent neural networks for segmentation of on-line handwritten English text. The networks allow incorporating long-range context from both forward and backward directions to improve the confident of segmentation over uncertainty. We show that applying the method in the semi-incremental recognition of online handwritten English text reduces up to 62% of waiting time, 50% of processing time. Moreover, recognition rate of the system also improves remarkably by 3 points from 71.7%.

The chapter is organized as follows: Section 5.1 is Introduction, Section 5.2 presents the segmentation methods for online handwritten text, Section 5.3 presents experimental results, Section 5.4 gives a discussion and finally Section 5.5 draws our conclusion.

5.1 Introduction

In online handwritten text recognition, explicit segmentation approaches face the difficulty of word segmentation and character segmentation due to the ambiguity in segmentation. Moreover, in continuous handwriting, characters tend to be written more cursively.

To deal with the problem, applying context for segmentation is crucial. The typical approach is using over-segmentation in combination with recognition results and linguistic context (Zhu et al. 2010). Based on geometric features, all the potential segmentation positions are determined to build up hypothetical segmentation paths. Then, recognition results and linguistic context is combined to evaluate and find the best path.

The SVM method, which has been widely applied to numerous classification tasks achieves good performance on segmentation of on-line handwritten text (Zhu and Nakagawa 2008). The segmentation task, however, can be further improved by incorporating context from both the

forward and backward directions. An improved version of bidirectional recurrent neural network: Bidirectional Long Short-term Memory BLSTM (Graves and Schmidhuber 2005) allows the network to access long-range context. BLSTM shows its effective in many sequence classification tasks.

In this work, we apply BLSTM for improving segmentation and evaluate its effect on the semi-incremental English recognition method.

5.2 Segmentation of online handwritten text

To recognize online handwritten text, there are two main streams: the segmentation free method and the dissection method. In this work, we focus on the dissection method since it is better for Chinese and Japanese text recognition (Wang, Liu, and Zhou 2012; Zhu et al. 2010) and it could produce better results even for western handwriting recognition for which the segmentation free method has been dominant.

5.2.1 Features for segmentation

For each off-stroke of the input stroke sequence, we extract the set of local and global features originally described in (C. T. Nguyen, Zhu, and Nakagawa 2014) and extend it into set of nine geometrical features. The features are listed in Tab. 5.1, the term are defined in Tab. 5.2. The features is based on geometric properties of the stroke pair preceding and succeeding the off-stroke (local geometrical features) and the geometric properties of the whole strokes correlating with current off-stroke (global geometrical features). Applying the both the global and local geometric features makes the segmentation more robust with various handwriting styles.

5.2.2 Segmentation by a SVM classifier

For word over-segmentation of a text line, the work in (Zhu and Nakagawa 2008) uses a SVM classifier to classify each off-stroke into two classes: segmentation point (*SP*) or non-segmentation point (*NSP*). A *SP* off-stroke separates two words while a *NSP* off-stroke indicates the off-stroke is within a word. Off-strokes with low confidence are classified as undecided point (*UP*).

Table 5.1 Features for English word segmentation

Global features	
F1	Distance between B_{s_all} and B_{p_all} in x-axis
F2	Average stroke length of P_{all} in x-axis
Local features	
F3	Average stroke length of P_{sub} in x-axis
F4	Overlap length between B_p and B_s in x-axis
F5	Overlap length between B_p and B_s in y-axis
F6	Minimum point distance between S_s and S_p
F7	Angle between the vector from the centroids of B_s and B_p and x-axis
F8	Ratio between B_s width and B_p width
F9	Ratio between B_s height and B_p height

Table 5.2 Terms of Features representation.

S_p	Immediate preceding stroke
S_s	Immediate succeeding stroke
B_p	Bounding box of S_p
B_s	Bounding box of S_s
B_{p_all}	Bounding box of all the preceding strokes in the latest text line
B_{s_all}	Bounding box of all the succeeding strokes in the latest text line
P_{all}	All the strokes in the latest text line
P_{sub}	Sub-pattern of S_p and S_s

In training the SVM, however, due to unbalance between the numbers of positive and negative training patterns (i.e. the number of *SP* and that of *NSP*), we need to adjust the cost of false positives and false negatives (Morik, Brockhausen, and Joachims 1999). The higher cost of false positives is set, the higher precision of determining *SP* is achieved. The same logic applies to false negatives and precision of determining *NSP*. We use a combination of two SVMs: one with high precision for determining *SP*, the other with high precision for determining *NSP*.

5.2.3 Segmentation by a BLSTM classifier

One of the key benefits of RNNs is their ability to use previous context. For standard RNN architectures, however, the range of context that can be accessed in practice is limited due to the vanishing gradient problem (Hochreiter et al. 2001).

Long Short-Term Memory (LSTM (Hochreiter and Schmidhuber 1997)) is an RNN architecture designed to address the vanishing gradient problem. A LSTM layer consists of multiple recurrently connected memory blocks. Each block contains a set of internal units, known

as cells, whose activation are controlled by three multiplicative 'gate' units. The effect of the gates is to allow the cells to store and access information over long periods of time.

For many tasks, it is useful to have access to future as well past context. Bidirectional LSTM (BLSTM) allows this (Graves and Schmidhuber 2005) by using two separate hidden layers to present input in forward and backward directions, both of which are connected to the same output layer to provide access to long-range bidirectional context.

We use BLSTM to employ the context of strokes written before and after an off-stroke for segmentation of that off-stroke. The training of BLSTM does not suffer the problem of different in number of class patterns. Therefore, we use BLSTM with two thresholds to make over-segmentation.

For over segmentation, we need to find all potential segmentation points of off-strokes (which could be then determined as segmentation or non-segmentation points), the remaining are non-segmentation points. Thus, we set a threshold TH1 to determine an off-stroke as a potential segmentation point if the score is above than TH1 and as a non-segmentation point if the score is below TH1. Likewise, we set another threshold TH2 to determine an off-stroke as a potential non-segmentation point or a segmentation point. The off-strokes whose score fall between TH1 and TH2 are classified as UP. Fig. 5.1 illustrates this method.

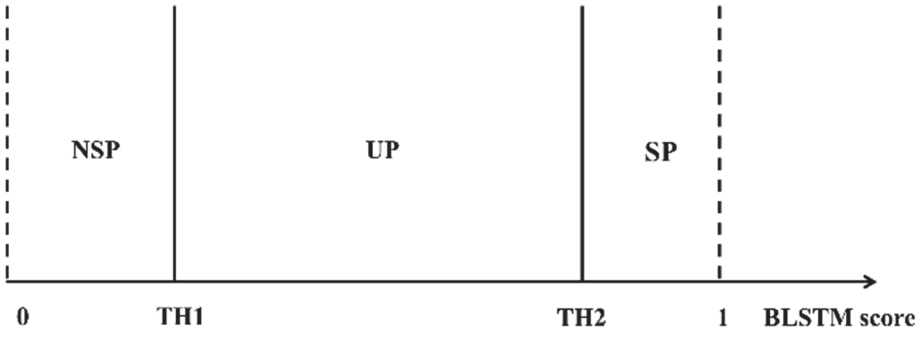


Figure 5.1 Over-segmentation using BLSTM.

5.3 Experiments

5.3.1 Experiment setup

We employ the IAM online database (IAM-OnDB) (Marti and Bunke 2003) which consists of pen trajectories collected from 221 different writers using an electronic whiteboard. We follow the handwritten text recognition task: IAM-OnDB-t1 in which the database is divided into a training set, two validation sets, and a test set containing 5,364, 1,438, 1,518 and 3,859 written lines, respectively. We use a trigram table extract from the LOB text corpus for language modeling.

