OFF-LINE HANDWRITTEN WORD RECOGNITION USING HIDDEN MARKOV MODELS

A. El-Yacoubi,[1,4] **R. Sabourin,**[1,2] **M. Gilloux**[3] and **C.Y. Suen**[1]

[1] Centre for Pattern Recognition and Machine Intelligence
Department of Computer Science, Concordia University
1455 de Maisonneuve Boulevard West
Suite GM-606, Montréal, Canada H3G 1M8

[2] Ecole de Technologie Supérieure
Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA)
1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3

[3] Service de Recherche Technique de La Poste
Département Reconnaissance, Modélisation Optimisation (RMO)
10, rue de l'île Mâbon, 44063 Nantes Cedex 02, France

[4] Departamento de Informatica (Computer Science Department)
Pontificia Universidade Catolica do Parana
Av. Imaculada Conceicao, 1155 - Prado Velho
80.215-901 Curitiba - PR - BRAZIL

This chapter describes a system that recognizes freely handwritten words off-line. Based on Hidden Markov models (HMMs), this system is designed in the context of a real application in which the vocabulary is large but dynamically limited. After preprocessing, a word image is segmented into letters or pseudo-letters and represented by two feature sequences of equal length, each consisting of an alternating sequence of shape-symbols and segmentation-symbols that are both explicitly modeled. The word model is made up of the concatenation of appropriate letter models which consist of elementary HMMs. Two mechanisms are considered to reject unreliable outputs, depending on whether or not the unknown word image is guaranteed to belong to the dynamic lexicon. Experiments performed on real data show that HMMs can be successfully used for handwriting recognition.

1 Introduction

Today, handwriting recognition is one of the most challenging tasks and exciting areas of research in computer science. Indeed, despite the growing interest in this field, no satisfactory solution is available. The difficulties encountered are numerous and include the huge variability of handwriting such as inter-writer and intra-writer variabilities, writing environment (pen, sheet, support, etc.), the overlap between characters, and the ambiguity that makes many characters unidentifiable without referring to context.

Owing to these difficulties, many researchers have integrated the lexicon as a constraint to build lexicon-driven strategies to decrease the problem complexity. For small lexicons, as in bank-check processing, most approaches are global and consider a word as an indivisible entity [1] - [5]. If the lexicon is large, as in postal applications (city name or street name recognition) [6] - [10], one cannot consider a word as one entity, because of the huge number of models which must be trained.

Therefore, a segmentation of words into basic units, such as letters, is required. Given that this operation is difficult, the most successful approaches are segmentation-recognition methods in which a loose segmentation of words into letters or pieces of letters is first performed, and the optimal combination of these units to retrieve the entire letters (definitive segmentation) is then obtained in recognition using *dynamic programming* techniques [11], [12], [13]. These methods are less efficient when the segmentation fails to split some letters. On the other hand, they have many advantages over global approaches. The first is that for a given learning set, it is more reliable to train a small set of units such as letters than whole words. Indeed, the frequency of each word is far lower than the frequency of each of its letters, which are shared by all the words of the lexicon. Furthermore, unlike analytic approaches, global approaches are possible only for lexicon-driven problems and do not satisfy the portability criterion, since for each new application, the set of words of the associated lexicon must be trained.

More recently, *hidden Markov models* (*HMMs*) [14], [15] have become the predominant approach to automatic speech recognition [16], [17], [18]. The main advantage of HMMs lies in their probabilistic nature, suitable for signals corrupted by noise such as speech or handwriting, and in their theoretical foundations, which are behind the existence of powerful algorithms to automatically and iteratively adjust the model parameters.

The success of HMMs in speech recognition has recently led many researchers to apply them to handwriting recognition, by representing each word image as a sequence of observations. According to the way this representation is carried out, two approaches can be distinguished: *implicit segmentation* [6], [19], [20], which leads to a speech-like representation of the handwritten word image, and *explicit segmentation* [7], [9] which requires a segmentation algorithm to split words into basic units such as letters.

