Pattern Recognition 51 (2016) 112–124

Contents lists available at ScienceDirect

Pattern Recognition

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

Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents

Truyen Van Phan, Masaki Nakagawa *

Department of Electronic and Information Engineering, Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho Koganei-shi, Tokyo 184-8588, Japan

ARTICLE INFO

Article history:
Received 26 September 2014
Received in revised form
14 April 2015
Accepted 24 July 2015
Available online 4 August 2015

Keywords:
Text/non-text classification
Ink stroke classification
Online handwritten documents
Heterogeneous documents
Recurrent neural networks
Long short-term memory

ABSTRACT

The task of text/non-text classification in online handwritten documents is crucially important to text recognition, text search, and diagram interpretation. It, however, is a challenging problem because of the large amount of variation and lack of prior knowledge. In order to solve this problem, we propose to use global and local contexts to build a high-performance classifier. The classifier assigns a text or non-text label to each stroke in a stroke sequence of a digital ink document. First, a neural network architecture is used to acquire the complete global context of the sequence of strokes. Then, a simple but effective model based on a marginal distribution is used for the local temporal context of adjacent strokes in order to improve the sequence labeling result. The results of experiments on available heterogeneous online handwritten document databases demonstrate the superiority and effectiveness of our context combination approach. Our method achieved classification rates of 99.04% and 98.30% on the Kondate (written in Japanese) and IAMonDo (written in English) heterogeneous document databases. These results are significantly better than others reported in the literature.

1. Introduction

The task of text/non-text classification in online handwritten documents is to classify handwritten strokes into two categories: text and non-text, where a stroke is a time sequence of pen-tip or finger-tip points recorded from pen-down to pen-up. This task can be used as a preprocessing step for text recognition, text search, or diagram interpretation. It is also a prerequisite to the selection of an appropriate engine for processing the handwritten objects further. The classification results are used to decide whether the text strokes should be sent to a handwriting recognizer or an ink search engine. On the other hand, non-text strokes can be grouped together and recognized as higher level graphical entities like flow-chart, finite automata, etc., by a diagram interpreter. Text and non-text classification can be extended for multi-class non-text classification as [1,2], but this paper focuses on text and non-text classification since it is most basic and generic for many applications.

In recent years, smart phones, tablets, tablet PCs, electronic whiteboards equipped with pen-based and touch-based handwriting interfaces have become popular with a vast number of people. Moreover, in the near future, electronic paper and paper-like PCs will become available. People are now able to take notes, draw sketches, and create diagrams on their mobile devices. Online pen-tip traces or finger-tip traces, which are also called digital ink, are a natural and efficient way to express ideas, draw up concepts or summarize knowledge without requiring people to pay any attention to the mode of input. Due to the heterogeneous mixture of text and graphics, however, the advent of digital ink has brought new challenges to document analysis and recognition systems. Many researchers have proposed methods to solve these problems [1,3–9] and several heterogeneous digital ink databases have been collected to evaluate their methods [9–12]. Fig. 1 shows examples of heterogeneous digital ink from commonly used databases: Kondate which is in Japanese [9] and IAMonDo which is in English [10].

In this paper, we propose a novel method to combine global and local contexts of a stroke sequence for the purpose of text/non-text classification in online handwritten documents. Global context refers to the feature vector sequence of an entire document and local context refers to the prediction of directly adjacent strokes. The method is simple but effective since the global context is determined from the output of a neural network and the local context is obtained from the relationship between temporally adjacent strokes. It also establishes a new level of performance for text/non-text classification in heterogeneous online handwritten documents.

The rest of this paper is structured as follows. Section 2 reviews related work on the classification of text and non-text contents in

* Corresponding author. Tel./fax: +81 423 88 7144.
E-mail address: nakagawa@cc.tuat.ac.jp (M. Nakagawa).

http://dx.doi.org/10.1016/j.patcog.2015.07.012
0031-3203/

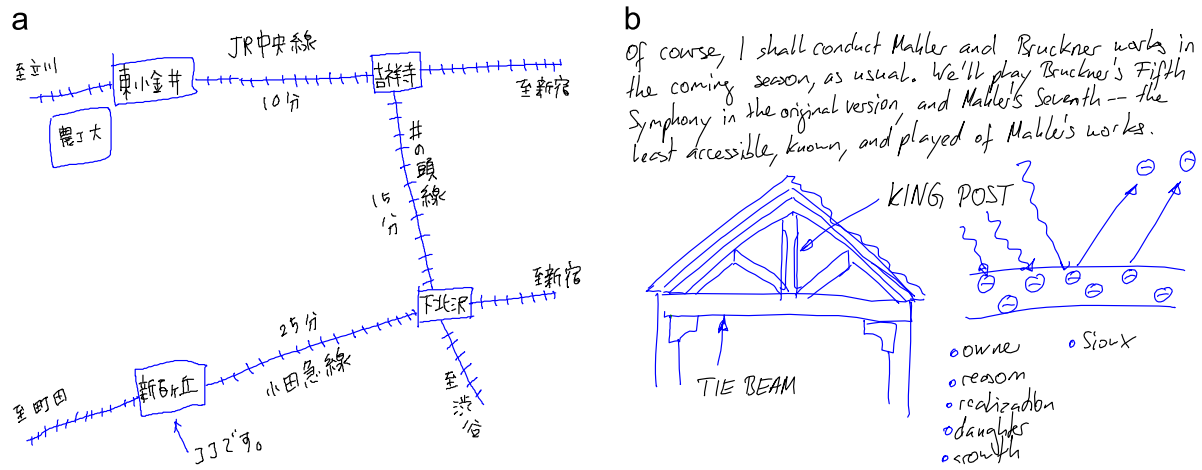

Fig. 1. Examples of heterogeneous digital ink from Kondate (a) and IAMonDo (b). Text strokes are shown in black, non-text strokes are shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Performance of various classifiers.

Database		*SVM*	*Our SVM*	*MLP*	*RNN*	*LSTM*	*BRNN*	*BLSTM*
Kondate *(11 features)*	*Rate* (%)	92.58 [4]	92.63	92.78	94.88	95.44	96.34	96.57
	Total time (s)	–	5.31	0.32	0.23	0.83	0.44	1.76
IAMonDo *(19 features)*	*Rate* (%)	94.44 [7]	93.09	93.74	96.57	97.25	97.47	97.72
	Total time (s)	–	20.55	0.31	0.25	0.83	0.44	1.78

heterogeneous online handwritten documents. This survey presents the requirements and ways of using contextual information in the text/non-text classification task. Section 3 overviews our approach, and Section 4 describes the method in detail and introduces recurrent neural network architectures that are used as classifiers. Section 5 evaluates the effectiveness of our approach on the available databases. Section 6 draws the conclusion.

2. Related work

The text/non-text classification task for digital ink is to classify online handwritten strokes into two categories: text and non-text. It is also called text/drawing segmentation [13], text/graphics separation [3,14], text/shape division [15–17], and so on. It is basically a two-category problem but text may be further divided into text and formulas [13], and non-text strokes may be further classified into several categories of graphics [18–21]. Another very similar task is mode detection [22–24]. It allows a user to write text and graphics without specifying or changing the mode in which the user is writing so that each graphic object is assumed to be written without switching to text. The difference between them lies in the input and output of the classification task. The input of text/non-text classification can be any mixed sequence of text and non-text strokes and the output is a sequence of labels for all the strokes, whereas the input of mode detection is a sequence of strokes for text or non-text without a specified mode and the output is the estimate of the mode. Although these tasks or problems are different, they share a number of technologies so both will be reviewed in this section.

2.1. Text/non-text classification

Many methods have been proposed for solving the text/non-text classification problem. They can be divided into three groups: isolated classification, context-integrated classification, and sequence classification according to how contextual information is made use of.

Isolated classification uses descriptions to classify strokes rather than exploiting any contextual information. Jain et al. [1] proposed a linear classifier to distinguish between text and non-text strokes represented by only two features: length and curvature. Isolated classification can also be employed before context-integrated classification [3–9].

Context-integrated classification makes use of several sources of contextual information (local, spatial and temporal) to improve classification accuracy. Mochida et al.'s study [9] was an early attempt. They used the stroke size feature to classify the digital ink and then modified the classification by taking into account stroke crossings. They further classified text into Japanese text and formulas based on the stroke density as well as on text and formula recognition scores. Bishop et al. [3], Zhou and Liu [4], and Delaye et al. [5–7] proposed probabilistic graphical models: Hidden Markov Models (HMMs), Markov Random Fields (MRFs), and Condition Random Fields (CRFs) for better integrating interactions between neighboring strokes. Bishop et al. [3] used a multilayer perceptrons (MLPs) for isolated classification to acquire the probability of a stroke being text or non-text. Then, they used a HMM model to incorporate temporal contexts between adjacent strokes. On the other hand, Zhou and Liu [4] and Delaye et al. [5–7] used support vector machines (SVMs) for isolated classification of single strokes and classification of stroke pairs before the context-integrated classification. The probabilities of single strokes and stroke pairs were created by fitting the SVM outputs to sigmoid functions. Zhou and Liu [4] incorporated spatial interactions between neighboring strokes in their MRF model. Furthermore, Delaye et al. [5–7] integrated multiple sources of context. They used a CRF framework to present the interactions between strokes in terms of neighboring systems and clique potentials. They proposed five neighboring systems describing different sources of contextual information for stroke classification: spatial system, temporal system, intersecting system, lateral system and stroke continuation system. The combination of all these systems performed the best. These methods demonstrate the superiority of

probabilistic graphical models taking advantage of context over the isolated classification of MLPs or SVMs.