For segmentation, we train both the SVM classifier and BLSTM classifier on segmented words of IAM-OnDB. The SVM classifiers use the RBF kernel with cost factor of 0.1 for high precision of *SP* determination and 7.5 for high precision of *NSP* determination. The BLSTM classifier uses a bi-directional layer of 20 LSTM blocks with one cell in each block. After training the BLSTM classifier, based on the distribution of output scores, we set TH1 = 0.1 and TH2 = 0.9.

We compare the performance of two semi-incremental recognition systems: the first system uses the SVM classifiers for segmentation (SVM system) and the second uses BLSTM classifier for segmentation (BLSTM system).

5.3.2 Over-segmentation

The BLSTM system outperforms the SVM system for both recall and detection rates. Recall rate has improved 1.5 point, while detection has significant improved from 48% to 86% as shown in Table 5.3.

Table 5.3 Over-segmentation Results

Measures	SVM	LSTM
Recall	96.91	98.57
Precision	99.25	99.06
F-measure	98.07	98.81
Detection	48.34	86.55

5.3.3 Recognition rate

The recognition rates of the SVM system and the BLSTM system with changing N_{seg} parameter are shown in Fig. 5.2. For each N_{seg} we run with the recognition trigger parameter (N_s) from 1 to 10 and take the average. The BLSTM system improves recognition rate by about 3 point. With high detection rate, the BLSTM system reduces a large number of *UPs* as compared with SVM. Therefore, the system reduces the ambiguity in best path search and improves recognition rate.

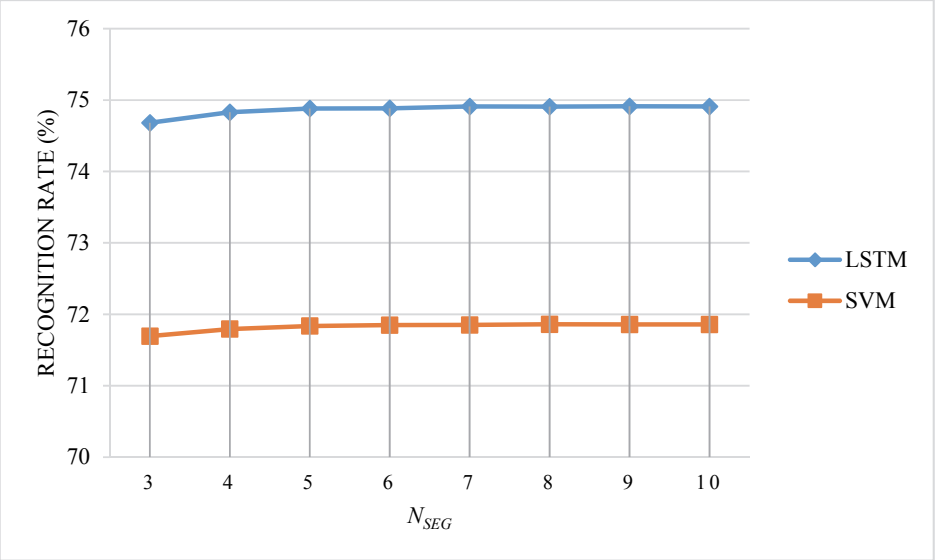


Figure 5.2 Recognition rate of the two systems.

5.3.4 Waiting time

We measure the average waiting time of the two systems with changing N_s. The BLSTM system has reduction rate of average waiting time from 36.65% to 62.43% over the SVM system as shown in Fig. 5.3.

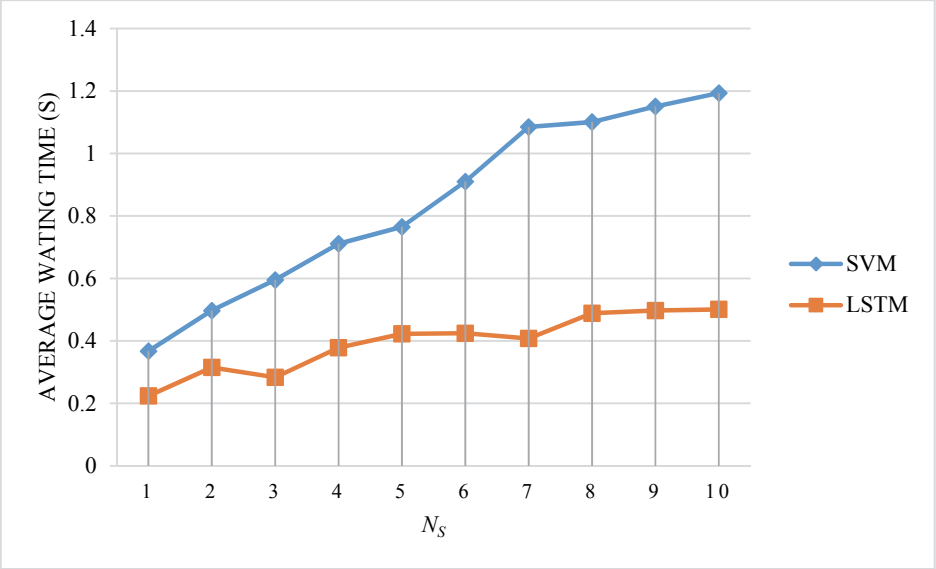


Figure 5.3 Average waiting time of the two systems.

5.3.5 CPU time

We also compare both of the systems in CPU time per stroke. Fig. 5.4 shows the results of the BLSTM and SVM systems with changing of N_s. The BLSTM system also reduces about 50% of CPU time as compared with the SVM system.

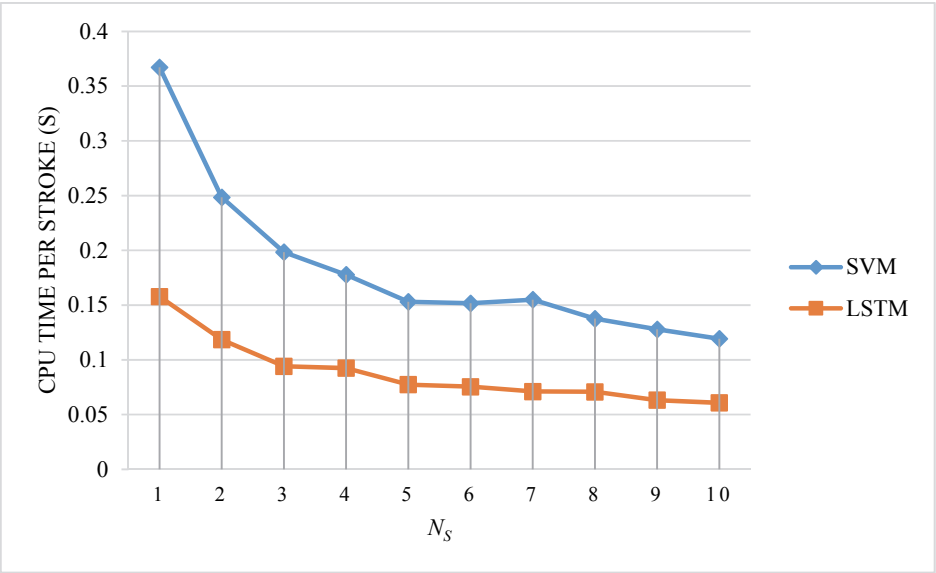


Figure 5.4 CPU time of the two systems.

