Dynamic and Contextual Information in HMM Modeling for Handwritten Word Recognition

Anne-Laure Bianne-Bernard, Farès Menasri, Rami Al-Hajj Mohamad, Chafic Mokbel, *Senior Member*, *IEEE*, Christopher Kermorvant, and Laurence Likforman-Sulem, *Senior Member*, *IEEE*

Abstract—This study aims at building an efficient word recognition system resulting from the combination of three handwriting recognizers. The main component of this combined system is an HMM-based recognizer which considers dynamic and contextual information for a better modeling of writing units. For modeling the contextual units, a state-tying process based on decision tree clustering is introduced. Decision trees are built according to a set of expert-based questions on how characters are written. Questions are divided into global questions, yielding larger clusters, and precise questions, yielding smaller ones. Such clustering enables us to reduce the total number of models and Gaussians densities by 10. We then apply this modeling to the recognition of handwritten words. Experiments are conducted on three publicly available databases based on Latin or Arabic languages: Rimes, IAM, and OpenHart. The results obtained show that contextual information embedded with dynamic modeling significantly improves recognition.

Index Terms—Latin and Arabic handwriting recognition, context-dependent HMMs, neural-network combination.

1 Introduction

There is a growing interest for handwriting recognition since handwriting is a widespread means of communication and can be found in both historical and modern documents. Among popular applications of handwriting recognition are bank check processing, mailed envelopes reading, and handwritten text recognition in documents and videos, for which different systems have been successfully developed [1], [2], [3], [4], [5]. The scope of recognition can be extended to reading letters sent to companies or public offices since there is a demand to sort, search, and automatically answer mails based on document content. Detecting a restricted set of handwritten keywords, such as in [6], is useful for sorting handwritten mails. However, a full transcription of the majority of words present in mail would be much more powerful. This transcription, provided in textual format such as ASCII, enables textual queries adapted to any user need. However, for reading most words of a handwritten letter, recognition systems must cope with the inherent handwriting variability and deal with a large number of word and character models. Stochastic modeling such as Hidden Markov Models (HMMs) has proven to be effective for modeling handwriting. In addition, word recognition with HMMs can be easily extended to line and paragraph recognition by introducing language models [7].

HMMs are effective for modeling unconstrained words since they can cope with nonlinear distortions. HMM systems can be divided into two types, depending on their strategy: holistic or analytical. The holistic strategy, restricted to small lexica, considers word images as a whole and does not attempt to segment words into characters or smaller units. In contrast, the analytical strategy consists of modeling words by the concatenation of compound character HMMs. The character-based representation of words which qualifies the analytical strategy is convenient for enlarging the vocabulary: Word models are built from the concatenation of trained character models. The advantage is that words which were not present in training data can still be modeled without requiring word images as soon as the characters composing these words are trained.

Representing words by characters may or may not require the segmentation of the words into characters, previous to recognition. There are analytical HMM systems which presegment words into characters or subunits based on maxima or minima points of the contours or maximum curvature points [8]. In contrast, the so-called Sliding Window Systems perform implicit segmentation jointly with recognition [6], [9], [10], [11]. Such systems can be easily applied and achieve state-of-the-art performance. A comprehensive survey on HMM systems for handwriting recognition along with their performance can be found in [12].

In the following, HMM-based classifiers are used for the recognition of handwritten words. Words are given isolated (see Fig. 1) thanks to a word segmentation scheme

- *A.-L. Bianne-Bernard is with A2iA SA, 40 bis rue Fabert, Paris 75007, France, and Télécom ParisTech, 46 rue Barrault, Paris 75013, France. E-mail: alb@a2ia.com.*
- *F. Menasri and C. Kermorvant are with A2iA SA, 40 bis rue Fabert, Paris 75007, France. E-mail: {fm, ck}@a2ia.com.*
- *R. Al-Hajj Mohamad is with the Lebanese International University, LIU Campus, PO Box 146404, Mazraa, Beirut, Lebanon. E-mail: rami.elhajj@liu.edu.lb.*
- *C. Mokbel is with the University of Balamand, PO Box 100, Tripoli, Lebanon. E-mail: chafic.mokbel@balamand.edu.lb.*
- *L. Likforman-Sulem is with Télécom ParisTech/LTCI, laboratoire TSI, 46 rue Barrault, Paris 75013, France. E-mail: likforman@telecom-paristech.fr.*

Manuscript received 4 May 2010; revised 16 Nov. 2010; accepted 7 Jan. 2011; published online 28 Jan. 2011.
Recommended for acceptance by E. Saund.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2010-05-0346.
Digital Object Identifier no. 10.1109/TPAMI.2011.22.

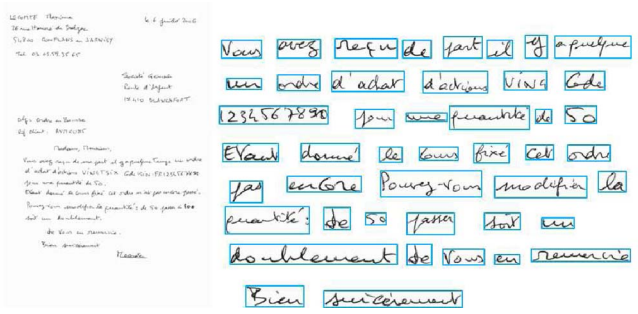


Fig. 1. Sample handwritten mail of the Rimes database and extracted words.

preperformed by the database holders. In comparison to a line-based approach, segmentation of a text into words has a lower computational cost. It has been shown that combination of different systems is complementary and more efficient. In this work, three HMM-based classifiers are combined. The three systems differ from their segmentation strategy and from the type of character modeling, which make them complementary. In order to build different systems, we have considered a first system that relates the character states to a clustering of the observed frames extracted from a sliding window. In contrast to this system, the second system refines character states according to small character variations. Those variations are due to the hand movement (*ductus*) when forming basic writing units which is under the control of the eye and the brain [13]. In [14], it is shown how such movement can be artificially generated according to delta lognormal laws. The basic units may be strokes, letters, parts of words, or whole words. Thus, the context of a character within a writing unit has great influence on its shape (see Fig. 2). For a given character, we have considered the influence of neighboring characters on its shape. We have used our knowledge of ligatures and the shapes of leftmost or rightmost parts of neighboring characters to assist the modeling of individual letters. Such knowledge has been embedded in decision trees used by our state clustering process (Section 4) and results in more accurate character models. Then, we have considered a third system where characters' states correspond to subunits such as graphemes.

Within the HMM framework, there are several ways to introduce context when modeling individual letters. One method consists of extending the size of sliding windows to capture the pixels or the features around the current character. Another method consists of adding features from neighboring windows to the current window. The first method is limited to capturing a close neighborhood (4 to 8 pixels) and the resulting observation probability densities may have large variance since the neighborhood is highly varying. Such variance is not desirable for an accurate character modeling. The second method is also limited to a close neighborhood since the larger the neighborhood, the larger the dimension of the feature vector extracted. This is also not desirable since it is well known that high-dimensional vectors need a large amount of data to estimate their distribution (the curse of dimensionality). An alternative method is to build different character models according to different contexts, i.e., specifying the

Fig. 2. Influence of context for handwriting on two words written by the same person ("Monsieur," "distinguée").

models of a character depending on its context. This approach is known as the context-dependent approach in the domain of speech recognition and has been applied to printed character recognition [15]. To our knowledge, only a few works deal with contextual modeling in handwriting recognition [16], [17], [18].

Contextual approaches lead to excessive growth in the number of models since one model is needed for each pair of adjacent characters. Parameter estimation may be unreliable since, for practical applications, a restricted set of training data is generally available. It is thus desirable to reduce the number of models and model parameters while preserving model refinement. Hence, model sharing and parameter tying are necessary to reduce the number of parameters. Schussler and Niemann [16] describe a context-dependent system using HMMs, where all subword units (from monographs to the whole word) are modeled within a word hierarchy. Models with not enough training samples are eliminated. The system is tested on a small and dynamical lexicon. The state-based tying proposed by Natarajan et al. [15] uses a mixture of 128 Gaussians associated to each state position of contextual models (trigraphs) corresponding to the same base character. The total number of models can also be reduced as in [18] by clustering all trigraphs according to contexts described not as characters but as ascending or descending strokes. Fink and Plotz [17] also proposed a system based on contexts, these contexts being defined as broad categories. A data-driven clustering of the 1,500 Gaussian densities of the mixture is performed at each state position and for each category. It is worth noting that clustering and tying the contextual models may offer, in addition to reducing the number of parameters, the possibility to automatically capture common contextual effects on handwriting a given letter.

The context-dependent system described below has some common characteristics with the context-based systems described in [15] and [17]. In both, trigraphs are modeled and parameters are shared at each state position. Our system contrasts with these previous approaches as our state clustering is knowledge-driven: We cluster models using decision trees where questions at each node are expert-based and specific to the way handwritten characters are drawn. In addition, we include context from the neighboring windows by adding dynamic features (derivative) in each feature vector. Such dynamic and contextual information drastically improves handwriting recognition (see Sections 3.1.3 and 4).