Finally, recently developed long short-term memory (LSTM), a type of recurrent neural network (RNN), has been used for text/non-text classification [8]. In particular, bidirectional LSTM (BLSTM) has shown better performances compared with other algorithms for classification of time-series sequence patterns in speech recognition [25] and handwriting recognition [26]. BLSTM has been applied to online handwriting documents represented as streams of local feature vectors of sampling points. The network translates them into sequences of labels representing text and non-text. The network is bidirectional so that the classification is influenced by temporal context in both directions: forward and backward. The method yields a very accurate classification on the IAMonDo database.

Of all approaches, those of Zhou and Liu [4] and Delaye et al. [7] have state-of-the-art performances for Japanese (the Kondate database) and English (the IAMonDo database). Their classification rates are 96.61 % and 97.23%. For that reason, we decided to compare our approach with these methods.

2.2. Mode detection

The mode detection task for digital ink is to detect the mode (text or non-text) of online handwritten strokes. It is helpful for real-time applications and improves the user interfaces of tablet PCs and similar pen-based devices. It runs on-the-fly while the user is writing on the surface of the devices.

There have been several studies on mode detection. Most of them conducted evaluations on the IAMonDo database, which serves as a benchmark. Indermuhle et al. [10] presented two methods to solve the mode detection problem. The first method follows after Jain et al. [1] by taking into account two simple features of a trace: length and accumulated curvature. The method achieved a classification rate of 91.3% on IAMonDo. The second method considers only offline information. After performing connected component analysis on a document, features are extracted from individual connected components. This method achieved 94.4% correct classification of pixels. The classifiers used in both systems are SVMs. Weber et al. [23] developed a multiple classifier system (MCS). It was an improvement of Liwicki et al.'s system [22] that added new features, applied feature selections, used several state-of-the-art recognizers and performed multiple classifier combination strategies. Its classification rate was 97%. Recently, Otte et al. [24] proposed a novel approach for online mode detection. The approach works on local features within an RNN model: a standard RNN or a more complex LSTM. Their LSTM approach outperformed other state-of-the-art classifiers including SVMs and MCS and had an accuracy of 98.47% on average.

3. Outline of our approach

In order for text/non-text classification to be practical, it should be highly accurate and quick. The context-integrated approaches in [4,7] have state-of-the-art performances; however, they are too costly due to their use of a SVM for single stroke classification and three SVMs for stroke pair classification. On the other hand, the sequence classification in [8] has a small time complexity since it simply considers interactions between pen-tip or finger-tip points. Its accuracy of 97.01% is not high enough to be useful.

Hence, we decided to take a context-integrated sequence classification approach to satisfy the requirements of accuracy and speed. Our approach is similar with these approaches in some points. It classifies single strokes and stroke pairs as in [4,7], and it uses recurrent neural network for classification as in [8]. The differences with [4,7], however, are to use RNNs instead of SVMs for isolated classification and to use only one RNN instead of three SVMs for stroke pair classification. On the other hand, the differences with [8] are the way the feature is extracted and the way RNN is used. The features in [8] are extracted on each point while our features are on each stroke. RNN in [8] is used as a sequence transcriber which transcribes a sequence of point feature vectors into a sequence of stroke labels (m points to n labels), while RNN we use is a sequence classifier which classifies each stroke represented by a feature vector as text or non-text label (n strokes to n labels).

In preliminary experiments, we tested RNNs to compare with SVMs in the same number of features as in [4] on Kondate (11 features) and as in [7] on IAMonDo (19 features). Our testing SVMs are used to estimate the classification time of the SVMs in [4,7]. The classification time is the total time for all documents in the testing dataset. The classification result of classifiers is shown in Table 1. It is obvious that RNNs outperform SVMs on both accuracy and speed. While SVMs and MLPs do not allow access to past context or future context in the classification, traditional RNNs (RNNs and LSTMs) allow access to past context, and bidirectional RNNs (BRNNs and BLSTMs) allow access to both past and future contexts. The accuracies of bidirectional RNNs are hence better than traditional RNNs, SVMs and MLPs.

In our use of RNNs for classification, the context is retrieved in the scope of a whole document so that it said to be global. Due to the large scope of the whole document, however, its effect is blurred to each stroke. Therefore, we combine the influence of neighbors on a stroke by considering the prediction of adjacent strokes. We use another RNN to classify stroke pairs. Since the network processes a sequence in temporal order, our local context is temporal. Our approach is a strategy that integrates global and local contexts.

The structure of our approach can be divided into three stages as shown in Fig. 2. The first stage of the classification applies a global sequence labeling prediction model (RNN) to all strokes, based on a fixed size of feature descriptors for each stroke. Another RNN is also applied to all stroke pairs. The second stage integrates the prediction of stroke pairs into the prediction of a single stroke by operating marginalization on each stroke based on local temporal neighborhood. The last stage combines the global-

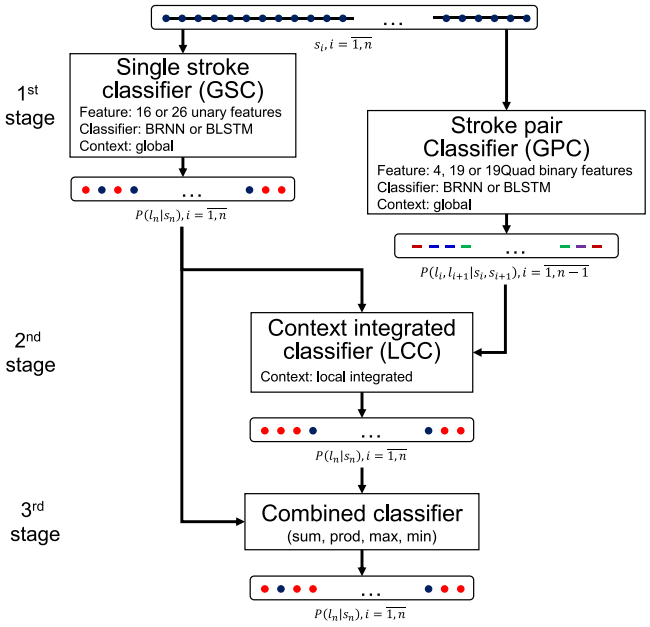

Fig. 2. Structure diagram of our approach.

Table 2
Unary features extracted from a single stroke and its local context.

Feature type		#	*Feature description*
Contour-based shape features	*Holistic*	*1*	Trajectory length
		2	Perimeter length
		3	Width of the bounding box
		4	Height of the bounding box
		5	Area of the convex hull
		6	Duration of the stroke
		7	Compactness
		8	Eccentricity
		9	Ratio of the principal axis of the stroke seen as a cloud of points
		10	Rectangularity of the minimum area bounding rectangle of the stroke
		11	Normalized centroid offset along major axis
		12	Ratio between first-to-last point distance and trajectory length
	Structural	*13*	Circular variance
		14	Accumulated curvature
		15	Accumulated perpendicular
		16	Accumulated signed perpendicular
Context features	*Spatial*	*17*	Number of spatial neighbour strokes
		18	Average of the distances from the stroke to spatial neighbour strokes
		19	Standard deviation of the distances from the stroke to spatial neighbour strokes
		20	Average of lengths of spatial neighbour strokes
		21	Standard deviation of lengths of spatial neighbour strokes
	Temporal	*22*	Number of temporal neighbour strokes
		23	Average of the distances from the stroke to temporal neighbour strokes
		24	Standard deviation of the distances from the stroke to temporal neighbour strokes
		25	Average of lengths of temporal neighbour strokes
		26	Standard deviation of lengths of temporal neighbour strokes

based classifier and the local-integrated classifier to achieve better performance.

Our main contributions are as follows. Firstly, we investigate and add new features including context features. Secondly, we make use of bidirectional networks to gain access to the global context of the whole document. Thirdly, we employ a simple but effective model to access the local context with temporally adjacent strokes. Finally, we propose multiple classifier combination strategies for combining global and local contexts to improve classification accuracy as much as possible.

4. Combination of global and local contexts

In this section, we present the context integration approach in detail. First, we present the use of a recurrent neural network to acquire global context and classify every stroke into text or non-text. Next, we describe the integration with local context of adjacent strokes. Then, we describe the combination of these contexts to achieve better performance. At last, we briefly introduce the recurrent neural network architecture which we use as a global context based classifier to label a sequence of single strokes or stroke pairs.

4.1. Global context based classifier

4.1.1. Global context model

The global context model is a bidirectional recurrent neural network which can map the entire history of input to each output in a document. Input is a sequence of feature vectors which is extracted from strokes in an online handwritten document and output is a sequence of labels for every stroke. A stroke is a time sequence of pen-tip points recorded between a pen-down event and a pen-up event. A feature vector, denoted by $\boldsymbol{s}$ is extracted from every stroke. The training samples consist of a set of N ordered strokes with feature vectors s_n, where $n=1,\ldots,N$ and

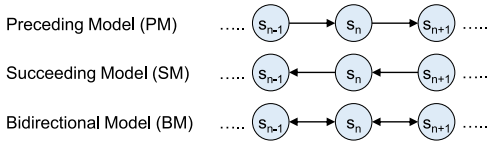

Fig. 3. Different correlations between a stroke with its adjacent strokes.

class labels $l_n \in \{0,1\}$ such that $l_n=1$ denotes a text stroke and $l_n=0$ denotes a non-text stroke. In this model, the neural network classifies a sequence of feature vectors to a sequence of labels. For classification of single strokes, we train the neural network using back propagation through time (BPTT). The output $y_n = y(s_n)$ of the resulting network model represents the probability of a stroke being text given the feature vector s_n. The probability distribution of l_n is a Bernoulli distribution expressed as

$$\boldsymbol{P}(\boldsymbol{l_n}\,|\,\boldsymbol{s_n}) = \boldsymbol{y_n}^{\boldsymbol{l_n}}(1-\boldsymbol{y_n})^{1-\boldsymbol{l_n}} \tag{1}$$

Hence, from the probability output of the network model, we decide the label of the stroke as follows:

$$\boldsymbol{l_n} = \begin{cases} 1 & \textbf{if } \boldsymbol{y_n} \geq 0.5, \\ 0 & \textbf{otherwise}. \end{cases} \tag{2}$$

We call this classifier the Global context based Single stroke Classifier (GSC) since it is a sequence classifier predicting one label for each stroke based on global context.