5.4 Discussion

Detection rate gives the ratio of segmentation points over all potential segmentation points. For the two systems, as the same recall rate, higher detection rate reduces the number of undecided points. Since each undecided point doubles the number of candidate word patterns which need to be recognized, high detection rate reduces processing time and waiting time. Each undecided point also doubles the number of search paths. Therefore, higher detection rate reduces the number of search paths, lowers ambiguity, and improves recognition rate.

5.5 Conclusion

We proposed a system using BLSTM recurrent neural network for segmentation of on-line handwritten English text. By large improvement in the detection rate of over-segmentation, BLSTM reduces the number of undecided points each of which doubles the number of candidate character patterns. The reduction of candidate character patterns is vital since character recognition is applied for each candidate. The BLSTM system reduces up to 62.34% of waiting time and around 50% of CPU time as compared with the SVM system. Moreover, reducing undecided points also reduces the number of search paths, and lowers ambiguity of recognition so that BLSTM improves recognition rate of the system by 3 points from 71.7%

Chapter 6

Decoding of Handwritten Text Using Recurrent Neural Networks

This chapter presents a Finite State Machine (FSM) to reduce user's waiting time to get the recognition result after finishing writing in recognition of online handwritten English text. The lexicon is modeled by a FSM, and then determination and minimization are applied to reduce the number of states. The reduction of states in the FSM shortens the waiting time without degrading the recognition accuracy. Moreover, by merging incoming paths to each state, the recognition rate is improved. The N-best states decoding method also reduces the waiting time significantly with small degradation in recognition accuracy. Experiments on IAM-OnDB and IBM_UB_1 show the effectiveness of the method in both reducing waiting and improving recognition accuracy.

6.1 Introduction

The development of pen-based or touch-based devices such as Tablet PCs, smart-phones, digital pens and so on, makes the problem of online handwriting recognition getting more attention. By achieving high recognition accuracy with small waiting time and providing natural interface, online handwritten text recognition has been a practical input method for these devices without keyboard (Liu, Jaeger, and Nakagawa 2004; Plamondon and Srihari 2000; Graves et al. 2009).

Compared to isolated character or word recognition, handwritten text recognition faces the difficulty of word segmentation and character segmentation due to the ambiguity in segmentation. The problem is more challenging while dealing with recognition of unconstrained handwriting, in which characters tend to be written cursively.

Segmentation-based approach (e.g. in (Zhu et al. 2010; C. T. Nguyen, Zhu, and Nakagawa 2014)) first separates handwritten text into smaller units (characters or words) then recognize them using a character or word recognizer. This approach not only faces the problem of segmentation, but the necessity of pre-segmentation of handwriting text for training.

Segmentation-free approach avoids the above problem and necessity. Graves et al. (Graves et al. 2009) introduce a connectionist temporal classification (CTC) layer for recurrent neural networks (RNN), allowing it learns end-to-end sequence mapping from handwriting input to text transcript. Not only benefitting from unnecessary pre-segmentation, end-to-end training and recognition show its effectiveness in recognition accuracy by avoiding errors caused by explicit segmentation.

Although segmentation-free method by applying end-to-end training and recognition may realize high recognition accuracy, the decoding (i.e. the task of finding the best label sequence from an input sequence) incurs large waiting time in large vocabulary recognition tasks. In (Graves et al. 2009), a decoding method based on token passing algorithm (Young, Russell, and Thornton 1989) is applied to constrain the output label sequence to a vocabulary. The token passing algorithm processes through the list of internal states in each word at each time step of an input sequence. As for the large number of words in vocabulary, the number of internal states is enlarged and therefore waiting time is extended.

In this paper, we focus on reducing waiting time of decoding for handwritten text recognition. We introduce the use of a Finite State Machine to reduce the number of internal states, which results in speeding up the recognition process. We also investigate to control the number of active states at each time step in the trade-off between waiting time and recognition accuracy. Finally, we show the effectiveness of merging the tokens in improving recognition accuracy of the decoding method.

6.2 Related works

6.2.1 Sequence to sequence learning with RNN

Graves et al. (Graves et al. 2009) use Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997), an advanced architecture of RNN designed to overcome the problem of vanishing or exploding gradients. LSTM could bridge long time delays between relevant input and target events, thus, it could incorporate long-range context for improving handwriting recognition.

CTC is an objective function for RNN designed to make RNN to learn directly from an input sequence to a target sequence without requiring pre-segmented input. It defines the output of RNN for each time *t* of the input sequence as the probability distribution over a fixed set of labels and an additional "blank" which denotes no label. The output of RNN for each label *k* (incl. *blank*) at time *t* is the conditional probabilities of observing label k at time *t* in the input sequence *x*:

$$y_k^t = p(k, t \mid x) \tag{6.1}$$

For an input sequence *x* of length *T*, the conditional probability of a path π through the lattice of output labels over all the time steps is calculated by multiplying the probabilities of labels along this path:

$$p(\pi \mid x) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{6.2}$$

where π_t is the label of path π at time *t*.

We also have the conditional probability of a sub-path of π spanning from a time *u* to time *v*:

$$p(\pi_{u:v} \mid x) = \prod_{t=u}^{v} y_{\pi_t}^t \tag{6.3}$$

where, for a sequence s , $s_{i:j}$ is the subsequence from s_i to s_j .

A label sequence is obtained from a path by a reduction process denoted as *B*, which firstly removes repeated labels, then removes *blanks* in this path. The probability of a label sequence *l* from an input sequence *x* is the total probability of all the paths, where each path is reduced into this label sequence by *B*. It is shown as follows:

$$p(l \mid x) = \sum_{B(\pi)=l} p(\pi \mid x) \tag{6.4}$$

Applying the CTC forward-backward algorithm (Graves et al. 2009), $p(l \mid x)$ in (6.4) is obtained efficiently. By minimizing the total negative log likelihood $-\ln p(z \mid x)$ over all pairs of

an input sequence *x* and a target label *z* from training patterns, the network could be trained with unsegmented patterns.

6.2.2 Constrained decoding

In this section, we focus on the problem of word decoding, i.e. finding the most probable word from an input sequence, where output words are constrained by a vocabulary.

Let *W* is a closed vocabulary, the problem is finding a word *w* in *W* for which the conditional probability from an input sequence *x* is maximum:

$$\hat{w} = \arg\max_{w \in W} p(w \mid x) \tag{6.5}$$

From (5), the probability $p(w \mid x)$ of the word w from the input sequence *x* in CTC is calculated by the following steps:

First, *w* is extended into *w'*, which contains *blank* between every pair of consecutive characters of *w* and the beginning and end of *w*. Thus, $|w'| = 2|w| + 1$.

We define a prefix label sequence as the result of applying a reduction process denoted as B' for a path. B' firstly removes repeated labels and then *blanks* except the *blank* being the last label of this path. For example, B'(-,f,-,e,-,-) yields the prefix label sequence (f,e,-).