The paper is organized as follows: Section 2 gives an overview of the overall combined system. Sections 3 and 4 detail our sliding window systems. First, preprocessing and feature extraction are described in Section 3.1. Then, Section 3.2 presents the generic system, using context-independent models. Finally, Section 4 introduces the enhanced sliding-window system with context-dependent models where states are clustered using a decision tree. The last

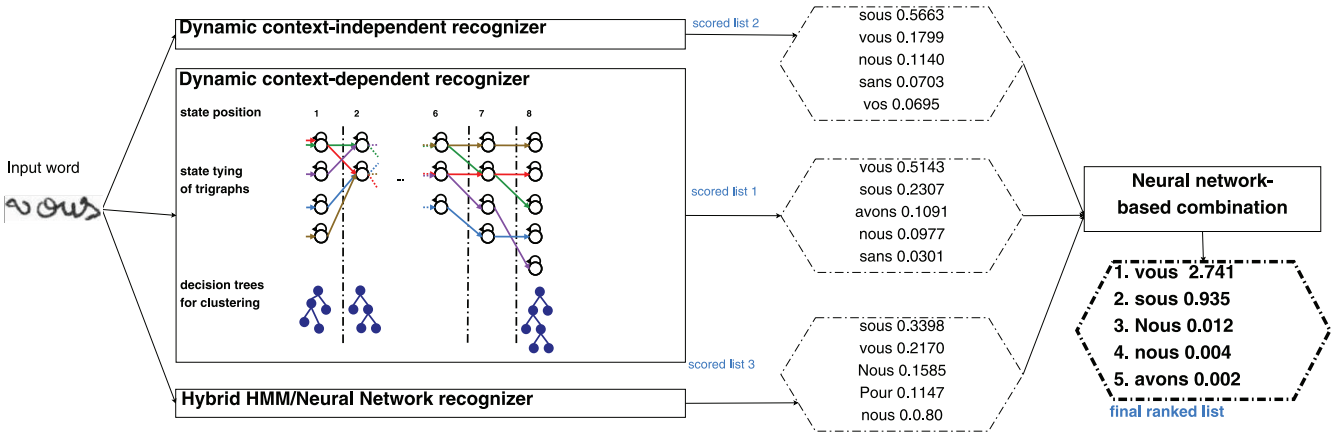


Fig. 3. System overview.

system, using explicit segmentation, is described in Section 5. Section 6 is dedicated to the proposed combination. We compare the enhanced system to the generic system and to related works in Section 7. Section 7 also compares the overall combined system to several state-of-the-art systems. Experiments are reported on three different databases: Rimes [19], IAM [20], and OpenHart [21].

2 System Overview

The aim of this study is to build an efficient word recognition system resulting from the combination of three HMM-based recognizers (see Fig. 3). The first system is a context-independent HMM-based system which includes dynamic information. The second recognizer is an original sliding window system which aims at enhancing performance accuracy and reducing decoding time. Performance is improved through contextual modeling, which provides a more accurate modeling of writing units. Decoding time reduction is obtained by reducing the number of Gaussian densities associated to state models. The smaller the number of Gaussians, the faster the computing of observation probabilities. In this work, reducing the number of Gaussian densities mainly relies on state tying. State tying consists of sharing the Gaussian densities associated to the states of trigraph models. Here, it is performed at each state position through a decision tree-based clustering. The decision tree is built according to expert-based questions on how characters are written. It is worth noting that in addition to decoding time reduction, state tying may lead to a better modeling by taking into account the dependency that may exist in different letters' models. The third system is a hybrid HMM/neural network-based system where observation probabilities are computed through a neural network.

These three systems are combined through a neural network-based classifier which outputs a scored list of candidate words. The combining system takes, as input, the scored lists provided by the three recognizers in order to fuse them in one output ranked list.

3 Dynamic Context-Independent System

The first HMM system is based on the sliding window approach. The system includes dynamic information related to derivative features. These features are computed from neighboring windows. As mentioned in Section 1, adding such features in the current window is one way to introduce context from neighboring characters. We will show in Section 7.2.2 that such derivative features are highly useful for the recognition of handwritten words. We first describe below the preprocessing and feature extraction steps of the context-independent system.

3.1 Preprocessing and Feature Extraction

3.1.1 Normalization and Preprocessing

The first steps of a recognition system consist of preprocessing the input images and extracting features. We limit the preprocessing to deslanting images since we have observed that words in the Rimes database are slanted with negligible writing skew. The features extracted are independent of the height of the word image and the HMM modeling can cope with sequences of variable length so that it can cope with variable word width. We also search for the main baselines around the word core zone to capture features related to descending and ascending strokes. Deslanting and baseline extraction are based on the work of Vinciarelli and Luettin [9].

3.1.2 Feature Extraction

Feature extraction is based on the work of El-Hajj et al. [11], [22]. The set of features is directly derived from the set defined by El-Hajj for Arabic handwriting, except for the fact that the sequence of vectors is extracted from left to right (through overlapping windows) instead of right to left. The windows are divided vertically into a fixed number of cells. Within each window, a set of geometric features is extracted; w features are related to pixel densities within each window's column (w is the width of the extraction window, in pixels), there are three density features extracted within the whole frame and above and under the lower baseline, two features are related to background/foreground transitions between adjacent cells, and three features to the gravity center position, including a derivative feature (difference between y positions). Twelve other features are related to local pixel configurations in order to capture stroke concavities. A subset of features is baseline dependent.

The features related to pixels densities are calculated directly on the gray-level images, unlike what is done in [22] and [11], where only binarized images are used, though

we estimated a binarization threshold for each image using the Otsu method [23] for the computation of features related to foreground/background transitions and pixel configurations. We optimized the features extraction parameters on a validation database: window width w, overlap between two sliding windows δ, number of cells per window n_c. The experimental setup is discussed in Section 7.2.1.

3.1.3 Dynamic Features

We introduce context at the feature extraction level through derivative features. In [5], Wieneke et al. also consider horizontal derivation of feature vectors in order to capture a wider temporal context at the frame level. Actually, such features represent the dynamics of features around the current window. The feature vector at horizontal pixel position p contains information on the frame at position p but also on the context of this frame from windows at positions $p - \delta{*}K$ to $p + \delta * K$ (δ is the shift of the window and K is the number of windows participating to the derivative feature). The derivation is computed with a regression. In the speech recognition domain, the first and second-order regressions are known as delta and delta-delta coefficients.

Let $\boldsymbol{o}_k$ be the feature vector at the horizontal pixel position p and $\boldsymbol{o}_{k+i}$ (respectively, $\boldsymbol{o}_{k-i}$) the feature vector of the sliding window shifted by $i * \delta$ (respectively, $-i * \delta$) pixels from the current window, at pixel position $p + i * \delta$ (respectively, $p - i * \delta$). The first-order regression of feature vector $\boldsymbol{o}_k$ is written

$$\Delta \boldsymbol{o}_k = \frac{\sum_{i=1}^{K} i(\boldsymbol{o}_{k+i} - \boldsymbol{o}_{k-i})}{2\sum_{i=1}^{K} i^2}, \quad (1)$$

K is the chosen depth of the regression, giving the number of surrounding feature vectors ($2 * K$) used for computing the dynamic features. The second-order regression $\Delta\Delta \boldsymbol{o}_k$ is simply derived from (1) by replacing $\boldsymbol{o}_k$ by $\Delta \boldsymbol{o}_k$. The final feature vector is thus the concatenation of the original vector $\boldsymbol{o}_k$, its first-order regression vector $\Delta \boldsymbol{o}_k$, and optionally its second order regression vector $\Delta\Delta \boldsymbol{o}_k$. We show in Section 7.2.2 that the dynamic features obtained by such derivations are efficient for handwriting recognition.

3.2 Training and Recognition

The dynamic context-independent system uses the analytical strategy and is segmentation-free. A word is modeled by the concatenation of its compound character models. All character models share the same HMM Bakis topology: S emitting states, one self transition, and left-right transitions allowed to the next two states. We consider HMMs with continuous observation densities so that the observation probability density for each state is a mixture of N_G Gaussian distributions. This mixture is obtained by incrementing, step by step, the number of Gaussian distributions in each state until a convenient HMM topology is reached. The number of Gaussian distributions is increased as follows:

- The mixture has n components and it is to be increased to $n + m$.
- For the kth mixture to be added, $k \in [1, \ldots, m]$, find the mixture with the largest weight and split this mixture:
  - divide the weight in two halves,
  - clone the mixture, and
  - add perturbation to each mean vector cloned by adding (respectively, subtracting) to it a small standard deviation σ_{split} (default σ_{split} is 0.2).

The HMM models are initialized with one Gaussian distribution per state and are trained thanks to the Baum-Welch algorithm. Then, for each Gaussian mixture incrementing step ($n- > n + 1$), HMM models are retrained with this same algorithm. The experimental setup is described in Section 7.2.1.

Since the system is context-independent, a single model corresponds to each letter. In the following, we call these context-independent models "monographs." The system aims at recognizing handwritten words in the French handwritten mail of the publicly available Rimes database (see Section 7.1). French words include a number of diacritical marks (accents) and the words' meanings are changed according to these marks. For instance "annule" means "cancel" and "annulé" means "canceled." This makes the recognition of French handwritten words a challenging task. This mail also includes contract numbers and dates (see Fig. 1). Thus, we define a total of 78 different case and accent sensitive monographs, which can be letters (accentuated or not), numerals, or symbols (hyphen, apostrophe, etc.).

Decoding is performed with the Viterbi algorithm. There is no prior on the likelihood of words so that all of the words contained in the test lexicon have the same probability of occurrence. We use the Hidden Markov Model Toolkit (HTK [48]) for training and recognition.

4 Dynamic Context-Dependent System

We have shown in the previous section how to introduce context at the feature level through derivative features. Now, we exploit the fact that a letter within a word adapts its shape to its adjacent characters. In the system presented here, the context is introduced at the model level: Characters' models differ according to their surrounding characters. In the speech recognition domain, such context-dependent modeling [24] has been proposed to take into account the coarticulation effect between phones.

As mentioned in the Introduction (Section 1), very few works in the handwriting recognition domain take advantage of context-dependent modeling. Actually, a major drawback of building context-dependent models is that the number of HMM parameters related to all possible letter contexts increases remarkably and there might be a lack of training data. This number of parameters can be reduced by sharing them between several models, and by selecting the characters to be context-dependent.

Our dynamic context-dependent system is illustrated in Fig. 4. This system is motivated by the fact that a large amount of character models (monographs) need refinement. We show in Section 4.1 that the likelihood of a number of monographs has a very large variance. In order

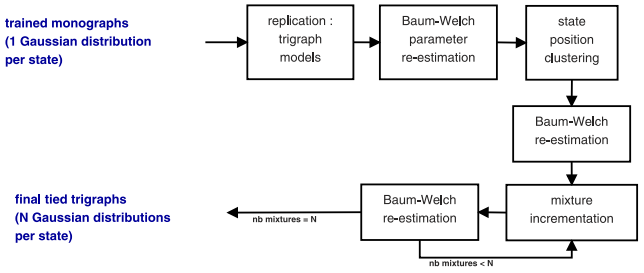


Fig. 4. Context-dependent HMM-based system overview.