4.1.2. Single stroke features

For classifying a single stroke into two categories, we exploit 16 contour-based shape features and 10 context features. They are all unary features. These features have been shown to be effective in text/non-text classification [4–7] and mode detection [22,23]. The shape features are extracted from each stroke directly, while the context features are extracted by considering its interactions with

Table 3
Binary features extracted from a pair of two adjacent strokes.

Feature type	#	*Feature description*
Basic	1	Minimum distance between 2 strokes
	2	Minimum distance between the endpoints of 2 strokes
	3	Maximum distance between the endpoints of 2 strokes
	4	Distance between the centers of the 2 bounding boxes of 2 strokes
Additional symmetric	5	Horizontal distances between the centroids of 2 strokes
	6	Vertical distances between the centroids of 2 strokes
	7	Off-stroke distance between 2 strokes
	8	Off-stroke distance projected on *X* and *Y* axes
	9	Temporal distance between 2 strokes
	10	Ratio of off-stroke distance to temporal distance
	11	Ratio of off-stroke distance projected on *X*,*Y* axes to temporal distance
	12	Ratio of area of the largest bounding box of 2 strokes to that of their union
Asymmetric	13	Ratio of widths of the bounding boxes of 2 strokes
	14	Ratio of heights of the bounding boxes of 2 strokes
	15	Ratio of diagonals of the bounding boxes of 2 strokes
	16	Ratio of areas of the bounding boxes of 2 strokes
	17	Ratio of lengths of 2 strokes
	18	Ratio of durations of 2 strokes
	19	Ratio of curvatures of 2 strokes

neighboring strokes. Our goal here is to investigate the effect of context features in text/non-text classification. Hence, two sets of features with and without context features that contain 16 and 26 features, respectively, are used to build two binary classifiers. The first set contains contour-based shape features that can be divided into holistic and structural features. Holistic features (#1–#12) are extracted from the dimensions of a stroke including length, area, and compactness. On the other hand, structural features (#13–#16) are extracted from the trajectory of a stroke, like curvature, perpendicularity, etc. The second set compiles five features (#17–#21) of the spatial context and five features (#22–#26) of the temporal context. All these unary features are listed in Table 2.

4.2. Local context integrated classifier

4.2.1. Local context model

The temporal distribution of strokes in an online document obviously provides extremely valuable information for the stroke classification task. It has been proven that the interactions between strokes based on temporal information are more informative than spatial information, intersection, lateral information or stroke continuation interactions [5–7]. Furthermore, it is obvious that the context is influenced by the distance between strokes. The closer two strokes are in the document, the more they will share the same context. The context of the interaction between more widely separated strokes can have a negative effect, so we decided to consider only temporal context between adjacent strokes. Another advantage of this approach is that it does not require any computation to determine whether two strokes are neighbors or not.

In order to exploit the temporal context of two adjacent strokes, we propose to use marginal distribution. Given two events X and Y whose joint distribution is known; the marginal distribution of X is simply the probability distribution of X averaged over the information about Y. In other words, it is typically calculated by summing or integrating the joint probability distribution over Y.

$$\boldsymbol{P}(\boldsymbol{X}=\boldsymbol{x})=\sum_{\boldsymbol{y}}\boldsymbol{P}(\boldsymbol{X}=\boldsymbol{x}|\,\boldsymbol{Y}=\boldsymbol{y})\boldsymbol{P}(\boldsymbol{Y}=\boldsymbol{y}) \tag{3}$$

In our model, the marginal distribution of a stroke with feature vector s_n is calculated by integrating the joint probability distribution of its preceding stroke and/or succeeding stroke.

Three types of local context models are shown in Fig. 3. We call them the preceding, succeeding, and bidirectional model (PM, SM, and BM). Their probabilities of a stroke being text correlated with its adjacent ones are as follows:

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{PM}}(\boldsymbol{l_n}=1|\,\boldsymbol{s_n})=&\boldsymbol{P}(\boldsymbol{l_n}=1,\boldsymbol{l_{n-1}}=1|\,\boldsymbol{s_n},\boldsymbol{s_{n-1}})\boldsymbol{P}(\boldsymbol{l_{n-1}}=1|\,\boldsymbol{s_{n-1}})\\&+\boldsymbol{P}(\boldsymbol{l_n}=1,\boldsymbol{l_{n-1}}=0|\,\boldsymbol{s_n},\boldsymbol{s_{n-1}})\boldsymbol{P}(\boldsymbol{l_{n-1}}=0|\,\boldsymbol{s_{n-1}})\end{aligned} \tag{4}$$

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{SM}}(\boldsymbol{l_n}=1|\,\boldsymbol{s_n})=&\boldsymbol{P}(\boldsymbol{l_n}=1,\boldsymbol{l_{n+1}}=1|\,\boldsymbol{s_n},\boldsymbol{s_{n+1}})\boldsymbol{P}(\boldsymbol{l_{n+1}}=1|\,\boldsymbol{s_{n+1}})\\&+\boldsymbol{P}(\boldsymbol{l_n}=1,\boldsymbol{l_{n+1}}=0|\,\boldsymbol{s_n},\boldsymbol{s_{n+1}})\boldsymbol{P}(\boldsymbol{l_{n-1}}=0|\,\boldsymbol{s_{n+1}})\end{aligned} \tag{5}$$

$$\boldsymbol{P}_{\boldsymbol{BM}}(\boldsymbol{l_n}=1|\,\boldsymbol{s_n})=\boldsymbol{P}_{\boldsymbol{PM}}(\boldsymbol{l_n}=1|\,\boldsymbol{s_n})+\boldsymbol{P}_{\boldsymbol{SM}}(\boldsymbol{l_n}=1|\,\boldsymbol{s_n}) \tag{6}$$

The probability of a stroke being non-text $P_{PM}(l_n=0|\,s_n)$, $P_{SM}(l_n=0|\,s_n)$, or $P_{BM}(l_n=0|\,s_n)$ is computed similarly. Although it should be theoretically the product rather than the sum if the constituent terms are independent, its estimation error is fatal so that we use the sum as often used in ensemble classifiers [27]. The stroke labeling result in the local context model is then determined by:

$$\boldsymbol{l_n}=\begin{cases}1 & \text{if } P(\boldsymbol{l_n}=1|\,\boldsymbol{s_n})\geq\boldsymbol{P}(\boldsymbol{l_n}=0|\,\boldsymbol{s_n}),\\ 0 & \text{otherwise.}\end{cases} \tag{7}$$

We call this classifier the Local Context integrated Classifier (LCC) since it integrates the local context into the global context based classifier.

We use a single stroke classifier in the global context model to get the probabilities $P(l_n|\,s_n)$, whereas a binary SVM classifier was used to get them in [4–7]. We use only one neural network classifier, i.e. a ternary or a quaternary classifier, to get the probabilities $P(l_n, l_{n-1}|\,s_n, s_{n-1})$, whereas three binary SVM classifiers were used to get them in [4,5]. We call this classifier the Global context based stroke Pair Classifier (GPC) since it is based on the global context as the same as the single stroke classifier. We train the GPC classifier in the same way as the single stroke classifier by using a neural network. This neural network is applied to a sequence of feature vectors extracted from pairs of two temporally adjacent strokes in an online handwritten document. If the document has n strokes, there are $(n-1)$ pairs of two adjacent strokes. The classifier is ternary or quaternary, depending on whether the temporal order of the two adjacent strokes is considered or not when they are different in type. In the case of the ternary classifier, a text:non-text pair is assumed to be

equivalent to a non-text:text pair as the same as in [4–7]. On the other hand, the quaternary classifier distinguishes between text: non-text and non-text:text, implying that the stroke pairs are classified into four categories: text:text, text:non-text, non-text: text, non-text:non-text.

4.2.2. Stroke pair features

To classify a pair of adjacent strokes into three or four categories, we exploit a set of 12 symmetrical and seven asymmetrical measures. They are binary features. These features representing the relationship between two neighboring strokes were used in [4–7]. We extract them from every pair of temporally adjacent strokes. While the symmetrical features are independent of the temporal order of the strokes, the asymmetrical features are not. Our aim here is to investigate the effect of these features, we use a set of four basic features, as in [4], and a complete set of 19 features are then used to build two versions of the ternary classifiers. Since there are asymmetrical features in the latter set, it is also used to build the quaternary classifier. All these binary features are listed in Table 3.