Secondly, let P_s^t is the probability to output the prefix label sub-sequence up to w'_s of w' at time *t*:

$$P_s^t = \sum_{\pi : B'(\pi_{1:t}) = B'(w'_{1:s})} p(\pi_{1:t} \mid x) \tag{6.6}$$

Thirdly, the probability P_s^t is calculated recursively by the following initialization and recursion:

$$
\begin{aligned}
P_1^1 &= y_{w'_1}^1 \\
P_2^1 &= y_{w'_2}^1 \\
P_s^1 &= 0, \forall s > 2 \\
P_s^t &= y_{w'_s}^t \begin{cases} P_s^{t-1} + P_{s-1}^{t-1} \text{if } w'_s = blank \text{ or } w'_s = w'_{s-2} \\ P_s^{t-1} + P_{s-1}^{t-1} + P_{s-2}^{t-1} \text{ otherwise} \end{cases}
\end{aligned} \tag{6.7}
$$

Finally, we obtain $p(w \mid x)$ by:

$$
p(w \mid x) = P_{|w'|}^T + P_{|w'|-1}^T \tag{6.8}
$$

where T is the number of time steps of the input sequence.

An example of calculating probability of a word 'feel' using CTC is illustrated in Fig. 6.1.

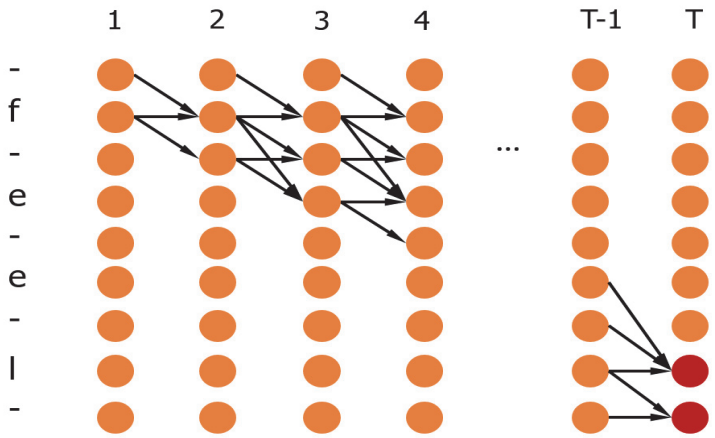


Figure 6.1 Example of calculating the probability of the word 'feel' in CTC.

Each node respresents the probability of P_s^t with t from 1 to T and s from 1 to 9 which is correllated with the extended word '_f_e_e_l_', where '_' denotes *blank*. Nodes shown in 'red' are the final nodes containing the probability of the word.

6.2.3 CTC token passing

The probability $p(l \mid x)$ in (6.4) could be approximated by:

$$p(l \mid x) \approx \max_{\pi, B(\pi)=l} p(\pi \mid x) \tag{6.9}$$

CTC token passing (Graves et al. 2009) uses this approximation for decoding words constrained by the closed vocabulary *W*. Hence, for the word decoding problem, we could apply Eq. (5) with $p(w \mid x)$ being calculated as follows:

$$p(w \mid x) = \max_{\pi, B(\pi)=w} p(\pi \mid x) \tag{6.10}$$

The probability P_s^t is now defined as:

$$P_s^t = \sum_{\pi: B'(\pi_{1:t})=B'(w'_{1:s})} p(\pi_{1:t} \mid x) \tag{6.11}$$

It is calculated recursively by the following initialization and recursion.

$$\begin{aligned} P_1^1 &= y_{w'_1}^1 \\ P_2^1 &= y_{w'_2}^1 \\ P_s^1 &= 0, \forall s > 2 \\ P_s^t &= y_{w'_s}^t \begin{cases} \max(P_s^{t-1}, P_{s-1}^{t-1}) \text{ if } w'_s = blank \text{ or } w'_s = w'_{s-2} \\ \max(P_s^{t-1}, P_{s-1}^{t-1}, P_{s-2}^{t-1}) \text{ otherwise} \end{cases} \end{aligned} \tag{6.12}$$

Then, we obtain the $p(w \mid x)$ by:

$$p(w \mid x) = \max(P_{|w'|}^T, P_{|w'|-1}^T) \tag{6.13}$$

For implementation, each label w'_s of the extended word *w'* has a single token which holds the probability P_s^t at time *t*. The probability P_s^t is the probability of the token with the highest probability reaching to w'_s at time *t* as specified in (6.11). Therefore, the probability $p(w \mid x)$ in (6.13) is obtained by the maximum probability of two tokens corresponding to the two last labels $w'_{|w'|-1}$ and $w'_{|w'|}$ of *w'* at the last time *T*.

Decoding a word sequence is an extension of decoding a word by employing an input token and an output token for each word.

At each time t, the output token of an extended word w' is the most probable token leaving w', whose probability is denotes as P_{-1}^{t}.

$$P_{-1}^{t} = \max(P_{|w'|}^{t}, P_{|w'|-1}^{t}) \tag{6.14}$$

At time t, the input token of an extended word w' is the most probable output token of the words arriving at w' at time t, which its probability is denote as P_{0}^{t}. Fig. 6.2 illustrates the CTC token passing for decoding a word sequence. As input tokens arrive at time t, the same recursive updating rule of (6.10) is applied to update the other tokens. As for the first two tokens:

$$\begin{aligned} P_{1}^{t+1} &= y_{w'_1}^{t+1} P_{0}^{t} \\ P_{2}^{t+1} &= y_{w'_2}^{t+1} \max(P_{0}^{t}, P_{1}^{t}) \end{aligned} \tag{6.15}$$

For decoding a word sequence, each token also records the path of words which are passed by. Thus, we get the most probable word sequence from the highest probability output token of words at the last time step T.

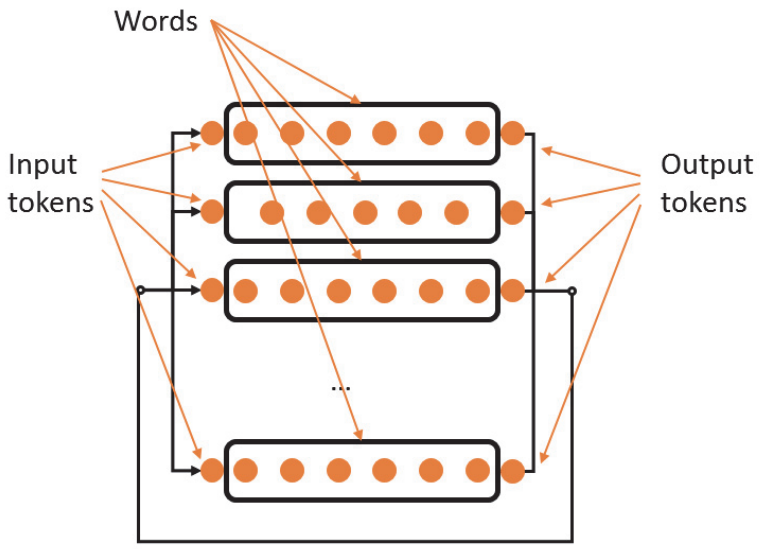


Figure 6.2 Decoding a word sequence with CTC token passing

6.3 Finite state machine token passing

6.3.1 Decoding word sequence with Lexicon State Machine

Firstly, we construct an equivalent Finite State Machine of CTC token passing for decoding words. In this state machine, for calculating $p(w \mid x)$ of a word *w*, the number of states excepting the start state is equal to |*w'*|. At time *t*, each state S_s represents the same probability P_s^t as in (6.6) and therefore, it accepts the path $\pi_{1:t}$ so that $B'(\pi_{1:t}) = B'(w'_{1:s})$. At time *t*, each state S_s only retains the most probable incoming path from the states in (*t-1*) and multiplies its probability with the probability of the w'_s to produce the probability of the state at *t*, as the same with recursive calculation of P_s^t in (6.12). Fig. 6.3 shows an example of the state machine for decoding the word 'feel'. The two final states (in double rounds, Fig. 6.3) at time *t* represent $P_{|w'|}^t$ and $P_{|w'|-1}^t$. We represent the vocabulary *W* using a state machine called lexicon state machine (LSM) by creating one state machine for each word in *W*, where all the word state machines are sharing the same start state S_0. To find the word with the highest probability from an input sequence of length *T*, we pass the tokens through the LSM until time *T* and find the highest probability token among those reaching the final states of LSM. We obtain the best word output by applying operator *B* over the path recorded by the highest probability token. We call the decoding method as LSM token passing.