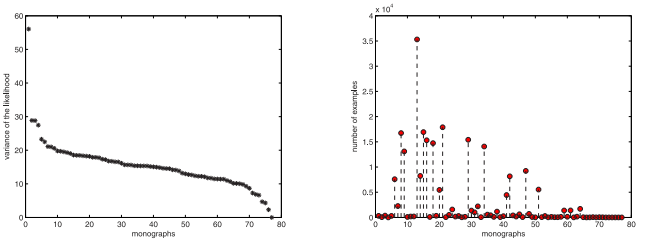


Fig. 5. (a) Likelihood's variance of monographs and (b) number of training samples for each monograph. Monographs are ranked according to likelihood's variance.

to efficiently train the system, reach high accuracy, and fasten decoding time, we propose building trigraph models instead of monographs and sharing parameters between those trigraphs. To reduce the number of trigraph parameters, we propose in Sections 4.3 and 4.4 a tying process which is based on a decision-tree state clustering relying on original expert-based questions. Preprocessing and feature extraction are similar to those described in Section 3.1.

4.1 Likelihood's Variance of Monographs

We propose using the likelihood's variance of a monograph to evaluate its efficiency for modeling a handwritten character. A monograph with a highly variable likelihood corresponds to a model that is not quite precise enough. The likelihoods are computed through Viterbi alignment for each monograph and in each training word. The likelihood's variance of monograph m_i is calculated as $\sigma^2_{L_{m_i}} = \frac{1}{N_{m_i}} \sum_{j=1}^{N_{m_i}} (L_{j,m_i} - \hat{L}_{m_i})^2$, where N_{m_i} is the number of examples of monograph m_i in the training data set, L_{j,m_i} is the likelihood of monograph m_i in the jth example, and $\hat{L}_{m_i}$ is the empirical mean of m_i's likelihoods.

Fig. 5a shows the likelihood's variance for each monograph. These monographs belong to the Rimes training database of handwritten words (see Section 7.1). Seventy percent of the monographs have a variance greater than 10 and 50 percent of them have a variance greater than 15. Fig. 5b shows the number of examples of the same monographs (ranked according to their likelihood's variance).

Looking to the Fig. 5b, it seems clear that the variance of the likelihood for a monograph is not necessarily related to the number of samples available. Monograph numbers 59, 61, and 64 in Fig. 5b do have small likelihoods' variances for a relatively large number of samples. Similarly, monographs number 1 to 5 have the largest variances for few examples. Hence, the selection of monographs to be contextualized could be performed on a validation set from a search of the best compromise between the number of examples related to a monograph and its likelihood's variance, though we decided to act on a finer level and the method presented in Section 4.4 operates at the state level. The proposed decision tree-based clustering selects which states of the monographs need to be contextualized and how these contextualized states should tie their Gaussian probability density functions.

4.2 Context-Dependent Character Models

The principle of context-dependent modeling of characters is quite natural, as it is obvious that our way of writing a character within a word evolves with its adjacent letters. An example of contexts influencing shape of letters can be seen in Fig. 2 for the lowercase characters "n" and "i." Considering that the two words have been written by the same person, we can easily imagine that the shape of letters is highly variable in a large writer-independent database.

The idea of context-dependent modeling is that instead of defining a word as a succession of characters, we define it as a succession of characters and their contexts. The monographs of Section 3.2 are replaced by trigraphs where the central character is influenced by the character on its left (left context) and the character on its right (right context). A left context is defined with a minus "$-$" sign, and a right context with a plus "$+$" sign. For example, in Fig. 2a, the letter "i" is surrounded by an "s" and an "e": The corresponding trigraph is written: $s-i+e$. Similarly, in Fig. 2b, the trigraph corresponding to the letter "d" surrounded by a blank and an "i" is written: $\emptyset - d + i$.

Once the models to be trained have been defined, the next step is the training phase. Training could be done directly on the trigraphs. However, a number of trigraphs which scarcely appear in training data would be badly trained. Moreover, the amount of computations to be made would be huge when increasing the number of Gaussian distributions per state. For a vocabulary of 1,500 words, more than 4,500 different trigraphs exist. If each final model had S emitting states and N_G Gaussian distributions, if S and N_G are of the order of 10, the number of distributions to compute would be nearly half a million. Considering that the Rimes database contains approximatively 50,000 words for the training phase, there may not be enough data to correctly estimate all parameters. That is why clustering and parameter tying are now taken into consideration. They solve two kinds of problems: lack of training data and an overabundance of models.

4.3 Training Overview

In this section, we present an overview of the training process for estimating the parameters of the trigraph models. The training process relies on parameter tying through state clustering, as illustrated in Fig. 6.

We start with trained monographs with one Gaussian distribution associated to each state. Trigraphs are initialized by copying monographs. All of the trigraphs associated to a given central letter are listed in the training database, and the initialized monograph model of the central letter is given as a first model for all those trigraphs. Then, a first and rough estimation of the trigraph

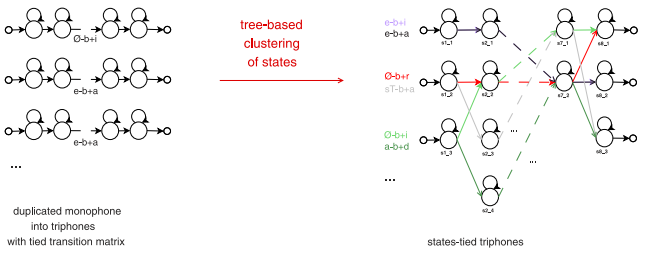


Fig. 6. Illustration of state clustering for the trigraphs centered on character b.

parameters is obtained with a single iteration of the Baum-Welch estimation algorithm on all of the different trigraphs. During the Baum-Welch estimation, we consider that state transition matrices are tied since we have observed that modifications in the transition matrix coefficients had very little influence on the recognition step. Regarding this, we impose that all trigraphs with the same central letter (all $*-b+*$ for instance) have the same transition matrix. Hence, the number of matrices to compute is reduced to the number of initial monographs. It can be noted that only the trigraphs present in the training database are created through this process.

The second step is the state tying process. State tying determines which states can share the same Gaussian distributions. As mentioned in Section 1, there are several methods which perform such tying. We choose a state position-based tying, such as performed in [17] and [15]. The principle is that for a given central letter, all states corresponding to the same position in an HMM model are subject to agglomerative clustering. Contrarily to [17] and [15], we use expert-based decision trees to perform state clustering. The process is described in detail in Section 4.4 and is illustrated in Fig. 6 for the central character b. It enables us to divide the total number of states by nearly ten.

Finally, a gathering of identical models is done. Actually, state clustering means fewer state models; many trigraphs find themselves sharing exactly the same states as other ones. This enables us to reduce the final number of different trigraph models by nearly a third.

4.4 Decision-Tree State Clustering

For a given character c, N_c different trigraphs with c as central letter exist. Let S be the number of states for trigraph models. For each state position i ($1 \leq i \leq S$) of the $*-c+*$ trigraphs, N_c different state models are to be computed and the total number of states, for all trigraphs centered on c, is $N_c{*}S$. The tree-based clustering of states enables the reduction of the number of different state models N_c for each state position. It results in pools of states, so that the ith state of each $*-c+*$ trigraph takes its value within the $n_{c,i}$ different models defined after the clustering, where $n_{c,i} < N_c$.

Decision tree-based clustering is an alternative to data driven clustering based on another distance measure between models (maximization of the log-likelihood) and on a more controlled split of clusters. The merging or splitting of state clusters is driven by a binary tree whose nodes correspond to rhetorical questions on the characteristics of the models. Such decision trees have been designed for speech recognition at the phone level by experts [25]. To our knowledge, no work using such trees for handwriting recognition exists.

In our case, decision trees are based on a set of questions on the behavior of left and right contexts, and are applied to states. Based on the same initial set of questions, one tree is built for every state position of all trigraphs with the same central letter. Starting at the root node, all of the states corresponding to the same position and the same central letter are gathered in a single cluster. Then, the binary question which maximizes the likelihood of the two children clusters it would create is chosen, and the split is made, creating two new nodes. This splitting continues until the increase in likelihood falls below a threshold or no questions are available to create nodes with a sufficient state occupancy count.

Let us consider a node containing the set of states $\boldsymbol{S}$ to be split in a given tree. The set $\boldsymbol{S}$ corresponds to the set of training frames $\{o_f\}_{f\in\boldsymbol{F}}$. As all states in $\boldsymbol{S}$ are tied in the node, they all share the same mean $\mu(\boldsymbol{S})$ and variance $\Sigma(\boldsymbol{S})$. The likelihood of $\boldsymbol{S}$ generating the set of frames is hence given by

$$L(\boldsymbol{S}) = \sum_{f\in\boldsymbol{F}} \sum_{s\in\boldsymbol{S}} log(Pr(\boldsymbol{o}_f; \mu(\boldsymbol{S}), \Sigma(\boldsymbol{S})))\gamma_s(\boldsymbol{o}_f), \tag{2}$$

where $\gamma_s(\boldsymbol{o}_f)$ is the a posteriori probability of frame $\boldsymbol{o}_f$ being generated by state $\boldsymbol{s}$. Based on the work of Young et al. [26] and assuming that we work with Gaussian probability density functions, $L(\boldsymbol{S})$ can be rewritten

$$L(\boldsymbol{S}) = -\frac{1}{2}(log[(2\pi)^n|\Sigma(\boldsymbol{S})|] + n)\Gamma(\boldsymbol{S}), \tag{3}$$

$\Gamma(\boldsymbol{S})$ is the accumulated state occupancy of the node, $\Gamma(\boldsymbol{S}) = \sum_{f\in\boldsymbol{F}}\sum_{s\in\boldsymbol{S}}\gamma_s(\boldsymbol{o}_f)$, and n is the dimension of the feature vectors.