4.3. Combined classifier

Since a combination of classifiers is more accurate than a single classifier in most cases, we combine the classifier of the global context model with that of the local context model. In this paper, we employ four basic combination rules as listed below:

Sum rule (SUM):

$$\boldsymbol{l_n^* = argmax}\left\{\sum_{k=1}^{K} \boldsymbol{f_k(l_n \mid s_n)}, \boldsymbol{l_n} \in \{0,1\}\right\} \tag{8}$$

Product rule (PROD):

$$\boldsymbol{l_n^* = argmax}\left\{\prod_{k=1}^{K} \boldsymbol{f_k(l_n \mid s_n)}, \boldsymbol{l_n} \in \{0,1\}\right\} \tag{9}$$

Max rule (MAX):

$$\boldsymbol{l_n^* = argmax}\{\boldsymbol{max_{k=1}^{K} f_k(l_n \mid s_n)}, \boldsymbol{l_n} \in \{0,1\}\} \tag{10}$$

Min rule (MIN):

$$\boldsymbol{l_n^* = argmax}\{\boldsymbol{min_{k=1}^{K} f_k(l_n \mid s_n)}, \boldsymbol{l_n} \in \{0,1\}\} \tag{11}$$

where $K=2, f_1(l_n \mid s_n)$ is the probability distribution of l_n calculated by the GSC classifier, i.e., (1) and $f_2(l_n \mid s_n)$ is one of the three probability distributions of l_n calculated by the LCC classifier, i.e., (4), (5) or (6).

Furthermore, we weighted each of the classifiers so that the final ensemble would reflect the reliability of each of them. We tried adjusting the contributions of the component classifiers with two functions:

a linear function

$$\boldsymbol{f_k'(l_n \mid s_n)} = \boldsymbol{\lambda_{k,1} f_k(l_n \mid s_n) + \lambda_{k,2}} \tag{12}$$

or an exponential function

$$\boldsymbol{f_k'(l_n \mid s_n)} = \boldsymbol{f_k(l_n \mid s_n)^{\lambda_k}} \tag{13}$$

where $0 \le \lambda_k \le 1$ is the classifier weighting parameter that controls the contribution of the classifier to get the probability of a stroke being text or non-text. There are two component classifiers and four combination rules so that there are totally 16 parameters for the linear functions and eight parameters for the exponential functions. Given the above definition of λ_k, it is obvious that if λ_k equals 1 then the kth classifier is fully reliable. Moreover, λ_k approaches 0, the kth classifier becomes less and less reliable. The weighting parameters are optimized by running a genetic algorithm on the validation dataset.

4.4. Recurrent neural network

The task of classifying single strokes or pairs of strokes in an ink document can be viewed as a sequence labeling task. Here, a sequence of feature vectors is transcribed into a sequence of labels. Recurrent neural network (RNN) has been applied with remarkable success to the field of speech and handwriting recognition. They are able to access the past as well as the future inputs (in the case of bidirectional RNN) of a stroke sequence. Therefore, we decided to employ a bidirectional architecture and the recently developed LSTM to solve the text/non-text classification problem.

A recurrent neural network is an artificial neural network (ANN) that allows cyclical connections. It is different from a multilayer perceptron (MLP) [28], an acyclic ANN; whereas an MLP can only map from input to output vectors, an RNN can map from the entire history of the previous inputs to each output. For this reason, MLP is suitable for pattern classification, while RNN is good for sequence labeling. The standard RNN, however, only considers the past context. The bidirectional recurrent neural network (BRNN) [29] offers a solution to the problem. It uses a finite sequence to label each element of the sequence on the basis of both the past and future context. This is done by adding the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left.

Among the various RNNs, LSTM [30] overcomes the main problem of standard RNNs, the vanishing gradient problem. The problem is that error gradients vanish exponentially over time as new inputs overwrite the activation of hidden units and the network almost forgets the first inputs. In order to solve this problem, the hidden layers in the LSTM architecture are made up of LSTM blocks instead of simple nodes.

In our architecture, the size of the input layer is the number of the features extracted from single strokes and pairs of strokes. The number of hidden layers is two, and they consist of k and l units that use a hyperbolic tangent (tanh) function for activation. The size of the output layer is two (text and non-text) in the case of the single stroke classifier (GSC) and three or four (text:text, non-text: non-text, text:non-text or/and non-text:text) in the case of the stroke pair classifier (GPC). The output activation functions used for RNNs are the same as standard ANNs: logistic sigmoid for binary classification and soft-max for multiclassification.

5. Experiments

We performed experiments to evaluate the contribution of global and local contexts to the task of text/non-text classification in heterogeneous online handwritten documents and to show the effect of combining these contexts. First, we describe the databases for the experiments. Next, we describe the feature normalization and present the settings for feature extraction and network training. After that, we show the results of the classifiers using global context, local context, and their combination. Finally, qualitative analyses are conducted.

Here, we employ each page in the databases as a whole document to derive the global context since each page stores handwriting on some topic by a single writer.

5.1. Databases

We evaluated our method on four databases of heterogeneous handwritten documents: the Japanese text database Kondate [9], the English text database IAMonDo [10], the flowchart database FC [11] and the finite automata database FA [12], which is available at http://cmp.felk.cvut.cz/~breslmar/finite_automata. Furthermore, save for the FA database, we used these databases to compare

our method with the previous ones using the same benchmarks for the text/non-text classification task. Besides the holdout validation method, in order to limit problems like overfitting when evaluating the true ability of the proposed method, we performed a four-fold cross validation on the two main databases, IAMonDo and Kondate.

The Kondate database has 669 digital ink document pages acquired from 67 writers, 10 pages per writer, except nine pages for the last writer. It has been used in text/non-text classification experiments reported in [4]. For the holdout validation, 310 pages were used for training the classifiers and 359 pages were used for testing. Hence, in our experiments, we used 210 pages for training, 100 pages for validation from the above training dataset, since we needed samples for validation, and the same 359 pages for testing. For the four-fold cross validation, the database was split into 4 disjoint sets each consisting of approximately 170 documents. No two document pages from the different sets were created by the same writer. Four sets were indexed from 0 to 3; two sets ($0+i$ mod 4) and ($1+i$ mod 4) were used for training, one ($2+i$ mod 4) for validation, and one ($3+i$ mod 4) for testing, where $i=0,\ldots,3$. Table 4 summarizes the number of document pages and strokes in each set, as well as the number and proportion of text (T) and non-text (N) strokes in these datasets for holdout and cross validation.

The IAMonDo database consists of about 1,000 heterogeneous online document pages produced by 200 writers. The database is split into five disjoint sets, each consisting of approximately 200 documents. It has been used for experiments on text/non-text classification [5,8]. For the holdout validation, 403 pages of set 1 and 2 were used for training, 200 pages of set 3 were used for validation and 203 pages of set 4 were used for testing. For the four-fold cross validation, the four sets from 1 to 3, except for 4, were used in the same way as in the Kondate database. Additionally, as in [24], we performed experiments on two particular sub-databases of tables and diagrams extracted from the complete database (called Tables_IAM and Diagrams_IAM). Tables_IAM contains straight lines as non-text and are therefore easy to distinguish from text, while Diagrams_IAM has much more variation, making them harder to separate from text. Table 5 summarizes the number of document pages and strokes, together with the number and proportion of text (T) and non-text (N) strokes in these databases.

The FC database has a total of 419 flowcharts drawn by 46 writers. In previous studies, 248 flowcharts were used for training and 171 flowcharts were used for testing flowchart recognition [31,32]. In our experiments, we used 187 diagrams for training and 61 diagrams for validation from the above training dataset. Testing was conducted on the remaining 171 flowcharts in the same way as the above. On the other hand, the experiment on text/non-text classification in [33] employed only 200 diagrams for training.

Lastly, the recently collected FA database contains 300 finite automata diagrams acquired from 25 persons. The database is divided into 132, 84, and 84 diagrams for training, validation, and testing, respectively.

Table 6 summarizes the number of flowcharts and finite automata diagrams as well as the number and proportion of text (T) and non-text (N) strokes in the datasets of the two databases FC and FA.

Table 4
Statistics for datasets in the Kondate database.

Validation method	*Dataset*	*#Pages*	*#Strokes*	*#T*	*#N*	*%T:%N*
Holdout	*Training*	210	41,190	34,406	6,784	83.53:16.47
	Validation	100	18,525	15,725	2,800	84.89:15.11
	Testing	359	71,846	61,384	10,462	85.44:14.56
Cross validation	*Set 1*	170	33,633	28,293	5,340	84.12:15.88
	Set 2	170	32,412	27,240	5,172	84.04:15.96
	Set 3	170	34,217	29,054	5,163	84.91:15.09
	Set 4	159	31,299	26,928	4,371	86.03:13.97

Table 5
Statistics for datasets in the complete and particular databases of IAMonDo.

Database	*Dataset*	*#Pages*	*#Strokes*	*#T*	*#N*	*%T:%N*
IAMonDo	*Set 1*	203	70,976	58,206	12,770	82.01:17.99
	Set 2	200	72,374	57,851	14,523	79.93:20.07
	Set 3	200	68,726	57,455	11,271	83.60:16.40
	Set 4	203	70,927	57,634	13,293	81.26:18.74
Tables_IAM	*Training*	184	13,010	12,221	789	93.94:6.06
	Validation	92	6,546	6,197	349	94.67:5.33
	Testing	102	6,857	6,483	374	94.55:5.45
Diagrams_IAM	*Training*	386	27,090	11,682	15,408	43.12:56.88
	Validation	191	11,812	4,912	6,900	41.58:58.42
	Testing	193	11,698	4,890	6,808	41.80:58.20

Table 6
Statistics for datasets in the FC and FA databases.