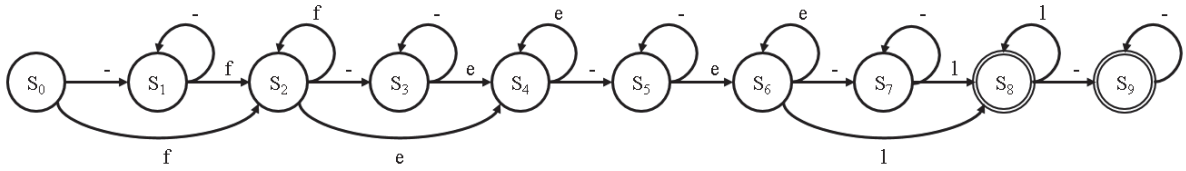


Figure 6.3 A FSM for decoding a word 'feel'.

For decoding a word sequence, we extend the LSM token passing method for decoding word by making recurrent connections from the final states of each word to the states following the start state of the LSM. Fig. 6.4 shows the LSM for decoding a word sequence.

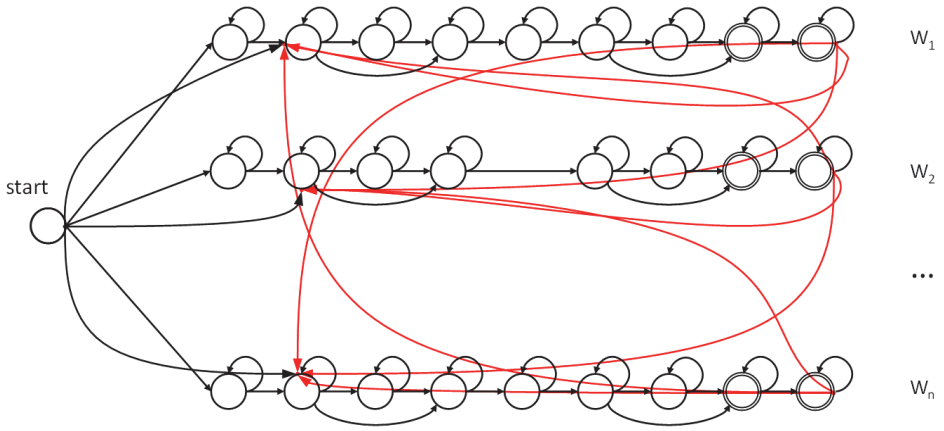


Figure 6.4 LSM for decoding the whole string.

Each state machine for W_1, W_2, .. W_n is sharing the same start state. The recurrent connection (shown in red) connect the final states to the state following the start state.

6.3.2 Reduction of states

As the number of states is large for large vocabulary recognition tasks, we could reduce it to speed up the decoding process.

The LSM constructed in section III.*A* is a non-deterministic finite automaton (NFA). We apply the determination and minimization to reduce the number of states in the LSM.

According to Theorem 2.1 and Theorem 3.10 in (Ullman and Hopcroft 1979), for a NFA *N*, there exists a deterministic automaton (DFA) D where $L(D) = L(N)$ and its minimization into a unique DFA *M* where $L(M) = L(D)$. Here, *L* of an automaton denotes the language accepted by the automaton.

Therefore, the DFA created by determinization and minimization of the LSM (we called it the minimized DFA) accepts all the paths as the same as the original LSM. Figure 6.5 shows an example of determinization and minimizing the LSM into a DFA.

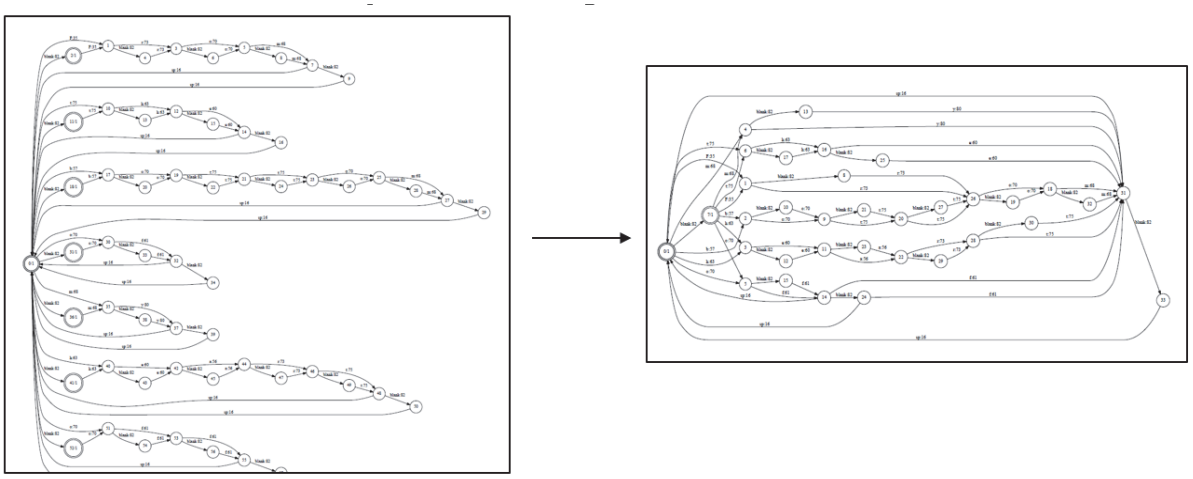


Figure 6.5 An example of determinizing and minimizing LSM.

Proposition 1. Applying token passing decoding through the states of the minimized DFA produce the same recognition accuracy with decoding through the original LSM.

Lemma 1. The best word sequence output of decoding through the original LSM is also the best word sequence output of decoding through the minimized DFA.

Proof. Let $\hat{w}$ is the best word sequence output of decoding through the original LSM. From the (6.6) and (6.10) there exists a best path $\hat{\pi}$ through the lattice of labels where $\hat{w} = B(\hat{\pi})$.

Since the minimized DFA accepts all the label paths as the same with the original LSM, for any path π accepted by the original LSM, we obtain the unique path of states θ through the states of the minimized DFA corresponding to π and vice versa. We denote this one-to-one mapping as operator F. We have the probability of a path θ from the start state to a final state of the minimized DFA with respect to input sequence *x*.

$$p(\theta \mid x) = p(\pi \mid x) \tag{6.16}$$

where $\pi = F^{-1}(\theta)$ is the label path accepted by original LSM.

Let $\hat{\theta} = \mathrm{F}(\hat{\pi})$ is the state path corresponding to $\hat{\pi}$. The path $\hat{\theta}$ is also the highest probable path of all the state paths θ through the minimized DFA according to (6.16).

Let S_{j1}, S_{j2}, S_{j3}, ... S_{jm} are the states of $\hat{\theta}$ which are also reached by other paths rather than $\hat{\theta}$. We call the states as junction states. Let t_{j1}, t_{j2} ,t_{j3}, ... t_{jm} are the time when $\hat{\theta}$ reaches the junction states. We prove the assertion that the path $\hat{\theta}$ is not pruned by other paths at any junction states at the time $\hat{\theta}$ reaches them.