We then introduce ΔL_q:

$$\Delta L_q = L(\boldsymbol{S}_{q+}) + L(\boldsymbol{S}_{q-}) - L(\boldsymbol{S}). \tag{4}$$

The split of the state set into two subsets $\boldsymbol{S}_{q+}$ (the answer to q is yes) and $\boldsymbol{S}_{q-}$ (the answer to q is no) is made by question q^* which maximizes ΔL_q, provided that $\Gamma(\boldsymbol{S}_{q+})$ and $\Gamma(\boldsymbol{S}_{q-})$ are over the minimal state occupancy threshold and that ΔL_q is above the threshold of minimal increase in likelihood. This condition can be reformulated [27]:

$$\begin{aligned} q^* = argmin_q \Big\{ &\Gamma(\boldsymbol{S}_{q+})log(|\Sigma(\boldsymbol{S}_{q+})|) \\ &+ \Gamma(\boldsymbol{S}_{q-})log(|\Sigma(\boldsymbol{S}_{q-})|) - \Gamma(\boldsymbol{S})log(|\Sigma(\boldsymbol{S})|) \Big\}. \end{aligned} \tag{5}$$

The parameters ensuring efficient sizes of state clusters, namely, the minimal state occupancy threshold and the minimal increase in likelihood, are tuned on the validation database (see Section 7.2.3). For a given monograph, the depth of the resulting trees is linked to the monograph likelihood variance and to the number of examples mentioned in Section 4.1. Actually, the larger the variance and the number of samples, the deeper the monograph's state trees and the larger the number of different trigraphs defined from the original monograph. Similarly, trees reduced to only their root are observed, corresponding to

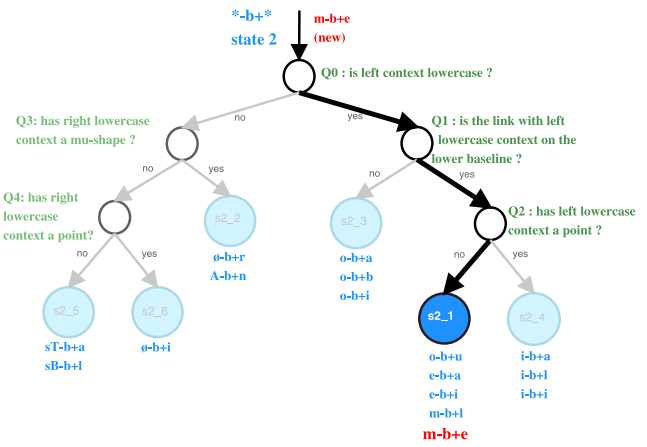


Fig. 7. Example of a decision tree for state clustering (questions and clusters are shown for the second state of all $-^{*}\mathrm{b}+^{*}$ trigraphs), and example of cluster allocation with a trained decision tree for state position number two of unseen triphone $\mathrm{m}-\mathrm{b}+\mathrm{e}$.

monographs with few examples which aim to tie all of their corresponding trigraphs into a single model.

We defined a set of expert-based questions that gathers look-alike character's context shapes. These questions are used to build the state clusters trees. Trigraphs within the same character-cluster may tie one or more of their states within a state-cluster, as shown in Fig. 7. To ensure efficient parameter tying, the shape of the context should be similar enough for the trigraphs in the same cluster. The set of questions includes global questions, yielding large clusters of characters, and more precise ones, yielding smaller clusters. It is important to propose both global and precise questions. If all questions were precise, only a few states could be shared within trigraphs. On the contrary, if all questions were global, a very large number of states would be shared, resulting in models similar to monographs. Here are some examples of questions:

- (global question) Is the left context uppercase? lowercase? Does it have an ascender? A descender? Or is it a small letter?
- (global question on the main shape of a context) Does the right context contain a loop ("$*-\mathrm{o}$", "$*-\mathrm{a}$", but also "$*-\mathrm{d}$", etc.)? A bar ("$*-\mathrm{t}$", "$*-\mathrm{p}$", etc.) ?
- (precise question on the shape of a context) Is the link with the previous letter on the upper baseline ("$\mathrm{v}+^{*}$", "$\mathrm{w}+^{*}$", etc.)? On the lower baseline ("$\mathrm{a}+^{*}$", "$\mathrm{c}+^{*}$", etc.)?

An example of decision tree is given in Fig. 7. It represents the tree calculated for the second state of all the trigraphs centered on the lowercase character b. (Bold lines are not to be taken into account for now on Fig. 7). Some of the questions we defined can be found in Appendix A, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.22, the rest of them are downloadable from [28].

Finally, once the state tying is completed, the system is ready to finish the training of the models with a high level of refinement. Using the Baum-Welch algorithm for model reestimation, we increase the number of Gaussian distributions per state, up to a chosen number N_G. We finally reach a similar topology to the generic context-independent system, with S emitting states and N_G Gaussian distributions per state for all of the monographs and trigraphs. This enables us to directly compare the two sliding window-based systems.

4.5 Recognition

Apart from the interesting expert-based clustering presented above, the tree-based clustering also overcomes the problem of unseen trigraphs. Actually, if we had chosen data-driven clustering, the state models would be clustered in a different way (data-driven clustering is based on euclidean distances between states) and new trigraphs not seen in the training phase could not be modeled. Since test lexicons often differ from training lexicons, the capability of modeling new unseen trigraphs is essential.

The modeling of new trigraphs with decision trees is very simple: Each state of the trigraph is positioned at the root node of the tree corresponding to the same state position and the same central letter. Then, each state descends the tree to which it belongs, answering questions on the trigraph contexts, until it reaches a node where a cluster is positioned. The state model representing the cluster will be the model assigned to the considered state number of the trigraph for its recognition. This is illustrated with bold lines in Fig. 7 for state position number 2 of the new unseen trigraph $m-b+e$. The clustering tree for this state and this central letter was computed during the training phase. Placing state number 2 of $m-b+e$ at the root node of this tree, a first question is asked: *"Q0: Is left context lowercase?"* As m is lowercase, the answer is *yes*, so the state goes down to the right-hand side node, where a new question is asked (*"Q1"*) and answered (*yes*), etc. In the end, state number 2 of $m-b+e$ reaches the cluster named $s2\_1$: It is the chosen model for this state.

The ability of trees to model any unseen trigraph with existing ones was very helpful for our experiments on the Rimes database, which will be presented in Section 7.1, as the test and training dictionaries were different and hundreds of new trigraphs had to be modeled.

5 Hybrid HMM/Neural Network System

The third system in the combination is also HMM-based but relies on a presegmentation of words into grapheme components, in contrast to previous systems which rely on sliding windows.

5.1 Grapheme Segmentation

In this system, the word recognition is based on an explicit segmentation of the word in subparts called graphemes. The segmentation is an oversegmentation, which means that a grapheme is either a character or a subpart of a character. The grapheme segmentation process is composed of the following steps: First, we detect the connected components and extract the internal and external contour information. Then, the skeleton of each connected component is computed and represented in a graph. Potential segmentation points are detected (bottom part of concavities, extremities of horizontal segments). Graphemes are defined as vertical or diagonal arcs which can be linked

without crossing a potential segmentation point. Then, the remaining arcs are either attributed to a neighboring grapheme or identified as a ligature. Finally, images of each potential grapheme are retrieved from the labeled arcs.

5.2 Feature Extraction

The features extracted for each grapheme are relatively simple and can be computed very quickly. They are composed of the height and width of the bounding box of the grapheme (2 values), height-width ratio (1 value), position of the top and bottom of the bounding box with respect to the baseline (2 values), position of the gravity center of the grapheme in the bounding box (2 values), black pixel density in the bounding box (1 value), black pixel density in three zones: above, under, and inside the baseline (3 values), surface of the loops in the three previous areas (3 values), value of the top, bottom, left, and right profiles of the grapheme taken in five points (4^*5 values), cumulated thickness of the grapheme, horizontally, vertically, and along the two diagonals (for each direction, the thickness is computed in five parallel areas—4^*5 values). The features are normalized with respect to the baseline height. The total number of features is 74.

5.3 Neural Network

In this system, a neural network is trained to evaluate the posterior probability of each grapheme with respect to the feature vector. The neural network is a multilayer perceptron with as many input neurons as features (74), one layer of hidden neurons (400 neurons), and as many output neurons as grapheme classes (172). The transfer function is a softmax function. The neural network was trained in a supervised way with a stochastic back-propagation training algorithm.

5.4 Training and Recognition

Hidden Markov models are used to model the decomposition of the words into letters and then each letter into graphemes. The topology of the model is a simple left-to-right Bakis model topology with three states for each letter HMM, each state corresponding to a grapheme. The hybrid NN-HMM was trained using the following procedure:

1. Decode the training set with the hybrid NN-HMM recognizer in order to create an annotated base of feature vectors.
2. Train the neural network on the annotated base of feature vectors.
3. Compute the sequences of observation probability with the new neural network for all words in the training set.
4. Train the HMM on the sequences of observation probability using the Baum-Welch algorithm.
5. Go back to item 1 until no improvement is observed in the recognition rate.

This procedure needs a basic recognizer to bootstrap the process; the convergence is usually observed after 10 to 20 iterations.

6 System Combination

Classifiers combination methods have proven successful in many applications, such as handwritten (isolated or cursive) and printed recognition and verification, document analysis, speech recognition, medical imaging, biometric verification (face, signature, fingerprints recognition), information retrieval, and expert systems. See [29] for an extended survey. Several combination schemes are possible (in sequential, parallel, or hierarchical order), with and without training [30] [31]. Here, we consider only combinations in parallel since we combine lists of candidates provided by expert classifiers considered as independent.

In [32] and [11], Al-Hajj et al. proposed a new neural network combination method in order to combine the results of three HMM word classifiers with sliding windows of different angles. This combination scheme provided better results than standard state-of-the-art methods (like Sum-Rule, Plural Vote, and Borda Count). El-Abed and Märgner [33] also applied this method to a larger number of classifiers. They successfully combined the results of the 11 systems submitted to the ICDAR 2007 Handwritten Arabic Competition. Here, we propose an improved neural network architecture, extending the work presented in [32] and [11]. This architecture can reorder the list of candidates so as to be able to compute the right answer even if none of the individual classifiers provides it in the first position.

6.1 Individual Classifiers' Outputs

Each word classifier is denoted by C_i. Since here we have three HMM classifiers, $i \in \{1, 2, 3\}$. For every image to be recognized, each classifier C_i provides a list of d candidates c_{ij} along with scores $s_{ij} = S_i(c_{ij})$, with $j \in \{1, \ldots, d\}$. The scores s_{ij} are the normalized likelihoods of the d-best word candidates for classifier C_i:

$$s_{ij} = P(O|M_{ij}) \Big/ \sum_{k=1}^{d} P(O|M_{ik}), \tag{6}$$

where M_{ij} is the HMM model of classifier C_i for the word c_{ij}. For each word to be recognized, the word classifier number i (C_i) provides the following list of candidates:

$$\begin{pmatrix} c_{i1} & s_{i1} = S_i(c_{i1}) \\ c_{i2} & s_{i2} = S_i(c_{i2}) \\ \vdots & \vdots \\ c_{id} & s_{id} = S_i(c_{id}) \end{pmatrix}.$$

W is the lexicon. $\forall w \in W$, $S_i(w)$ is the score of word w according to classifier C_i. If $w \notin \bigcup_{j=1}^{d} c_{ij}$, then $S_i(w) = 0$.