Database	*Dataset*	*#Diagrams*	*#Strokes*	*#T*	*#N*	*%T:%N*
FC	*Training*	187	17,752	11,074	6,678	62.38:37.62
	Validation	61	5,607	3,481	2,126	62.08:37.92
	Testing	171	15,696	9,614	6,082	61.25:38.75
FA	*Training*	132	6,792	3,261	3,531	48.01:51.99
	Validation	84	4,059	2,189	1,870	53.93:46.07
	Testing	84	4,125	2,077	2,048	50.35:49.65

5.2. Feature normalization

The distribution of each feature extracted from a single stroke and stroke pair is distorted from a normal distribution. Hence, we use a power transformation to make the density function closer to a Gaussian. We set the power to 0.5 to transform each feature value. Furthermore, in order to standardize the feature values, we normalize the values of each feature based on the mean μ_f and standard deviation σ_f of that feature. The normalized feature value is then calculated as

$$v_f' = \frac{v_f - \mu_f}{\sigma_f} \tag{14}$$

The mean and standard deviation are calculated for each feature over the training samples. These values are stored and used for normalizing the training, validation, and testing samples.

5.3. Parameter settings

In the feature extraction of single strokes and pairs of strokes, two strokes are considered as temporal and spatial neighbors if the temporal and spatial distances between them are less than thresholds of 3.5 s and 4 pixels, respectively. The FC and FA databases have no information about the stroke writing time, so we assumed a velocity of 1 pixel/s in order to obtain the temporal distance from the trajectory length.

In the training of the single stroke and stroke pair classifiers, we used four-layered networks with two fully connected recurrent

hidden layers. The numbers of units/blocks k and l in these two layers are set to 10 and 30. The learning rate was 10^{-5} on BRNNs and 10^{-4} on BLSTMs. The momentum rate was 0.9 for all trainings.

The above parameters were optimized on Kondate. They also work well on the other databases so that we use them without any attempt to optimize them again.

For testing, each experiment was repeated 20 times to make the results stable and independent of the random weight initialization. Our experiments on BRNN and BLSTM were performed with the open source software library RNNLIB [34].

5.4. Results of experiments

This section presents the results of the experiments on the four databases: Kondate, IAMonDo, FC, and FA. We performed four evaluations of the global context based single stroke classifier and stroke pair classifiers, local context integrated classifiers, and combined classifiers. Next, we did cross-validation experiments on the two main databases: Kondate and IAMonDo. After that, we experimented on Tables_IAM, Diagrams_IAM, FC, and FA. Next, we examined the computational complexity of the method. At last, we analyzed some qualitative results. The experiments employed two single stroke classifiers (a 16-feature classifier and a 26-feature classifier) and three stroke pair classifiers (a 4-feature ternary classifier and 19-feature ternary and quaternary classifiers). We refer to these classifiers as GSC16, GSC26, GPC4, GPC19 and GPC19Q. Moreover, we used two types of neural network architectures for each classifier: BRNN and BLSTM. The 16-feature single stroke classifier modeled by BRNN and the 4-feature stroke pair classifier modeled by BLSTM are called here GSC16_RNN and GPC4_LSTM, respectively. The other classifiers are named similarly.

In experiments, when we compare systems and methods, we use their means to evaluate the difference. Although the large size of the testing datasets and especially the large number of strokes would enable us to make reliable comparisons, we also perform t-test to evaluate the significance of the difference statistically for important comparisons. When we compare two of our systems, we can obtain classification result for every stroke and every document. We apply the classification to each document page and validate the difference employing a paired t-test. On the other hands, when we compare our system with another system in previous works, often the mean of the classification rate is reported but the variance is not reported so that validation is generally difficult. Owing to the special case, however, we can perform an unpaired t-test with mean and variance values if we apply the classification to each stroke. In previous works, classification rates are calculated on the whole testing dataset and the measurement is the classification result of all the single strokes. In this case, we can obtain the variance σ^2 value from the mean μ value reported in the previous works as follows:

$$\begin{aligned}\sigma^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - 2\frac{1}{n}\sum_{i=1}^{n}x_i\mu + \frac{1}{n}n\mu^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}x_i - 2\mu\mu + \mu^2 \quad \text{when } x_i = 0 \text{ or } 1 \\ &= \mu - \mu^2 \end{aligned} \tag{15}$$

where n is the number of strokes and x_i is the classification result of every stroke (1 for true and 0 for false).

Table 7
Performance of global context based classifiers GSCs.

Database	*Classifier*	*Overall* (%)	*Text* (%)	*Non-Text* (%)	*Time* (s)
Kondate	*GSC16_RNN*	96.67 (95.35–97.37)	98.67	84.99	0.39
	GSC16_LSTM	97.22 (96.59–97.56)	98.81	87.88	1.56
	GSC26_RNN	97.98 (97.39–98.46)	99.08	91.47	0.42
	GSC26_LSTM	**98.13** (97.50–**98.69**)	**99.11**	**92.41**	1.66
IAMonDo	*GSC16_RNN*	95.94 (94.98–96.78)	97.97	87.09	0.37
	GSC16_LSTM	96.83 (96.57–97.17)	98.45	89.80	1.52
	GSC26_RNN	97.32 (96.86–97.65)	98.49	92.26	0.40
	GSC26_LSTM	**97.56** (97.30–**97.81**)	**98.58**	**93.15**	1.62

5.4.1. Evaluation of global context based classifiers

In the global context model, which is realized by single stroke classifiers, strokes can be classified with a binary classifier trained by BRNN or BLSTM. Table 7 shows the overall, text and non-text classification accuracies for these single stroke classifiers on Kondate and IAMonDo. The computational time is the total time for classification and does not include the time taken for feature extraction on the entire testing dataset, i.e., 359 pages in Kondate and 203 pages in IAMonDo.

From the table, we can see that the BLSTM classifiers behave more stably than the BRNN classifiers. The differences between the maximum and minimum accuracies of the BLSTM classifiers are smaller than those of the BRNN classifiers by 1 point. For instance, the GSC16 classifiers for IAMonDo differ from the BLSTM classifier by 0.6 points (96.57–97.17%), and from BRNNs by 1.8 points (94.98–96.78%). The BLSTM classifiers are more accurate although they are about four times slower than the BRNN classifiers. Accuracy improves if a larger number of features are extracted. The BLSTM classifiers with 26 features were the most accurate on both the databases. On the Kondate database, their mean accuracy (98.13%) and even their minimum accuracy (97.50%) were better than the MRF's result (96.61%) in [4]. As for the IAMonDo database, the mean accuracy (97.56%) and the minimum accuracy (97.30%) were better than those of the CRF (96.66%) in [5] and those of the BLSTM using local features (97.01%) in [8]. Moreover, they are slightly higher than the accuracy of the state-of-the-art method (97.23%) in [7].

5.4.2. Evaluation of stroke pair classifiers

In order to classify strokes in the local context model, the stroke pair classifiers were trained using BRNN or BLSTM. In order to classify pairs of strokes into three or four categories: text:text (TT), text:non-text (TN) and/or non-text:text (NT), and non-text:non-text (NN). The performance of these classifiers are listed in Table 8.

The same as with the single stroke classifiers, the BLSTM classifiers were more accurate than the BRNN classifiers, but about four times slower. Although only four simple features were used, the GPC4 classifiers were comparable to the GPC19 classifiers on Kondate. With 19 features including symmetrical descriptions, the quaternary classifiers were slightly more accurate than the ternary classifiers.

It is noteworthy that the classification accuracy of the text:non-text or/and non-text:text pairs was very low on the IAMonDo database containing many short strokes in the graphics and diagrams. Since the features are related to the size of and the distance between strokes, the text:non-text pairs are likely misclassified as text:text pairs.

5.4.3. Evaluation of local context integrated classifiers

To evaluate the classifiers in the local context model, we selected the BLSTM classifiers of the single stroke classifications using 16 and 26 unary features and the stroke pair classifications

Table 8
Performance of stroke pair classifiers GPCs.

Database	*Classifier*	*Overall* (%)	*TT* (%)	*TN* (%)	*NT* (%)	*NN* (%)	*Time* (s)
Kondate	*GPC4_RNN*	95.51 (93.54–96.71)	98.26	67.42		86.83	0.39
	GPC4_LSTM	96.75 (95.59–97.32)	98.67	80.21		89.61	1.54
	GPC19_RNN	96.26 (95.25–97.21)	98.86	68.56		88.46	0.42
	GPC19_LSTM	**97.07** (96.24–**97.65**)	98.99	79.22		90.36	1.58
	GPC19Q_RNN	96.70 (95.74–97.48)	98.86	77.51	71.43	90.00	0.43
	GPC19Q_LSTM	**97.17** (96.42–**97.61**)	99.02	81.84	76.11	91.10	1.60
IAMonDo	*GPC4_RNN*	92.86 (90.98–94.71)	97.76	25.78		83.32	0.37
	GPC4_LSTM	94.48 (94.02–94.96)	97.73	47.82		88.58	1.47
	GPC19_RNN	95.22 (93.97–96.01)	97.84	51.26		91.72	0.40
	GPC19_LSTM	**95.84** (95.42–**96.23**)	98.12	61.29		92.05	1.54
	GPC19Q_RNN	95.49 (92.95–96.20)	97.91	58.43	52.64	92.12	0.41
	GPC19Q_LSTM	**95.99** (95.59–**96.35**)	98.19	64.44	60.58	92.40	1.56

Table 9
Accuracies (%) of local context integrated classifiers LCCs compared with global context based classifiers GSCs.