Assumimg that there exists a path θ reaching to a junction state S_{jk} at time t_{jk} ($\theta_{t_{jk}} = \hat{\theta}_{t_{jk}} = S_{jk}$) and having higher probability than the path by $\hat{\theta}$ reaching to S_{jk}. We have:

$$p(\theta_{0:t_{jk}} \mid x) > p(\hat{\theta}_{0:t_{jk}} \mid x) \tag{6.17}$$

Let S_{jk+1} is the state following S_{jk} in the path $\hat{\theta}$, from (14):

$$p(\theta_{0:t_{jk}} \mid x)\, p(\hat{\theta}_{t_{jk}+1:T} \mid x) > p(\hat{\theta}_{0:t_{jk}} \mid x)\, p(\hat{\theta}_{t_{jk}+1:T} \mid x) \tag{6.18}$$

That is:

$$p(\theta_{0:t_{jk}} \mid x)\, p(\hat{\theta}_{t_{jk}+1:T} \mid x) > p(\hat{\theta} \mid x) \tag{6.19}$$

It turns out that the probabilty of the path created by concatenating $\theta_{0:t_{jk}}$ and $\hat{\theta}_{t_{jk}+1:T}$ is higher than that of the best path $\hat{\theta}$. This contradicts the above assumption and therefore the assertion is proved.

Thus, the path $\hat{\theta}$ is not pruned by any other paths and the token passing through $\hat{\theta}$ could reach the final states and become the most probable token. The best word sequence output of decoding through the minimized DFA is therefore equal to $B(F^{-1}(\hat{\theta})) = B(\hat{\pi}) = \hat{w}$. This completes the proof.

Proof of Proposition 1. Since recognition result of an input sequence is obtained from the best word sequence output of decoding method, from Lemma 1, recognition results by decoding through the original LSM are the same with those by decoding through the minimized DFA. Therefore, the recognition accuracy of both the decoding method are the same. This proves the proposition.

6.3.3 Merging of same paths

The token passing algorithm only preserves the path with the highest probability which comes to a state as shown in the approximation by (6.9). This approximation may reduce recognition accuracy.

To deal with the problem, before determining the best path for each state, we merge the paths which produce the same output label sequence by summing the probabilities of the paths. In Fig. 6.6 the path through '-','i' and the path through 'i' produce the same output 'i', therefore they are merged together.

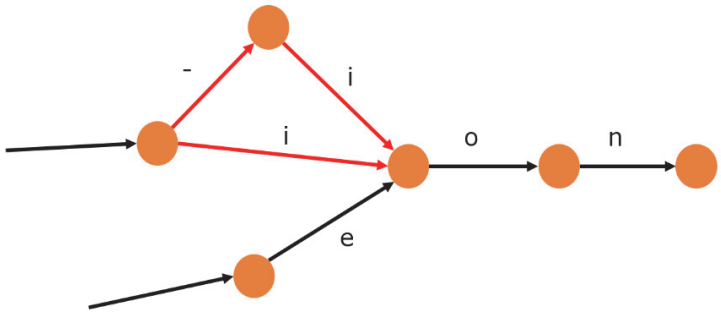


Figure 6.6 Merging of token paths

6.3.4 N-best state LSM decoding

For each time step, the decoding process must go through all the states of the LSM, but this process is heavily time consumptive. Therefore, we cut down low probability states to speed up the decoding process. We retain only the N-best states at each time step and use them for the next time step.

This reduction hence leads to reduction of recognition accuracy. However, with the trace off between recognition rate and waiting time, we could tune the parameter of N to compromise the recognition performance with hardware limitation.

6.4 Experiments

We implemented the LSM token passing algorithm as well as the original CTC token passing algorithm. The LSM is modeled, determinized and minimized using the Open-FST library (Allauzen et al. 2007). We made experiments on both the IAM online handwriting database (IAM-OnDB) (Marti and Bunke 2003) and the IBM-UB database (IBM_UB_1) (Shivram et al. 2013). We follow the IAM-OnDB-t2 handwritten text recognition task, in which the database is divided into a training set, two validation sets and a test set containing 5,364, 1,438, 1,518, and 3,859 written lines, respectively. For IBM_UB_1, the recognition task also use a training set, two validation sets and a test set with 31,888, 6,519, 6742, and 18,811 words, respectively.

We apply bidirectional LSTM (BLSTM) networks (Graves and Schmidhuber 2005) with CTC using RNNLIB (Graves 2013) for training on both of the databases. We extract a set of point-based features including: (1) normalized distance between two consecutive points, (2, 3) sine and cosine of the angle between the current line segment and the horizontal line, (4) pen-up/pen-down feature. We use two layers of BLSTM with a sub-sampling layer in the middle with the number of hidden nodes in each BLSTM layer being 32 and 64, respectively. The number of hidden nodes in sub-sampling is 48.

For decoding in each database, a closed vocabulary extracted from the test set is used to constrain the output. The vocabulary of IAM-OnDB and IBM_UB_1 contains 5,597 words and 4,962 words, respectively.

The results is shown in Table 6.1 for experiments on both IAM-OnDB and IBM_UB_1. LSM token passing through the minimized DFA yields the same recognition rate with the CTC token passing while the former makes reduction of waiting time over 56% in the experiment of IAM-OnDB and over 63% in the experiment of IBM_UB_1. Merging token paths yields better recognition rate than the CTC token passing algorithm even in the case of limiting the number of active states (N). Improvement of word recognition rate is about 0.06 point and 0.78 point as experiments on IAM-OnDB and IBM_UB_1, respectively. The best word recognition rate of the method for IAM-OnDB is 85.82%, which is better than the result in (Graves et al. 2009). Note that (Graves et al. 2009) use a sophisticated pre-processing and feature extraction method.

Limiting the number of active states at each time step (N-best state decoding) speeds up the recognition. As compared with CTC token passing, N-best state LSM decoding with setting N to 100 makes reduction of waiting time as 95.51% and 94.33% in IAM-OnDB and IBM_UB_1, respectively. In IBM_UB_1, setting N to 100 slightly degrades the recognition rate by 0.05 point, while the experiment with IBM_UB_1 shows the degradation of over 0.13 point. Here, we could see the trade-off between recognition speed and recognition accuracy of setting N in N-best path LSM decoding. For practical use, we choose $N = 200$ since the loss on recognition accuracy is not large and the waiting time acceptable .

Table 6.1 Performance of LSM token passing as compared with CTC token passing

Database	Measure	LSM token passing				CTC token passing	CTC token passing (Graves et al. 2009)
		Minimized DFA	*Merging paths – N-best*				
			100	*200*	*1000*		
IAM-OnDB	Word recognition rate (%)	85.76	85.69	85.78	85.82	85.76	85.30
	Waiting time (s)	26.37	2.71	4.30	24.75	60.42	-
IBM_UB_1	Word recognition rate (%)	92.70	93.43	93.47	93.48	92.70	-
	Waiting time (s)	1.43	0.22	0.35	1.94	3.88	-

6.5 Conclusions and Future works

We presented the Finite State Machine based token passing decoding method for online handwritten English text recognition. In this method, a finite state machine is used to model a closed vocabulary and the token passing method is applied through the state machine to obtain the most probable word sequence. Applying determination and minimization to the state machine reduces the number of states while preserving all the paths corresponding to the words in the vocabulary. As a result, waiting time for decoding is reduced by over 56% in IAM-OnDB and over 63% in IBM_UB_1 without degrading recognition rate.