6.2 Standard Methods

Sum-Rule. The sum-rule is one of the easiest ways to combine lists of candidates. Yet it is quite resilient to estimation errors [30].

Plural Vote. Plural Vote is a rank method. It considers the top choices of each classifier. The class that is the most frequent in the first d ranks is considered the best class.

Borda Count. Like Plural Vote, Borda Count is also a rank method. A score is computed depending on its rank in the list. If lists of d candidates are considered, the first place

candidate receives d votes, the second candidate $d-1$, etc., and the words of the lexicon not in the list receive 0 votes.

Exponential Borda Count. Exponential Borda Count is a modified version of the Borda Count method, presented in [34], resulting from the observation that the probability of the correct word being at rank i decreases faster than linearly as i increases. Hence, the ith-place candidate receives a score of $(d-i+1)^p$ instead of $d-i+1$. The value of p is found on a validation database.

6.3 Neural Network-Based Combination

Here, we use a neural network similar to the one proposed by Al-Hajj et al. [32]. For the sake of completeness, we briefly recall the method. A Multilayer Perceptron with one hidden layer is used. A feature vector of dimension N^2 (where N is the number of classifiers) is extracted from the lists of candidates provided by the classifiers (using the scores described above, see (6) in Section 6.1). The neural network has N sigmoid outputs. Each of those N outputs (denoted out_i with $i \in \{1,\ldots,N\}$) represents the confidence value of the corresponding classifier. The number of hidden cells is determined empirically. As a result of the combination, the top candidate of the classifier ending with the highest confidence score according to neural network is selected: $class_{NN} = c_{k1}$, where C_k is the most reliable classifier according to the neural network. Hence,

$$out_k = \max_{i\in\{1,\ldots,N\}} out_i.$$

We propose an extension of this combination scheme. The neural network described above was trained to select the best classifier. Then the result of the combination was the best answer of this selected classifier. A drawback of this method is that if none of the classifiers output the right answer in first position, the combination has no chance of finding the true class (no reordering of answers is made). The feature vector considers only the results in first position for each of the classifiers. One way to improve the results could be to go deeper in the candidate lists in order to allow more answers than just the top candidate of each list. The size of the feature vector now becomes $N^2 \times d$ and the size of the output vector $N \times d$. Again, the number of hidden cells is found empirically.

The feature vector for the combination of 2 classifiers is

$$feature\_vect(d) = \Bigg[\underbrace{S_1(c_{11}), S_2(c_{11}), S_1(c_{21}), S_2(c_{21})}_{\text{1st position}}, \ldots, \underbrace{S_1(c_{1d}), S_2(c_{1d}), S_1(c_{2d}), S_2(c_{2d})}_{d-\text{th position}}\Bigg]^T,$$

where $S_i(c_{kj})$ is the score provided by classifier i for the hypothesized word c_{kj} ranked at jth position by classifier k (see Section 6.1). This feature extraction can be easily extended to N classifiers.

In this extended scheme, the first answer is not the only one considered. The output vector has $N \times d$ components. Here, we describe the target vector for the combination of two classifiers. This target vector can be easily extended to N classifiers. If w^* is the true class of the word, then:

$$target\_vect(d) = \Bigg[\underbrace{\delta(c_{11}, w^*), \delta(c_{21}, w^*)}_{\text{1st position}}, \underbrace{\delta(c_{12}, w^*), \delta(c_{22}, w^*)}_{\text{2nd position}}, \ldots, \underbrace{\delta(c_{1d}, w^*), \delta(c_{2d}, w^*)}_{d-\text{th position}}\Bigg]^T,$$

and

$$\delta(x,y) = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{otherwise.} \end{cases}$$

This new neural network provides a rescoring of the candidates c_{1i} ($i \in \{1,\ldots,d\}$) and c_{2j} ($j \in \{1,\ldots,d\}$). Then the sum-rule is applied to merge the answers of candidates which correspond to the same word classes (i.e., those for which $c_{1i} = c_{2j}$). The example of Fig. 3 shows the lists of candidates for all three systems and their combination with a neural network taking the five best answers of each recognizer ($d = 5$). The detailed procedure to combine the three outputs, given in Appendix B, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.22, shows the interest of applying the NN approach.

7 Experiments

7.1 Rimes Database

The Rimes database [35], [36] was presented in 2006 and gathers more than 12,500 handwritten documents written by about 1,300 volunteers. It was created to provide data relative to advanced mailrooms, and the panel of documents offers large variability and makes the database a challenging one. Based on the Rimes database, it is possible to proceed logo recognition, document structure retrieval, and word and character recognition. Since 2007, evaluation campaigns have occurred [19] which enable participants to compare their results. In our work, we use the presegmented word images given by the database to perform isolated word recognition. The database consists of 59,203 word images divided into three subsets: 44,197 images for training, 7,542 for validation, and 7,464 for testing. Word samples are shown in Fig. 1.

For the Rimes database, three experiments are conducted using different dictionary sizes. The first experiment considers the full dictionary of size 5,334, including all words from the training, validation, and test sets. The second experiment considers only the reduced test dictionary of size 1,612 words. The last experiment uses the 4,943-words dictionary of training and validation databases, giving 392 out-of-vocabulary words (5.5 percent of the test set). About 300 new trigraphs are present in the test set and not in the other sets. This yields to the unseen trigraph problem mentioned in Section 4.5.

7.2 Experimental Setup

7.2.1 Basic HMM Parameters for the Sliding Window Systems

We first tune the basic parameters for the sliding window systems. These parameters are the width of the window, the number of cells, and the shift of the window for feature extraction, and the number of states and size of mixtures for

TABLE 1
Setup of Parameters on a Generic Context-Independent Recognizer

features set	w=8 δ=4 $n_c = 20$	w=10 δ=5 $n_c = 20$	w=10 δ=6 $n_c = 20$	w=14 δ=7 $n_c = 20$
optimal N_S	8	8	6	6
8 Gaussians	61.6%	59.2%	53.9%	50.5%
14 Gaussians	65.6%	63.7%	58.5%	57.1%
20 Gaussians	68.4%	65.4%	61.1%	59.1%
30 Gaussians	69.8%	66.2%	62.0%	60.5%
40 Gaussians	71.8%	67.9%	64.7%	62.7%

The parameters are the window width w, the shift δ, the number of cells n_c, and the number of states for character models. Recognition rates are computed on the Rimes validation database.

character models. We set these parameters on the validation set for the Rimes database with a generic HMM recognizer, context-independent, where no difference is made between a letter accentuated, lowercase, or uppercase. Hence, "a," "A," and "à" share the same models. This generic system is trained on the Rimes train data set with different sets of parameters and the corresponding accuracy is computed on the validation data set. We calculated many sets of features from the images, making the values of the window width w, the shift δ, and the number of cells n_c vary. Values were tested between 6 and 14 pixels for w, and between 10 and 20 for n_c. We tested two values only for δ: $w/2$ and $w/2+1$. For each set, we made the number of states per model vary from 5 to 15 in order to find the optimal number needed. After this first exhaustive search, we noticed that higher values for n_c were the best, so we set n_c to 20. We then kept the four best sets of features along with their optimal number of states per character model, and made the number of Gaussians vary in each mixture from 8 to 40. Table 1 shows these four experiments. The recognition rate is computed case and accent-insensitive.

From these preliminary experiments, we choose to pick the best feature set on the validation database, that is to say, the one using windows of width $w = 8$ pixels, with a shift of $\delta = w/2 = 4$ pixels between two consecutive windows and a vertical division of $n_c = 20$ cells, giving 28 different features. We call this feature set the w8d4c20 set. The best number of emitting states for this set is $S = 8$. We choose to set the number N_G of Gaussian distributions per state to $N_G = 20$. To make this choice, we considered the compromise between accuracy and computational cost: Decoding time is decreased by 50 percent when using $N_G = 20$ instead of $N_G = 40$, while the accuracy drops only of 3 percent, and the difference between $N_G = 20$ and $N_G = 30$ is not large enough to keep 50 percent more parameters to compute. Finally, the basic HMM-sliding window systems parameters for the Rimes database are: the w8d4c20 feature set, with 8 emitting states and 20 Gaussian distributions per state.

TABLE 2
Comparison of the Different Orders of Regression for Features on a Generic Context-Independent Recognizer

derivation type	none	1st order	1st & 2nd order
recognition rate	68.42	72.33	71.89

Recognition rate is computed on the Rimes validation database.

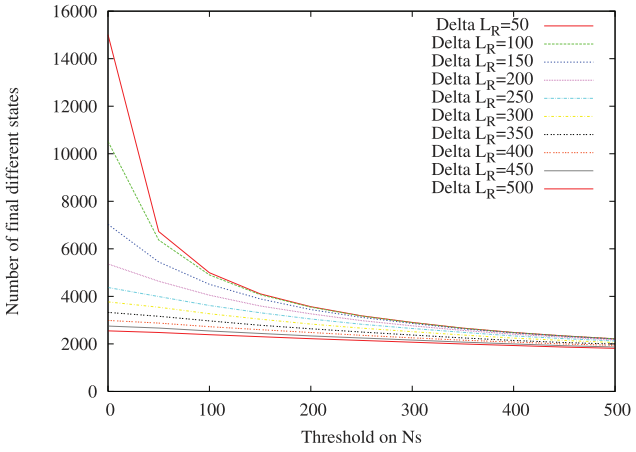


Fig. 8. Influence of N_S on the final number of different states, for fixed values of ΔL_R, on the Rimes validation database.

7.2.2 Features Derivation

Using the generic case insensitive recognizer presented previously for the Rimes feature extraction parameter setup and the w8d4c20 feature set with the convenient HMM topology, we ran experiments on the base of first and second-order regression of the features on the Rimes database. The results are shown on Table 2 for systems trained and tested on the same training and validation sets as Table 1.