Database	*GSC classifier*	*GSC accuracy* (%)	*GPC classifier*	*GPC accuracy* (%)	*LCC accuracy* (%)		
					PCC	*SCC*	*BCC*
Kondate	*GSC16_LSTM*	*97.56*	*GPC4_LSTM*	97.32	97.72	97.73	**98.26**
			GPC19_LSTM	97.65	97.94	97.91	**98.44**
			GPC19Q_LSTM	97.61	98.48	98.45	**98.48**
	GSC26_LSTM	*98.69*	*GPC4_LSTM*	97.32	98.11	98.11	**98.41**
			GPC19_LSTM	97.65	98.25	98.23	**98.55**
			GPC19Q_LSTM	97.61	**98.65**	98.64	98.63
IAMonDo	*GSC16_LSTM*	*97.17*	*GPC4_LSTM*	94.96	96.71	96.72	**97.04**
			GPC19_LSTM	96.23	97.12	97.04	**97.54**
			GPC19Q_LSTM	96.35	97.78	97.74	**97.81**
	GSC26_LSTM	*97.81*	*GPC4_LSTM*	94.96	96.99	96.98	**97.27**
			GPC19_LSTM	96.23	97.38	97.31	**97.67**
			GPC19Q_LSTM	96.35	**97.96**	97.89	97.93

using 4 and 19 binary features that had the maximum accuracies. There are three local context models for the stroke pair classifier: preceding, succeeding, and bidirectional. Hence, we have three local context integrated classifiers: preceding-model based classifier (PCC), succeeding-model based classifier (SCC) and bidirectional-model based classifier (BCC) for each pair of unary features and binary features. Table 9 shows the classification results obtained by these classifiers.

From these results, it is clear that the LCC classifier using both preceding and succeeding information in the bidirectional model (BCC) was more accurate than the one using only the preceding information (PCC) or succeeding information (SCC) except for quaternary classifiers using 26 unary features and 19 binary features, in which case the classifiers using only preceding information were a little more accurate than the others. The accuracies of the bidirectional-model classifiers integrating local contexts with the 16-feature global context based classifiers were better than the original GSC classifiers alone in most cases. In particular, the PCC local context integrated classifiers had accuracies 98.48% and 97.81%, which were significantly higher than those of the original global context based classifiers (97.56% and 97.17%, respectively) with both $p < 0.0001$ by paired t-tests on 359 testing pages of Kondate and 203 testing pages of IAMonDo. This is to certify that the pairwise stroke features in LCC classifiers have supplemented the single stroke features in GSC classifiers. The highest accuracies of the LCC classifiers on Kondate and IAMonDo were 98.65% and 97.96%, respectively.

Table 10
Accuracies (%) of combined classifiers compared with global context based and local context integrated classifiers.

Database	*GSC type*	*GSC Acc* (%)	*LCC Acc* (%)	***Combination rule*** (*without weighting parameters*)			
				SUM	*PROD*	*MAX*	*MIN*
Kondate	*16_LSTM*	*97.56*	*98.48*	98.41	**98.61**	98.32	**98.59**
	26_LSTM	*98.69*	*98.63*	98.97	**99.04**	98.91	**99.00**
IAMonDo	*16_LSTM*	*97.17*	*97.81*	97.94	**98.02**	97.82	**98.02**
	26_LSTM	*97.81*	*97.93*	98.21	**98.27**	98.09	**98.30**

5.4.4. Evaluation of combined classifiers

In this experiment, we measured the effect of combining the global context based classifier GSC and the local context integrated classifier LCC. We combined the global context based classifier GSC16_LSTM or GSC26_LSTM with a bidirectional local context integrated classifier integrating the stroke pair classifier GPC19Q_LSTM with the single stroke classifier GSC26_LSTM (denoted by BCC26_19Q_LSTM). The experiment on the four combination rules was performed without weighting parameters in (12) or (13). Table 10 shows the results.

On both databases, most of combinations were effective since the accuracies of the combined classifiers, those using the product and

Table 11
Parameter values for weighted combination.

Combination rule	Linear function $\lambda_{1,1}, \lambda_{1,2}, \lambda_{2,1}, \lambda_{2,2}$	Exponential function λ_1, λ_2
SUM	0.205, 0.220, 0.285, 0.500	0.015, 0.030
PROD	0.910, 0.075, 0.290, 0.001	0.235, 0.480
MAX	0.725, 0.155, 0.070, 0.780	0.590, 0.010
MIN	0.600, 0.575, 0.235, 0.575	0.325, 0.650

Table 12
Accuracies (%) of combined classifiers with and without weighting parameters.

Combination rule	Without parameters	With parameters of linear function	With parameters of exponential function
SUM	*98.21*	**98.30**	98.17
PROD	*98.27*	**98.30**	98.23
MAX	*98.09*	98.05	**98.20**
MIN	***98.30***	98.26	**98.30**

min rules were better than those of the global context based classifiers or the local context integrated classifiers alone. The product and min rules were equally good and more effective than the sum and max rules. The combined classifiers of GSC26 and BCC26_19Q trained using BLSTM were the most accurate on both databases. On the Kondate database, PROD(GSC26_LSTM, BCC26_19Q_LSTM) had a correct classification rate of 99.04%, which is higher than the 96.61% in the previous study [4]. On the IAMonDo database, MIN (GSC26_LSTM, BCC26_19Q_LSTM) yielded a correct classification rate of 98.30%, which is also higher than what have been reported in the literature, i.e., 97.01% reported by Indermuhle et al. [8] and 97.23% reported by Delaye et al. [7]. When we performed unpaired *t*-tests for significance to compare our best systems with systems in previous works, the classification results of 71,846 strokes on Kondate and 70,927 strokes on IAMonDo was used. The two-tailed *p* values in all *t*-tests are less than 0.0001 so that the differences are considered extremely significant.

To evaluate the effect of the weighting parameters, we conducted an experiment on the GSC26_LSTM and BCC26_19Q_LSTM combination using the IAMonDo database. The weighting parameters for the four combination rules were optimized on the validation set using the genetic algorithm. The parameters for the linear and exponential functions of the text and non-text probabilities retrieved from the global context based and the local context integrated classifiers are shown in Table 11. The classification results of the combined classifiers are shown in Table 12. We can see that the weighting parameters of the combination had a little effect. It seems that our method reached the limit of its capability.

5.4.5. Cross-validation experiments and their results

In order to reduce the risk of a biased dataset division, we performed four-fold cross validation on IAMonDo and Kondate. On IAMonDo, we conducted three experiments on three datasets because the first dataset already had been evaluated in the holdout validation. On Kondate, we conducted four experiments on four datasets. We employed the combined classifier PROD(GSC26_LSTM, BCC26_19Q_LSTM) in all experiments. Since the graphical non-text strokes account for less than 20% of the total strokes in these databases, they have less influence on the overall accuracy. For the extraction of structured graphical elements such as tables, flow charts, finite automata, etc., however, the accuracy of non-text detection is important. Hence, in the experiments, we calculated the precision, recall and *f*-measure of the non-text detection. Table 13 shows the results.

Table 13
Accuracies (%) of combined classifiers in four-cross validations.

Database	Dataset	Overall (%)	Text (%)	Non-text		
				Precision (%)	Recall (%)	f-Measure (%)
Kondate	*Fold 1*	*99.34*	99.67	97.25	97.97	97.61
	Fold 2	*99.03*	99.75	95.22	98.60	96.88
	Fold 3	*98.99*	99.75	95.01	98.62	96.78
	Fold 4	*99.11*	99.6	96.32	97.74	97.02
	Average	***99.12***	**99.69**	**95.95**	**98.23**	**97.07**
IAMonDo	*Fold 1*	*98.27*	98.98	95.19	95.57	95.38
	Fold 2	*97.80*	99.07	91.98	95.61	93.76
	Fold 3	*98.23*	99.18	94.43	96.66	95.53
	Fold 4	*98.24*	98.98	94.46	94.80	94.63
	Average	***98.14***	**99.05**	**94.02**	**95.66**	**94.83**

Table 14
Accuracies (%) on Tables_IAM, Diagrams_IAM, FC and FA databases.

Database	Overall (%)	Text (%)	Non-text		
			Precision (%)	Recall (%)	f-Measure (%)
Tables_IAM	**99.88**	99.97	98.40	99.46	98.93
Diagrams_IAM	**93.36**	92.07	94.29	94.30	94.29
FC	**98.55**	98.66	98.37	97.89	98.13
FA	**99.61**	99.52	99.71	99.51	99.61

Our method achieved overall classification rates of 99.12% and 98.14% on average on Kondate and IAMonDo, respectively. The text detection accuracies were all over 99%, and the non-text detection accuracies were all over 94%. Thus, our method was better at detecting graphics than hierarchical random fields in [14] (precision around 90% and recall of only 80–85%) in the IAMonDo database.