Moreover, applying combination of tokens prevents information loss and therefore improves the recognition rate by 0.78 point in IBM_UB_1 and 0.06 point in IAM-OnDB. Limiting the number of active states of the state machine for each time step greatly reduces waiting time while degrading recognition rate slightly. With maintaining recognition rate higher than the original CTC token passing, the method reduces up to 92.88% and 94.33% of waiting time for IAM-OnDB and IBM_UB_1, respectively.

For future works, we will extend the method for decoding word sequence constrained by high level context (such as bi-gram, tri-gram).

Chapter 7

Conclusion and Future works

In this thesis, we presented a semi-incremental recognition method for on-line handwritten text. By employing local processing, average waiting time has been reduced. Moreover, determining *SP* off-strokes based on recognition result shortens block lengths, bounds waiting time and even increases the recognition rate slightly. Skipping the recognition of partial patterns and reusing recognized character patterns in the src-lattice are also shown to be effective in reducing the waiting time.

The semi-incremental recognition method is superior to the batch recognition method clearly in waiting time and even in recognition rate. It also excels the pure incremental recognition method in recognition rate and total CPU time.

The semi-incremental recognition method has been applied to Japanese text and English text, the method should also work for other languages by changing the parameters.

We also presented the improvement of segmentation which leads to the improvement of recognition accuracy and speed for online handwriting recognition system using recurrent neural networks (RNN).

For English text recognition, we also study the segmentation-free recognition method using the state of the art LSTM, we presented a Finite State Machine based decoding method for LSTM, which does not only reduce the waiting time of recognition but also improves the recognition accuracy of the recognition system.

For future works, we will extend the method for decoding word sequence constrained by high level context (such as bi-gram, tri-gram). Applying BLSTM to improve segmentation of online/offline Japanese handwriting recognition is also a good topic for future researches.

References

Allauzen, C., M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. 2007. “OpenFst: A General and Efficient Weighted Finite-State Transducer Library.” In *12th International Conference on Implementation and Application of Automata*, 11–23. Prague, Czech.

Graves, Alex. 2013. “RNNLIB: A Recurrent Neural Network Library for Sequence Learning Problems.” http://sourceforge.net/projects/rnnl/.

Graves, Alex, Santiago Fernandez, Faustino Gomez, and Jurgen Schmidhuber. 2006. “Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks.” *Proceedings of the 23rd International Conference on Machine Learning*, 369–76. doi:10.1145/1143844.1143891.

Graves, Alex, Marcus Liwicki, Horst Bunke, Juergen Schmidhuber, and Santiago Fernández. 2008. “Unconstrained On-Line Handwriting Recognition with Recurrent Neural Networks.” In *Advances in Neural Information Processing Systems*, 577–84.

Graves, Alex, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. “A Novel Connectionist System for Unconstrained Handwriting Recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (5): 855–68. doi:10.1109/TPAMI.2008.137.

Graves, Alex, and Jürgen Schmidhuber. 2005. “Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures.” *Neural Networks : The Official Journal of the International Neural Network Society* 18 (5–6): 602–10. doi:10.1016/j.neunet.2005.06.042.

Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. “Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies.” *A Field Guide to Dynamical Recurrent Neural Networks*.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9 (8). MIT Press: 1735–80. doi:10.1162/neco.1997.9.8.1735.

Jaeger, S., S. Manke, J. Reichert, and a. Waibel. 2001. “Online Handwriting Recognition: The NPen++ Recognizer.” *International Journal on Document Analysis and Recognition* 3 (3): 169–80. doi:10.1007/PL00013559.

Johansson, Stig. 1978. *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers*. Oslo: University of Oslo.

Liu, Cheng-Lin, Stefan Jaeger, and Masaki Nakagawa. 2004. “Online Recognition of Chinese Characters: The State-of-the-Art.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2). IEEE Computer Society: 198–213. doi:10.1109/TPAMI.2004.1262182.

Liwicki, Marcus, and Horst Bunke. 2006. “HMM-Based On-Line Recognition of Handwritten Whiteboard Notes.” In *Tenth International Workshop on Frontiers in Handwriting Recognition*. La Baule (France).

Liwicki, Marcus, Horst Bunke, James a. Pittman, and Stefan Knerr. 2011. “Combining Diverse Systems for Handwritten Text Line Recognition.” *Machine Vision and Applications* 22 (1): 39–51. doi:10.1007/s00138-009-0208-9.

Marti, U. V., and H. Bunke. 2003. “The IAM-Database: An English Sentence Database for Offline Handwriting Recognition.” *International Journal on Document Analysis and Recognition* 5 (1): 39–46. doi:10.1007/s100320200071.

Matic, N.P., J.C. Platt, and T. Wang. 2002. “QuickStroke: An Incremental on-Line Chinese Handwriting Recognition System.” *Object Recognition Supported by User Interaction for Service Robots* 3. doi:10.1109/ICPR.2002.1047941.

Matsushita, Tomohisa, and Masaki Nakagawa. 2014. “A Database of On-Line Handwritten Mixed Objects Named Kondate.” *2014 14th International Conference on Frontiers in Handwriting Recognition*, no. 1: 369–74. doi:10.1109/ICFHR.2014.68.

McDermott, Erik, Timothy J. Hazen, Jonathan Le Roux, Atsushi Nakamura, and Shigeru Katagiri. 2007. “Discriminative Training for Large-Vocabulary Speech Recognition Using Minimum Classification Error.” *IEEE Transactions on Audio, Speech, and Language Processing* 15 (1): 203–23. doi:10.1109/TASL.2006.876778.

Morik, Katharina, Peter Brockhausen, and Thorsten Joachims. 1999. “Combining Statistical Learning with a Knowledge-Based Approach - A Case Study in Intensive Care Monitoring.” In *The Sixteenth International Conference on Machine Learning* , 268–77. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.7528.

Nakagawa, Masaki, Kimiyoshi Machii, Naoki Kato, and Toshio Souya. 1993. “Lazy Recognition as a Principle of Pen Interfaces.” In *INTERACT ’93 and CHI ’93 Conference Companion on Human Factors in Computing Systems - CHI ’93*, 89–90. New York, New York, USA: ACM Press. doi:10.1145/259964.260121.

Nakagawa, Masaki, and Kaoru Matsumoto. 2004. “Collection of on-Line Handwritten Japanese Character Pattern Databases and Their Analyses.” *Document Analysis and Recognition* 7 (1). doi:10.1007/s10032-004-0125-4.

Nakagawa, Masaki, Bilan Zhu, and Motoki Onuma. 2005. “A Model of On-Line Handwritten Japanese Text Recognition Free from Line Direction and Writing Format Constraints.” *IEICE Transactions on Information and Systems* 88 (8). The Institute of Electronics, Information and Communication Engineers: 1815–22. http://ci.nii.ac.jp/naid/110003214382/en/.

Nguyen, Cuong-Tuan, Bilan Zhu, and Masaki Nakagawa. 2016. “Semi-Incremental Recognition of On-Line Handwritten Japanese Text.” *IEICE Transactions on Information and Systems* E99.D (10). The Institute of Electronics, Information and Communication Engineers: 2619–28. doi:10.1587/transinf.2016EDP7051.

Nguyen, Cuong Tuan, Bilan Zhu, and Masaki Nakagawa. 2013. “A Semi-Incremental Recognition Method for On-Line Handwritten Japanese Text.” In *2013 12th International Conference on Document Analysis and Recognition*, 84–88. IEEE. doi:10.1109/ICDAR.2013.25.