We can deduce from this experiment that regression significantly increases the recognition rate. However, adding second-order regression decreases performance for the Rimes database. This may be due to the overabundance of features created compared to the size of the database (the new number of features in this case is $3 * 28$, where 28 is the initial number of features). Further investigation on larger databases might show if an enhancement can be observed when using second-order regression, but it will not be used in these experiments.

7.2.3 Thresholds Setup for Tree Splitting

As is presented in Section 4.4, tree splitting is done by finding question q^* maximizing $\Delta L(q)$. Some parameters have yet to be set in order to control the tree's depth and the maximization step: ΔL_R and N_S.

- ΔL_R is the minimal value $\Delta L(q^*)$ can take at each step. If the maximum of ΔL for a given node is below ΔL_R, then the node is not split and becomes a states cluster.
- N_S is the "minimal state occupation count" corresponding to a cluster of states $j \in \boldsymbol{S}$. It ensures that the states in the cluster are correctly estimated. For a given state j, N_j is the sum over all the training frames of the probability of being in state j while a given frame is observed. N_S is the sum of the N_js over the set $\boldsymbol{S}$.

These two parameters are set on the validation database.

Fig. 8 shows the influence of N_S on the final number of different states in the Rimes system, given ΔL_R. For these experiments, we used mixtures of 10 Gaussian distributions per state.

Clearly, the higher ΔL_R and N_S, the fewer the number of final states. Fig. 9 shows the influence of this number of states on the recognition rate on Rimes validation database.

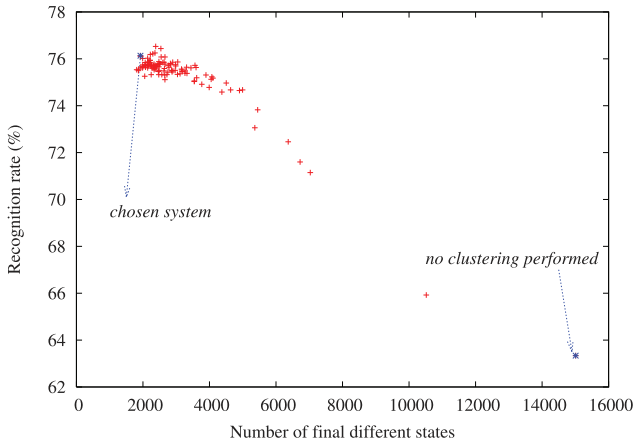


Fig. 9. Influence of the final number of different states on the recognition rate (Rimes validation database; 8 states/trigraph, 10 Gaussians/state topology).

With this in mind, we chose to set $\Delta L_R^* = 450$ and $N_S^* = 450$ for our system used on Rimes. The point corresponding to $(\Delta L_R^*, N_S^*)$ is colored in blue on Fig. 9. Graphically, it corresponds to the leftmost-utmost point, that is to say, the best equilibrium between the recognition rate (76.13 percent), the recognition time and the clusters computation on the validation database for an 8-state per trigraph, 10 Gaussians per state topology. This system with these parameters defines 1,924 different states.

Because of the decrease in the number of states, some trigraphs happen to share exactly the same clusters for their state models; hence they are strictly equivalent. This brings the final number of different trigraphs to 2,130. This decrease also causes some trigraphs centered on the same character to share a unique model: the monograph model. Actually, some monographs have few occurrences in the training database, so, split into different trigraphs, the state occupation counts do not reach high levels. The following central characters are those whose trigraphs models gather all in one (given the parameters chosen in this paragraph): 1, 2, 3, 4, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W, X, Y, Z, k, w, y, z, à, â, ç, É, è, ê, ë, î, ï, ô, ù, û, /, and %.

This makes 51 remaining monographs within the trigraphs, and $78 - 51 = 27$ monographs split in 2,080 different models, sharing about 1,530 states. The systems results shown on Fig. 9 are computed with mixtures of 10 Gaussian distributions per state. For the chosen pair $(\Delta L_R^* = 450,\ N_S^* = 450)$, we incremented the number of mixtures up to 20 in order to compare our new results to those we previously published on the same database [37]. We observed a 15 percent reduction of the error rate, jumping from 74.1 to 78.0 percent of words correctly recognized on the Rimes validation set. This shows that our new method to learn models with tied states is more precise and helps us to reach a better accuracy.

7.2.4 Sliding-Windows Systems Results

Results for the sliding windows systems on the Rimes2009 database [19], respecting the 2009 word recognition competition procedure, are given in Table 3. In this table, CI means context-independent system, CD means context-dependent, T stands for Rimes training database (44,197 images), and T+V for Rimes training plus validation database (51,739 images). The size of each lexicon used is given into the brackets. The first system is the context-independent case-sensitive system run as explained in Section 3.2, with eight states and a mixture of 20 Gaussians per state. The second system is the context-dependent system presented in Section 4. We counted a difference of 297 undefined trigraphs between the test and train-valid lexicons. The decision trees presented in Section 4.5 enables us to associate these trigraphs to existing state clusters.

TABLE 3
Comparison of the Proposed Sliding Window Systems, with or without Context Dependent Models, on the 7,464 Images in Rimes Test Database

system	Training Dataset	Test Lexicon (1612)	Full Lexicon (5334)	Tr+Val Lexicon (4943)
CI	T	73.04%	66.0%	63.61%
CI	T+V	75.52%	68.57%	65.77%
CD	T	79.34%	74.33%	71.07%
CD	T+V	**80.35%**	**75.53%**	**72.31%**

The skeleton of this system is learned on the training database: Decision trees and state clusters allocation to trigraphs, as well as splitting thresholds, are set with the 44,197 training samples. The final number of different states is 1,924. In order to improve our system accuracy, we add the validation data to the training database, giving 51,739 examples for training. The improvement can be seen in Table 3.

We can bring two main conclusions on the presented sliding window systems from Table 3. First, the addition of data in the training phase improves the recognition rate: The improvement is greater than 2 percent for the context-independent system and greater than 1 percent for the context-dependent one. This is motivation to continue adding data in the training step, after setting the system skeleton. This issue is not discussed here, as we follow the protocol of the Rimes competition, but this can be kept as a system improvement for other applications. Second, the modelization of characters with their context drastically improves the recognition rate, whatever the size of the dictionary. An increase of 4.8 percent in the accuracy is observed for the smaller dictionary, and of more than 6 percent for the larger ones. These are very interesting results: The context-dependent system performs better than the context-independent one. The parameters of the context-dependent system were efficiently estimated thanks to the clustering method. Such clustering overcomes the issue of the large number of trigraph models which increases with the size of dictionaries. Without clustering, the accuracy of the context-dependent system falls to 63.3 percent, as shown in Fig. 9.

7.3 System Combination

In this section, we present the results of the three systems presented in this paper and their combination on the Rimes database.

7.3.1 Motivation

The counts of words correctly and wrongly recognized by the context-independent and context-dependent systems on the Rimes test set are presented in Table 4. The systems are

TABLE 4
Comparison of the Outputs of the CI and CD Sliding Window Systems on the Rimes Test Database

system performance	CI wrong	CI right
CD wrong	1,119	472
CD right	893	4,980

Systems are trained on the training+validation database and tested with the 1,616-words lexicon.

both trained on the largest database and tested with the small dictionary. It appears that, if the context-dependent system outperforms the context-independent one, there are still 472 words well recognized by the latter, giving errors for the former. This encourages us to use both systems in combination, as different information is brought by the two output lists.

We also observed from Table 4 that 1,119 words are badly recognized by the two sliding-windows systems: This represents 15 percent of the test database. This is why we propose combining all systems, the two sliding-window systems and the hybrid HMM/NN system presented in Section 5. The recognition rates of the hybrid system on the test set are of 79.5 and 72.5 percent for the test dictionary and the full dictionary, respectively. We believe that the combination of these three systems will improve the overall performance since they rely on different modeling strategies.

7.3.2 Parameters Setup for Neural-Network Combination

In [38], Duin criticizes the idea of training the combination system on the same training set that has already been used to train the individual classifiers. The reason is because the outputs of the classifiers on the training set are not representative for new objects. We have noticed empirically that this claim is true. Training the individual classifiers and the combination system on the same data led to poor combination results, while splitting the training set and using one part for training the individual classifiers and the other for training the combination system provided much better results.

The training set of 44,197 images is thus split into 36,445 and 7,752 images, with the list of their scores provided by the HMM systems. The first part is used to train the three HMM systems and the second one is divided into two subsets (6,201 images for training and 1,551 for validation) to train the neural networks combination system. The result of the combination is estimated on a 7,742 independent test set.

We vary the considered depth (d) of the candidate lists (which influences both the number of inputs and the number of outputs of the neural network) from 1 to 10, and also the number of hidden units (nhu) from 5 to 50 by increments of 5. The right answer is usually among the first candidates of each list. Increasing the value of d gives the system the capability of retrieving the right answer even if it is deeper in the list. But it also makes the task much more difficult since it also introduces noisy features that are irrelevant most of the time, and increases the dimensionality of the feature vectors and the number of parameters of the neural network. The values of d and nhu are a trade-off between the capacity of the neural network and the amount of data available to train its parameters.

TABLE 5
Comparison of the Different Combination Methods Proposed on the Rimes 2009 Test Set

Combination method	Recognition rate
Borda Count	84.5%
Plural Vote	85.5%
Exp. Borda Count	87.8%
Sum Rule	88.8%
Neural Network	**89.1**%

In the experiments we performed on a validation set, the configuration with $d = 5$ and $nhu = 30$ provided good and consistent results. It is selected as the best configuration given the amount of data used to train the combination neural network.

7.3.3 Combination Results

The three input candidate lists for the combination do result from the two sliding windows systems (one context-independent and one context-dependent), and the hybrid system presented in Section 5. The result of their combination via the different methods proposed in Section 6 are presented in Table 5. Results are shown for the Rimes test set, using the limited test dictionary.

Table 5 clearly shows that combining classifiers outputs can drastically reduce the error rate. On average, each classifier achieves a little less than 80 percent of recognition rate individually. Combined, they can reach nearly 90 percent, which corresponds to a 50 percent error reduction. Regarding the combination methods, those based on the rank (Borda Count, PluralVote, Exponential Borda Count) are less efficient than the ones using the recognition score (Sum-rule, neural network). The best recognition rate is reached with the neural network combination, with a recognition rate of 89.1 percent.