5.4.6. Experiments on other databases and their results

We trained and evaluated our method on Tables_IAM and Diagrams_IAM. We also evaluated it on specific domains: flow charts (FC database) and finite automata (FA database). The classification rates together with the precision, recall and *f*-measure of the graphics detection on these datasets are listed in Table 14. With Tables_IAM, since the text strokes account for over 95% and the non-text strokes are mostly straight lines, it is easy for our method to separate them. The accuracy was 99.88% even when we used only the global context based classifier GSC16_RNN. As with Tables_IAM, the FC and FA databases were also easy to learn. They contain non-text strokes such as curve lines, arrows, circles, ellipses, rectangles, etc. which could be easily distinguished from text strokes. Hence, we obtained very accurate results for these two databases by using the global context based classifiers. Moreover, we achieved overall classification rates of 98.55% and 99.61% on FC and FA, respectively, when we used the combined classifier PROD(GSC26_LSTM, BCC26_19Q_LSTM). Not only the overall rates, but also the text and non-text classification rates were high (over 98%). The accuracies were higher than those of for overall 93.06%, for text 91.25%, and for non-text 94.87% reported in [33]. Although Tables_IAM, FC and FA were easy to learn, Diagrams_IAM was hard to learn because non-text strokes make up nearly 60% of it and

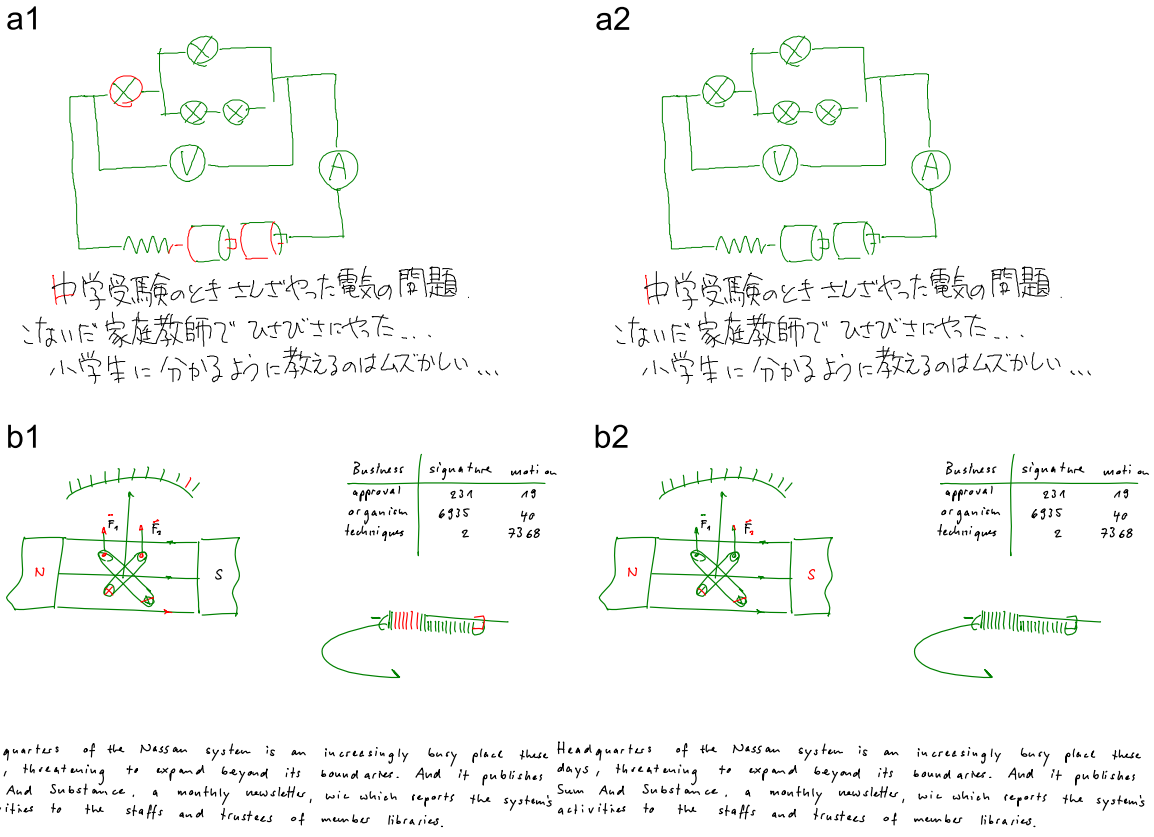

Fig. 4. Examples of improvements to text/non-text classification. Pages a(1) and a(2) are from Kondate, while b(1) and b(2) are from IAMonDo. a(1) and b(1) are results classified by GSC26_LSTM while a(2) and b(2) are results classified by MIN(GSC26_LSTM, BCC26_19Q_LSTM). Text strokes are shown in black, non-text strokes are shown in green, and misclassified strokes are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

they have large variations. In this case, the classification rate was only 93.36%.

5.4.7. Computational complexity

We performed the experiments on a computer with an Intel® Core™ i7-4770 CPU (3.40 GHz). The computational time was measured on the entire testing dataset of the IAMonDo database (203 documents). The extraction of 16 and 26 unary features took 0.28 and 2.82 s, respectively. The extraction of 4 and 19 binary features cost 0.32 and 0.38 s, respectively. The costs of the global context based and the local context integrated classifiers using BRNN and BLSTM were 0.89 and 3.23 s and the combination took just a few milliseconds. In sum, the total time for classifying digital ink strokes in the 203 documents was 6.43 s, even when the most time-consuming classifiers such as the combined classifier PROD (GSC26_LSTM, BCC26_19Q_LSTM) were used. Hence, the time required for processing each document from the test dataset was about 38 ms on average, which would not appear as a noticeable delay in showing the classification. It is faster than the time of 1.53 s reported by Delaye et al. [7] for a classification done on a similar CPU. The total time for classifying digital ink strokes in the 359 documents of the Kondate database was 9.28 s, even when the most time-consuming classifiers, i.e., a combined classifiers, were used.

5.4.8. Qualitative analyses

Fig. 4 presents examples of text/non-text classification results on the Kondate and IAMonDo databases. They were obtained with the 26-feature global context based classifier GSC26_LSTM and the context combined classifier MIN(GSC26_LSTM, BCC26_19Q_LSTM), which is the classifier combining GSC26_LSTM with the bidirectional local context integrated classifier with the min-rule mentioned above. It shows that the combination of the local context with the global context has reduced errors.

Although accuracies close to 99% have been achieved for text/non-text classification, some problems remain to be solved. In particular, we still face problems stemming from 1) a lack of contextual information when isolated symbols are written separately, as shown in Figs. 5(a); and 2) short non-text strokes and long text strokes as shown in Fig. 5(b). These aspects make text/non-text classification of free-hand diagrams difficult. We think that these errors can be reduced with post-processings using heuristic and handwriting recognition. If non-text strokes are inside a text line, they can be reclassified as text strokes. If the recognition scores of a clique of text strokes are small, they can be converted into non-text strokes. Handwriting recognition seems to be the ultimate goal with which we could realize segmentation or classification by recognition. Handwritten graphics recognition is, however, still a difficult problem since casually hand-drawn graphics are often outside of the domain.

6. Conclusion

This paper described a novel method for text/non-text classification of online handwritten documents that is based on a combination of global and local contexts with recurrent neural networks. A bidirectional network architecture is used to access to the complete global context of stroke sequences. Moreover, a simple but effective model is used for the local temporal context of adjacent strokes. Global and local contexts complement each other so that their combination improves the text/non-text classification of strokes. Our method achieved classification rates of

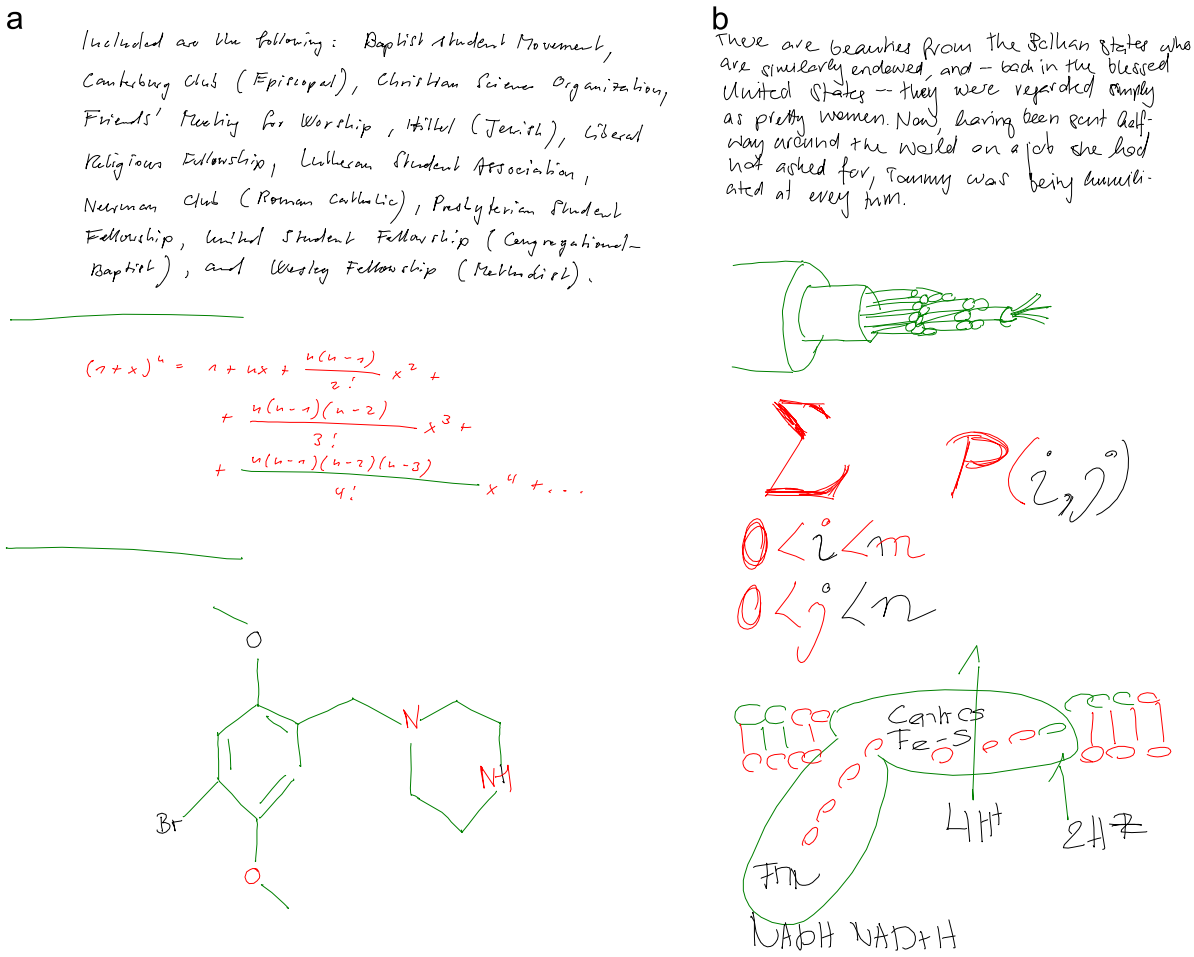

Fig. 5. Examples of errors in text/non-text classification. Text strokes are shown in black, non-text strokes are shown in green, and misclassified strokes are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

99.04% on the Kondate database of Japanese digital ink documents and 98.30% on the IAMonDo database of English digital ink documents. It outperformed other state-of-the-art methods from the literature.