———. 2014. “A Semi-Incremental Recognition Method for On-Line Handwritten English Text.” *2014 14th International Conference on Frontiers in Handwriting Recognition*, 234–39. doi:10.1109/ICFHR.2014.47.

Oda, Hideto, Bilan Zhu, Junko Tokuno, Motoki Onuma, and Akihito Kitadai. 2006. “A Compact on-Line and off-Line Combined Recognizer.” *Icfhr*, 133–38. http://hal.inria.fr/inria-00104438/.

Plamondon, Réjean, and Sargur N. Srihari. 2000. “On-Line and Off-Line Handwriting Recognition : A Comprehensive Survey.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (1): 63–84. doi:10.1109/34.824821.

Schenkel, M., I. Guyon, and D. Henderson. 1995. “On-Line Cursive Script Recognition Using Time-Delay Neural Networks and Hidden Markov Models.” *Machine Vision and Applications* 8 (4): 215–23. doi:10.1007/BF01219589.

Shivram, Arti, Chetan Ramaiah, Srirangaraj Setlur, and Venu Govindaraju. 2013. “IBM-UB-1: A Dual Mode Unconstrained English Handwriting Dataset.” *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 13–17. doi:10.1109/ICDAR.2013.12.

Tanaka, Hiroshi. 2002. Implementation of real-time box-free online Japanese handwriting recognition system. 3925247, issued 2002.

Tanaka, Hiroshi, Katsuhiko Akiyama, and Kazushi Ishigaki. 2002. “Realtime Box-Free On-Line Handwriting String Recognition Using Layer-Delayed Segmentation Method.” *IEICE Technical Report (Institute of Electronics, Information and Communication Engineers)* 101 (712(PRMU2001 244-271)): 155–62.

Ullman, Jeffrey, and John Hopcroft. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Viterbi, A. 1967. “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm.” *IEEE Transactions on Information Theory* 13 (2): 260–69. doi:10.1109/TIT.1967.1054010.

Wang, Da Han, Cheng Lin Liu, and Xiang Dong Zhou. 2012. “An Approach for Real-Time Recognition of Online Chinese Handwritten Sentences.” *Pattern Recognition* 45 (10). Elsevier: 3661–75. doi:10.1016/j.patcog.2012.04.020.

Young, S.J., N.H. Russell, and J.H.S Thornton. 1989. “Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems.” *Technical Report (Univ. of Cambridge. Dept. of Engineering)* 38. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.7829.

Zhou, Xiang-Dong, Da-Han Wang, and Cheng-Lin Liu. 2009. “A Robust Approach to Text Line Grouping in Online Handwritten Japanese Documents.” *Pattern Recognition* 42 (9): 2077–88. doi:10.1016/j.patcog.2008.10.019.

Zhu, Bilan, JinFeng Gao, and Masaki Nakagawa. 2011. “Objective Function Design for MCE-Based Combination of On-Line and Off-Line Character Recognizers for On-Line Handwritten Japanese Text Recognition.” In *2011 International Conference on Document Analysis and Recognition*, 594–98. IEEE. doi:10.1109/ICDAR.2011.125.

Zhu, Bilan, and Masaki Nakagawa. 2008. “Segmentation of On-Line Freely Written Japanese Text Using SVM.” *IEICE Transactions on Information and Systems*, no. 1: 105–13. doi:10.1093/ietisy/e91.

———. 2014. “Building a Compact Online MRF Recognizer for Large Character Set by Structured Dictionary Representation and Vector Quantization Technique.” *Pattern Recognition* 47 (3). Elsevier: 982–93. doi:http://dx.doi.org/10.1016/j.patcog.2013.09.031.

Zhu, Bilan, Arti Shivram, Srirangaraj Setlur, Venu Govindaraju, and Masaki Nakagawa. 2013. “Online Handwritten Cursive Word Recognition Using Segmentation-Free MRF in Combination with P2DBMN-MQDF.” In *2013 12th International Conference on Document Analysis and Recognition*, 349–53. IEEE. doi:10.1109/ICDAR.2013.77.

Zhu, Bilan, Xiang-Dong Zhou, Cheng-Lin Liu, and Masaki Nakagawa. 2010. “A Robust Model for on-Line Handwritten Japanese Text Recognition.” *International Journal on Document Analysis and Recognition (IJDAR)* 13 (2): 121–31. doi:10.1007/s10032-009-0111-y.

Author publications

Journals

[1] Cuong Tuan Nguyen, Bilan Zhu and Masaki Nakagawa. **Semi-Incremental Recognition of On-Line Handwritten Japanese Text**. *IEICE Transactions on Information and Systems*, Vol. E99.D, No. 10, pp. 2619-2628, (2016)

International Conferences

[2] Cuong Tuan Nguyen, Bilan Zhu and Masaki Nakagawa: **A semi-incremental recognition method for on-line handwritten Japanese text**, Proc. 12th International Conference on Document Analysis and Recognition (ICDAR2013), Washington D.C., USA, pp.85-88 (2013.8). (Oral)

[3] Cuong Tuan Nguyen, Bilan Zhu and Masaki Nakagawa. **A semi-incremental recognition method for online handwritten English text**, in Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR2014), Crete, Greece, pp.234-239 (2014.9). (Poster)

[4] Cuong Tuan Nguyen and Masaki Nakagawa: **An Improved Segmentation of Online English Handwritten Text using Recurrent Neural Networks**, in Proc. of *the 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia (2015.11). (Poster)

[5] Cuong Tuan Nguyen and Masaki Nakagawa: **Finite State Machine based Decoding of Online Handwritten Text using Recurrent Neural Networks**, in Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR2016), Shenzhen, China, pp. (2016.10). (Oral)

Patent

[6] 中川正樹，朱碧蘭，グェン　トアン　クーン：プログラム、情報記憶媒体及び文字列認識装置，特願 2013-100118（出願 2013 年 5 月 10 日，出願人：東京農工大学学長），特許第 5807881 号（平成 27 年 9 月 18 日登録）

Joint work publications

International Conferences

[7] Hung Tuan NGUYEN, Cuong Tuan NGUYEN, Pham The BAO, Masaki NAKAGAWA: **A Vietnamese Online Handwriting Database**, in Proc. of *the 2015 Fourth ICT International Student Project Conference, Tokyo University of Agriculture and Technology*, Tokyo, Japan (2015.5). (Oral)

[8] Khanh Minh Phan, Cuong Tuan Nguyen, Anh Duc Le and Masaki Nakagawa: **An Incremental Recognition Method for Online Handwritten Mathematical Expressions**, in Proc. of *the 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia (2015.11). (Poster)

[9] Hung Tuan Nguyen, Cuong Tuan Nguyen, Pham The Bao and Masaki Nakagawa: **Preparation of an Unconstrained Vietnamese Online Handwriting Database and Recognition Experiments by Recurrent Neural Networks**, Proc. 15th *International Conference on Frontiers in Handwriting Recognition (ICFHR2016)*, Shenzhen, China, 2016. (2016.10) (Poster)

Domestic Conferences

[10] Hung Tuan Nguyen, Cuong Tuan Nguyen, Pham The Bao, Masaki Nakagawa: **Preparation of an Unconstrained Vietnamese Online Handwriting Database and Recognition Experiments by BLSTM**, *IEICE technical report*, Vol. 115, No. 517, pp.59-64 (2016.3). (Oral)