7.3.4 Extension to Other Databases

The rhetorical rules designed for the Latin cursive script and for the Rimes database can easily be applied to another database. As an example, we apply the same rule set to the publicly available IAM database [20] which provides 9,862 handwritten text lines segmented into words. We extract from the correctly segmented and annotated text-line images [39] a training set of 46,901 word images, two distinct validation sets containing, respectively, 6,442 and 7,061 word images, and a 13,752 words test set. We observed that the sizes of the characters in this database (in terms of pixels) were similar to the sizes of the characters in Rimes database. Hence, we chose to use the same w8d4c20 extraction parameters than those used for Rimes and to search for the accurate number of states per character model. The training/validation sets are used to select this number of states and the $(\Delta L_R^*, N_S^*)$ clustering parameters. The best parameters for the IAM database are the following: $S = 10$ emitting states per character model, $\Delta L_R^* = 450$, $N_S^* = 600$. As previously, the system is retrained with all training and validation sets, keeping the same tree structures provided during setting of the $(\Delta L_R^*, N_S^*)$ parameters.

We use a lexicon containing all the 10,500 words of the IAM database (built from the training, validation, and test sets). It can be noted that the size of this dictionary is greater

TABLE 6
Performance of the Proposed Systems on the IAM Test Database

HMM-based system	recognition rate on IAMdb
context-independent	64.6%
context-dependent	**67.3%**
proposed combination	**78.1%**

than that of the Rimes test dictionary (1,600 words). Results for the IAM database are shown in Table 6. Recognition rates are case sensitive. Results show that the context-dependent system based on the same set of questions outperforms the context-independent one. This shows that the questions included in our rule set can be used for any language and database using the same set of Latin characters.

We also trained and tested the hybrid system on the IAM database in order to combine the three outputs as proposed for the Rimes database. We reached an accuracy of 78.1 percent on the word recognition task with the 10,500 words lexicon and without using any language model. Thus, a comparison with systems such as [40] and [41], which process whole text lines and use a language model, is not straightforward. However, our 78.1 percent word recognition accuracy is comparable to the accuracies obtained at the word level by those systems: 74.1 and 79 percent, respectively.

We performed another experiment on Arabic script, using the same features as those presented in Section 3.1.2, except that we extracted them from right to left. Since the set of characters is different, we designed another set of rules to be applied on Arabic characters. This resulted in a set of 75 rules we obtained after four hours. The decision trees thresholds were set up on a validation database, in a similar way to what is done for the Rimes and IAM databases. The recognition system obtained with this new set of rules also performed better than the noncontextual one. The combination of this system and both the context-independent and the hybrid ANN/HMM system excelled at the OpenHaRT 2010 competition [21].

7.3.5 Comparison to the State-of-the-Art

The complete results of the neural-network combination on the Rimes test set is presented in Table 7, along with the results of the other systems presented in the Rimes2009 Handwriting recognition competition [19]. The word recognition rate is calculated case insensitive and accent sensitive.

The UPV (Universidad Politecnica de Valencia) system is a voting-based combination of different classifiers [41]. The combined classifiers are three hybrid HMM/MLP systems. They use restricted vocabularies for decoding, based on image criteria (size, aspect ratio, etc.). The system presented by Siemens is based on the Hidden Markov Recognizer (HMR) for Latin script that is widely in use within Siemens AG for postal automation projects [42], [43]. The preprocessing is comparable to ours and the features extracted from sliding windows are structural. The system presented is also a combination of multiple recognizer outputs. The difference between the combined systems is the topology of the character HMMs and the sets of features extracted [44]. The Irisa system is based on continuous densities HMMs. The preprocessing used is standard; the features are extracted from sliding windows and depend on the image zones, defined by the baselines [45]. The number of states per character varies and the final results come from this stand-alone system. The Litis proposed a system based on a multistream segmentation-free HMM [46]. After a standard preprocessing step, two sets of features are extracted (contours based and density based) from two distinct sliding windows on each image. Each set produces an HMM system, giving a two-stream final model. The system presented by Itesoft is based on the Non-Symmetric Half-Plane Hidden Markov Model (NSHP-HMM) [47]. This model combines a Markov Random Field (MRF) and a HMM, and works directly on binary patterns. It can be noted that the system is inherently accent insensitive, which yields lower results for these recognition tasks.

TABLE 7
Comparison with the State-of-the-Art HMMs and Non-HMMs Systems on the Rimes 2009 Test Database (from [19])

system	word recognition rate			
	Test Lexicon		Full Lexicon	
	top1	top10	top1	top10
TUM	93.2%	99.0%	91.0%	98.3%
Proposed NN-combin.	**89.1%**	**98.7%**	**85.3%**	**97.4%**
UPV	86.1%	98.0%	83.2%	96.8%
SIEMENS	81.3%	96.4%	73.2%	93.4%
Previous system	80.2%	87.9%	76.3%	86.9%
IRISA	79.6%	92.8%	74.7%	90.3%
LITIS	74.1%	94.9%	66.7%	91.6%
ITESOFT	59.4%	76.7%	50.4%	72.9%

Results in Table 7 show that our proposed system performs better than the other HMM-based systems. The absolute increase in accuracy for top-1 results is higher than 3 and 2 percent compared to the UPV system and according to the size of the dictionary. In [19], the results of our previous system reached 80.2 percent on the small dictionary and 76.5 percent on the larger one. We now reduce the error rate of our system by nearly 9 percent. This gap is mainly due to the fact that the system we worked with in 2009 did not include accents. It is also due to better refinement in finding appropriate thresholds for tree splitting and enhancing the training procedure.

While our top10 accuracy is comparable, Table 7 indicates that recurrent neural networks (RNNs) can reach, on the same database and with the same protocol, remarkable top1 accuracy as high as 93.2 percent on the small dictionary. This could be due to several reasons. First, RNN training is discriminative while HMM training is generative (at least for the Dynamic Context-Dependent and Context-Independent systems which are GMM-based). Second, the feature extraction used by the RNN is adaptive (learned from the data) and might be better than the handcrafted ones that we use. And third, unlike the HMMs, where the probability of each observation depends only on the current state, RNNs do not make the assumption of independence of observations. Hence, we look forward to combining HMMs and RNNs in the future as it seems very promising.

8 Conclusion and Future Work

We introduced a novel approach to build efficient context-dependent word models based on the HMM framework. The key features of such approach are the use of dynamic features and state-based clustering. The clustering of trigraph states is performed at each state position and based

on binary trees. We proposed a set of expert-based questions on how characters are drawn and linked together. The context-dependent modeling proved to be more efficient on publicly available data. We also showed that context-dependent systems can be complementary to context-independent ones in a way that their combination performs better than other state-of-the-art HMM-based approaches.

Future work consists of improving the contextual-based system following two main directions: feature extraction and modeling. Additional features may complement the current 28-feature frames. One idea is to add features related to correlations between cells of different frames. Such correlations may exist in handwriting since strokes vary slowly. However, adding such features may require additional data frames to correctly estimate the additional parameters. Improved HMM models can also be built by estimating nondiagonal covariance matrices. Since too many parameters may be estimated, semi-tied covariance matrices may be obtained by applying a common transform to diagonal matrices. Other transforms can be searched to adapt the HMM models to a specific writer. In the case of the Rimes database, all words from a given letter document are written by the same writer. This could be performed in an unsupervised way when enough words from each writer are available. Finally, language models can efficiently improve HMM systems by correctly reranking the scored words provided by the recognition process. Language models such as bi-gram or tri-gram models are built from large corpora of text documents and can be used when words are provided within sentences.

REFERENCES

[1] R. Palacios, A. Gupta, and P. Wang, "Handwritten Bank Check Recognition of Courtesy Amounts," *Int'l J. Image and Graphics,* vol. 4, no. 2, pp. 203-222, 2004.

[2] W. Ding, C. Suen, and A. Krzyzak, "A New Courtesy Amount Recognition Module of a Check Reading System," *Proc. 19th Int'l Conf. Pattern Recognition,* pp. 1-4, 2008.

[3] C. Suen, L. Lam, D. Guillevic, N. Strathy, M. Cheriet, J. Said, and R. Fan, "Bank Check Processing System," *Int'l J. Imaging Systems and Technology,* vol. 7, no. 4, pp. 392-403, 1996.

[4] T. Paquet and Y. Lecourtier, "Automatic Reading of the Literal Amount of Bank Checks," *Machine Vision and Applications,* vol. 6, nos. 2/3, pp. 151-162, 1993.

[5] M. Wienecke, G.A. Fink, and G. Sagerer, "Toward Automatic Video-Based Whiteboard Reading," *Int'l J. Document Analysis and Recognition,* vol. 7, nos. 2/3, pp. 188-200, 2005.

[6] J.A. Rodriguez and F. Perronnin, "Local Gradient Histogram Features for Word Spotting in Unconstrained Handwritten Documents," *Proc. First Int'l Conf. Frontiers in Handwriting Recognition,* 2008.

[7] A. Vinciarelli and S. Bengio, "Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 6, pp. 709-720, June 2004.

[8] E. Kavallieratou, N. Fakotakis, and G.K. Kokkinakis, "An Unconstrained Handwriting Recognition System," *Int'l J. Document Analysis and Recognition,* vol. 4, no. 4, pp. 226-242, 2002.

[9] A. Vinciarelli and J. Luettin, "A New Normalization Technique for Cursive Handwritten Words," *Pattern Recognition Letters,* vol. 22, no. 9, pp. 1043-1050, 2001.

[10] A.H. Toselli, "Reconocimiento de Texto Manuscrito Continuo," PhD dissertation, Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia, 2004.

[11] R. Al-Hajj Mohamad, L. Likforman-Sulem, and C. Mokbel, "Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 7, pp. 1165-1177, July 2009.

[12] T. Plötz and G. Fink, "Markov Models for Offline Handwriting Recognition: A Survey," *Int'l J. Document Analysis and Recognition,* vol. 12, pp. 269-298, 2009.

[13] C. Sirat, "Handwriting and the Writing Hand," *Writing Systems and Cognition,* vol. 6, 1994.

[14] R. Plamondon and W. Guerfali, "The Generation of Rapid Human Movements I, a Delta-Lognormal Law," *Biological Cybernetics,* vol. 78, nos. 93/94, pp. 119-132, 1998.