Our method was extremely accurate (about 99%) on the Kondate database, but faced challenges on the IAMonDo database, especially on the extracted sub-database, Diagrams_IAM. In this database, the number of non-text strokes is larger than the number of text strokes. We plan to investigate other global features to improve the accuracies of the single stroke classifiers and stroke pair classifiers, as well as other frameworks for combining global and local contexts to enhance the overall performance. The pairwise stroke features can be integrated directly. For example, it can be used as weights of the connections between two strokes in a unique network. We also plan to investigate the range of global context. We have considered the global context in the range of a document page since each page stores handwriting on some topic by a single writer. If a page includes various objects, or if a series of pages are homogeneous, however, different ranges of global context may produce better performance. Furthermore, we will try to use local features and the strategy of mode detection to make the task of text/non-text classification run in real-time.

Acknowledgments

This research was supported by Grants-in-Aid for Scientific Research from the Japan Society for the Promotion of Science organization (Contract numbers: (B) 24300095 and (S) 25220401). The authors would like to thank Mr. Cuong Tuan Nguyen and Mr. Anh Duc Le for giving helpful comments and suggestions.

References

[1] A.K. Jain, A.M..Namboodiri, J. Subrahmonia, Structure in online documents, in: Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR2001), Seattle, WA, USA, 2001, 844–848.
[2] A. Delaye, C.-L. Liu, Multi-class segmentation of free-form online documents with tree conditional random fields, Int. J. Doc. Anal. Recognit. 17 (4) (2014) 313–329.
[3] C.M. Bishop, M. Svensen, G.E. Hinton, Distinguishing text from graphics in online handwritten ink, in: Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR2004), Washington, DC, USA, 2004, 142–147.
[4] X.D. Zhou, C.-L. Liu, Text/non-text ink stroke classification in Japanese handwriting based on Markov random fields, in: Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR2007), Curitiba, Brazil, 2007, 377–381.
[5] A. Delaye, C.-L. Liu, Text/non-text classification in online handwritten documents with conditional random fields, in: Proceedings of the Chinese Conference on Pattern Recognition (CCPR2012), Beijing, China, 2012, 514–521.
[6] A. Delaye, C.-L. Liu, Context modeling for text/non-text separation in free-form online handwritten documents, in: Proceedings of the 20th International Conference on Document Recognition and Retrieval (DDR2013), SPIE 8658, 2013.
[7] A. Delaye, C.-L. Liu, Contextual text/non-text stroke classification in online handwritten notes with conditional random fields, Pattern Recognit. 47 (3) (2014) 959–968.
[8] E. Indermühle, V. Frinken, H. Bunke, Mode detection in online handwritten documents using BLSTM neural networks, in: Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR2012), Bari, Italy, 2012, 302–307.

[9] K. Mochida, M. Nakagawa, Separating figures, mathematical formulas and Japanese text from free handwriting in mixed on-line documents, Int. J. Pattern Recognit. Artif. Intell. 18 (7) (2004) 1173–1187.
[10] E. Indermühle, M. Liwicki, H. Bunke, IAMonDo-database: an online handwritten document database with non-uniform contents, in: Proceedings of the 9th International Workshop on Document Analysis Systems (DAS2010), Boston, USA, 2010, 97–104.
[11] A.M. Awal, G. Feng, H. Mouchere, C. Viard-Gaudin, First experiments on a new online handwritten flowchart database, in: Proceedings of the Document Recognition and Retrieval XVIII, 2011, 1–10.
[12] M. Bresler, T.V. Phan, D. Prusa, M. Nakagawa, V. Hlavac, Recognition system for on-line sketched diagrams, in: Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR2014), Crete, Greece, 2014, 563–568.
[13] K. Machii, H. Fukushima, M. Nakagawa, On-line text/drawings segmentation of handwritten patterns, in: Proceedings of the 2nd International Conference on Document Analysis and Recognition (ICDAR1993), Tsukuba, Japan, 1993, 710–713.
[14] A. Delaye, C.-L. Liu, Graphics extraction from heterogeneous online documents with hierarchical random fields, in: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR2013), Washington, DC, USA, 2013, 1007–1011.
[15] A. Bhat, T. Hammond, Using entropy to distinguish shape versus text in hand-drawn diagrams, in: Proceedings of the 21st International Joint Conference on Artificial Intelligent (IJCAI2009), California, USA, 2009, 1395–1400.
[16] R. Patel, B. Plimmer, J. Grundy, R. Ihaka, Ink features for diagram recognition, in: Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM'07), New York, NY, USA, 2007, 131–138.
[17] R. Blagojevic, B. Plimmer, J. Grundy, Y. Wang, Using data mining for digital ink recognition: dividing text and shapes in sketched diagrams, Comput. Graph. 35 (5) (2011) 976–991.
[18] E. Lank, J.S. Thorley, S.J-S. Chen, An interactive system for recognizing hand drawn UML diagrams, in: Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON'00), Ontario, Canada, IBM Press, 2000, p. 7.
[19] T. Hammond, D. Randall, Tahuti: a geometrical sketch recognition system for UML class diagrams, in: ACM SIGGRAPH 2006 Courses, 2006, p. 25.
[20] B. Plimmer, I. Freeman, A. Toolkit, Approach to sketched diagram recognition, in: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1, British Computer Society, 2007, 205–213.
[21] R.C. Zeleznik, A. Bragdon, C.-C. Liu, A. Forsberg, Lineogrammer: creating diagrams by drawing, in: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, 2008, 161–170.
[22] M. Liwicki, A. Weber, A. Dengel, Online mode detection for pen-enabled multi-touch interfaces, in: Proceedings of the 15th Conference of the International Graphonomics Society (IGS2011), Cancun, Mexico, 2011, 78–81.
[23] M. Weber, M. Liwicki, Y.T. Schelske, C. Schoelzel, F. Strauß, A. Dengel, MCS for online mode detection: evaluation on pen-enabled multi-touch interfaces, in: Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR2011), Beijing, China, 2011, 957–961.
[24] S. Otte, D. Krechel, M. Liwicki, A. Dengel, Local feature based online mode detection with recurrent neural networks, in: Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR2012), Bari, Italy, 2012, 533–537.
[25] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Netw. 18 (6) (2005) 602–610.
[26] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber, A. Novel, Connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–869.
[27] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[28] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition (1), MIT Press, Cambridge, MA, USA, 1985, 318–362.
[29] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[30] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[31] A. Lemaitre, H. Mouchere, J. Camillerapp, B. Coüasnon, Interest of syntactic knowledge for online flowchart recognition, in: Y.B. Kwon, J.M. Ogier (Eds.), Graphics Recognition. New Trends and Challenges, Springer, Berlin Heidelberg, 2013, 89–98.
[32] C. Carton, A. Lemaitre, B. Couasnon, Fusion of statistical and structural information for flowchart recognition, in: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR2013), Washington, DC, USA, 2013, 1210–1214.
[33] M. Bresler, Text/non-text classification of strokes using the composite descriptor, in: Proceedings of the 17th International Student Conference on Electrical Engineering (POSTER2013), Prague, Czech, 2013, 1–5.
[34] A. Graves, RNNLIB: A Recurrent Neural Network Library for Sequence Learning Problems, ⟨http://sourceforge.net/projects/rnnl/⟩, 2013.

Truyen Van Phan was born on 13 February 1984 in Vietnam. He received his B.Sc. degree in Computer Science and Engineering from Ho Chi Minh City University of Technology (HCMUT), Vietnam in 2007. In 2012, he received his M.Sc. degree in Computer and Information Sciences from the Tokyo University of Agriculture and Technology (TUAT), Japan. He received his Ph.D. degree in Department of Electronic and Information Engineering from TUAT in March 2015. He is working on ink document analysis, off-line handwriting character recognition, and historical document analysis, recognition, and retrieval.

Masaki Nakagawa was born on 31 October 1954 in Japan. He received his B.Sc. and M.Sc. degrees from the University of Tokyo in 1977 and 1979, respectively. During the academic year 1977/78, he enrolled in the Computer Science course at Essex University in England and received the M.Sc. with distinction in Computer Studies in July 1979. He received his Ph.D. in Information Science from the University of Tokyo in December 1988. He has been working at Tokyo University of Agriculture and Technology since April 1979. He is currently a Professor of Media Interaction in the Department of Computer and Information Sciences.