[15] P. Natarajan, S. Saleem, R. Prasad, E. MacRostie, and K. Subramanian, "Multi-Lingual Offline Handwriting Recognition Using Hidden Markov Models: A Script-Independent Approach," *Proc. Summit on Arabic and Chinese Handwriting Recognition,* 2006.

[16] M. Schussler and H. Niemann, "A HMM-Based System for Recognition of Handwritten Address Words," *Proc. Int'l Workshop Frontiers in Handwriting Recognition,* pp. 505-514, 1998.

[17] G. Fink and T. Plotz, "On the Use of Context-Dependent Modeling Units for HMM-Based Offline Handwriting Recognition," *Proc. Int'l Conf. Document Analysis and Recognition,* vol. 2, pp. 729-733, 2007.

[18] R. El-Hajj, C. Mokbel, and L. Likforman-Sulem, "Recognition of Arabic Handwritten Words Using Contextual Character Models," *Proc. SPIE Document Recognition and Retrieval,* 2008.

[19] E. Grosicki and H. El-Abed, "ICDAR 2009 Handwriting Recognition Competition," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 1398-1402, 2009.

[20] U. Marti and H. Bunke, "The IAM-Database: An English Sentence Database for Off-Line Handwriting Recognition," *Int'l J. Document Analysis and Recognition.,* vol. 5, pp. 39-46, 2002.

[21] http://www.nist.gov/itl/iad/mig/hart2010.cfm, 2011.

[22] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependant Features and Hidden Markov Modeling," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 893-897, 2005.

[23] N. Otsu, "A Threshold Seclection Method from Gray-Level Histograms," *IEEE Systems, Man, and Cybernetics,* vol. 9, no. 1, pp. 62-66, Jan. 1979.

[24] K.-F. Lee, "Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition," *Readings in Speech Recognition,* pp. 347-366, Morgan Kaufmann Publishers, 1990.

[25] C. Chelba and R. Morton, "Mutual Information Phone Clustering for Decision Tree Induction," *Proc. Int'l Conf. Spoken Language Processing,* 2002.

[26] S.J. Young, J.J. Odell, and P.C. Woodland, "Tree-Based State Tying for High Accuracy Acoustic Modelling," *Proc. Workshop Human Language Technology,* pp. 307-312, 1994.

[27] H. Zen, K. Tokuda, and T. Kitamura, "Decision Tree Based Simultaneous Clustering of Phonetic Contexts, Dimensions, and State Positions for Acoustic Modeling," *Proc. European Conf. Speech Comm. and Technology,* pp. 3189-3192, 2003.

[28] http://www.telecom-paristech.fr/~lauli/LatinScriptRules/questionstree_accents.txt, 2011.

[29] A.F.R. Rahman and M.C. Fairhurst, "Multiple Classifier Decision Combination Strategies for Character Recognition: A Review," *Int'l J. Document Analysis and Recognition,* vol. 5, no. 4, pp. 166-194, 2003.

[30] J. Kittler, M. Hatef, R.P. Duin, and J. Matas, "On Combining Classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, pp. 226-239, Mar. 1998.

[31] R. Polikar, "Ensemble Based Systems in Decision Making," *IEEE Circuits and Systems Magazine,* vol. 6, no. 3, pp. 21-45, July-Sept. 2006.

[32] R. Al-Hajj, C. Mokbel, and L. Likforman-Sulem, "Combination of HMM-Based Classifiers for the Recognition of Arabic Handwritten Words," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 959-963, 2007.

[33] H. El-Abed and V. Märgner, "Reject Rules and Combination Methods to Improve Arabic Handwritten Word Recognizers," *Proc. Int'l Conf. Frontiers in Handwriting Recognition,* p. 6p, 2008.

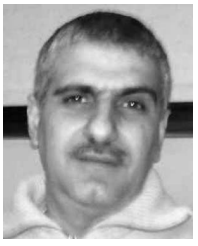
[34] V. Frinken, T. Peter, A. Fischer, H. Bunke, T.-M.-T. Do, and T. Artieres, "Improved Handwriting Recognition by Combining Two Forms of Hidden Markov Models and a Recurrent Neural Network," *Proc. 13th Int'l Conf. Computer Analysis of Images and Patterns,* pp. 189-196, 2009.

[35] E. Augustin, M. Carre, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Preteux, "Rimes Evaluation Campaign for Handwritten Mail Processing," *Proc. Int'l Workshop Frontiers in Handwriting Recognition,* pp. 231-235, 2006.
[36] http://rimes.it sudparis.eu/, 2010.
[37] A.-L. Bianne, C. Kermorvant, and L. Likforman-Sulem, "Context-Dependent HMM Modeling Using Tree-Based Clustering for the Recognition of Handwritten Words," *Proc. SPIE Document Recognition and Retrieval,* 2010.
[38] R.P.W. Duin, "The Combining Classifier: To Train or Not to Train?" *Proc. 16th Int'l Conf. Pattern Recognition,* vol. 2, pp. 765-770, 2002.
[39] M. Zimmermann and H. Bunke, "Automatic Segmentation of the IAM Off-Line Database for Handwritten English Text," *Proc. 16th Int'l Conf. Pattern Recognition,* vol. 4, pp. 35-39, 2002.
[40] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 5, pp. 855-868, May 2009.
[41] S. Espana-Boquera, M. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 33, no. 4, pp. 767-779, Apr. 2011.
[42] T. Caesar, J. Gloger, and E. Mandler, "Preprocessing and Feature Extraction for a Handwriting Recognition System," *Proc. Second Int'l Conf. Document Analysis and Recognition,* pp. 408-411, 1993.
[43] A. Kaltenmeier, T. Caesar, J. Gloger, and E. Mandler, "Sophisticated Topology of Hidden Markov Models for Cursive Script Recognition," *Proc. Second Int'l Conf. Document Analysis and Recognition,* pp. 139-142, 1993.
[44] M.-P. Schambach, "Model Length Adaptation of an HMM Based Cursive Word Recognition System," *Proc. Seventh Int'l Conf. Document Analysis and Recognition,* vol. 1, pp. 109-113, 2003.
[45] B. Issam, R. Schwartz, and J. Makhoul, "An Omnifont Open-Vocabulary OCR System for English and Arabic," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 6, pp. 495-504, June 1999.
[46] Y. Kessentini, T. Paquet, and A. Benhamadou, "A Multi-Stream Hmm-Based Approach for Off-Line Multi-Script Handwritten Word Recognition," *Proc. Int'l Conf. Frontiers in Handwriting Recognition,* 2008.
[47] C. Choisy, "Dynamic Handwritten Keyword Spotting Based on the NSHP-HMM," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 242-246, 2007.
[48] S.J. Young, G. Evermann, M.J.F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P.C. Woodland, "The HTK Book, Version 3.4," Cambridge Univ. Eng. Dept., 2006.

Anne-Laure Bianne-Bernard received the graduation degree from the French engineering school Télécom SudParis and the MSc degree with distinction in signal processing and communication from the University of Edinburgh, United Kingdom, in 2007. She started working toward the PhD degree in 2008 in handwriting recognition with Télécom ParisTech and A2iA SA, Paris, France. Her paper at the CIFED 2010 conference was among the best papers selection. Her research interests are in the area of handwriting recognition and machine learning.

Farès Menasri received the degree in engineering from the Ecole Française d'Electronique et d'Informatique (EFREI), Villejuif, France, in 2003, the master's degree in image processing and artificial intelligence from the Université Pierre & Marie Curie (IARFA), Paris, France, in 2004, and the PhD degree from the Université Paris Descartes, Paris, France, in 2008. He joined A2iA SA in 2003 as a research and development engineer. His research interests include document layout analysis, Latin and Arabic handwriting recognition, neurocomputing, and machine learning.

Rami Al-Hajj Mohamad received the BS degree in applied mathematics and the DEA degree in mathematical modeling and intensive computation from the Lebanese University, Beirut, in 1999 and 2001, respectively, and the PhD degree in signal and image processing from Télécom ParisTech, France, in 2007. He is currently an associate professor in the Department of Computer Sciences and Information Technology at the Lebanese International University, Beirut. His research interests include the automatic recognition of handwriting, pattern recognition, data mining, and artificial intelligence. He and two of the coauthors won first place at the ICDAR 05 Competition on Arabic handwritten word recognition in Seoul.



Chafic Mokbel received the degree in electronic engineering from the Lebanese University in 1988, the DEA (MSc) degree in electronic systems from the Institut National Polytechnique de Grenoble, France, in 1989, and the PhD degree in signal processing from Télécom ParisTech, France, in 1992. His PhD research focused on speech recognition in adverse environments. He then joined the Centre National des Etudes des Télécommunications (CNET) at France Telecom, where he participated in and led several projects on speech recognition systems. In 1999, he joined IDIAP, Switzerland, and in 2000, he joined Jinny Software, Dublin, Ireland, where he worked on the development of telecommunications services. Since 2001, he has been with the University of Balamand, Lebanon. As an associate professor, his research activities cover different domains of signal processing, statistical modeling, speech, language, handwriting, video processing, and biomedical imaging. He has developed several software toolkits like BECARS and HCM. From 1996 to 1998, he was an associate editor of *Speech Communication*. He is a senior member of the IEEE and of the IEEE Signal Processing Society.

Christopher Kermorvant received the engineering diploma from the Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), Evry, France, and the MSc degree in computer science from the University of Manchester, England, in 1996. From 1998 to 2000, he worked as a research assistant at the IDIAP institute, Martigny, Switzerland. He received the PhD degree in computer science from the Université Jean Monnet, Saint-Etienne, France, in 2003. He was a postdoctoral researcher at the Université de Montréal, Canada, until he joined A2iA, Paris, France, as a research engineer in 2005. He is now a research manager at A2iA. His current research interests are machine learning and statistical modeling applied to handwriting recognition, information extraction, and document classification.

Laurence Likforman-Sulem received the engineering degree from ENST-Bretagne (Ecole Nationale Supérieure des Télécommunications) in 1984, and the PhD degree from ENST-Paris in 1989. She is an associate professor at TELECOM ParisTech, where she serves as a senior instructor in pattern recognition and document analysis. She chaired the program committee of CIFED (Conference Internationale Francophone sur l'Ecrit et le Document) held in Fribourg, Switzerland, in 2006 and the program committees of two DRR Conferences (Document Recognition and Retrieval) held in 2009 and 2010 in San Jose, California. She is a senior member of the IEEE.