In this chapter, we propose an explicit segmentation-based HMM approach to recognize unconstrained handwritten words (uppercase, cursive and mixed). An example of the kind of images to be processed is shown in Figure 1. This system uses *three* sets of features: the first two are related to the shape of the segmented units, while the features of the third set describe segmentation points between these units. The first set is based on global features such as loops, ascenders and descenders, while the second set is based on features obtained by an analysis of the bidimensional contour transition histogram of each segment. Finally, segmentation features correspond to either spaces, possibly occurring between letters or words, or to the vertical position of the segmentation points splitting connected letters. Given that the two sets of shape-features are separately extracted from the image, we represent each word by two feature sequences of equal length, each consisting of an alternating sequence of shape-symbols and segmentation-symbols.

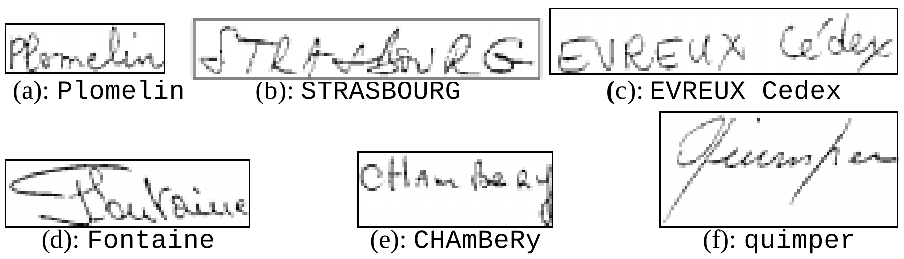

Figure 1. Examples of handwritten word images of city names in France.

In the problem we are dealing with, we consider a vocabulary which is large but dynamically limited. For example, in city name recognition, the contextual knowledge brought by the postal code identity can be used to reduce the lexicon of possible city names to a small size. However, since the entire lexicon is large, it is more realistic to model letters rather than whole words. Indeed, this technique needs only a reasonable number of models to train (and to store). Then each word (or word sequence) model can be built by concatenating letter models. This modeling is also more appropriate to available learning databases, which often do not contain all the possible words that need to be recognized.

This chapter is organized as follows. Section 2 describes the theory of HMMs, and particularly emphasizes on some variants that can enhance the standard modeling. Section 3 recalls the steps of preprocessing, segmentation and feature extraction. Section 4 deals with the application of HMMs to handwritten word recognition in a dynamic vocabulary. Section 5 presents the experiments performed to validate the approach. Section 6 concerns the rejection mechanism considered by our system. Finally, Section 7 presents some concluding remarks and perspectives.

2 Hidden Markov models

During the last 15 years, HMMs have been extensively applied in several areas including speech recognition [18], [21], [22], [23], language modeling [24], handwriting recognition [6], [9], [25], [26], on-line signature verification [27], human action learning [28], fault detection in dynamic systems [29] and recognition of moving light displays [30].

A hidden Markov model is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word *hidden*), but can be observed through another stochastic process that produces the sequence of observations [14]. The hidden process consists of a set of *states* connected to each other by *transitions* with probabilities, while the observed process consists of a set of *outputs* or *observations*, each of which may be emitted by each state according to some output probability density function (*pdf*). Depending on the nature of this *pdf*, several kinds of HMMs can be distinguished. If the observations are naturally discrete or quantized using *quantization* or *vector quantization* [31], and drawn from an *alphabet* or a *codebook*, the HMM is called *discrete* [16], [17]. If these observations are continuous, we are dealing with a *continuous* HMM [17], [32], with a continuous *pdf* usually approximated by a mixture of normal distributions. Another family of HMMs, a compromise between discrete and continuous HMMs, are *semi-continuous* HMMs [33] that mutually optimize the vector quantized codebook and HMM parameters under a unified probabilistic framework.

In some applications, it is more convenient to produce observations by transitions rather than by states. Furthermore, it is sometimes useful to allow transitions with no output in order to model, for instance, the absence of an event in a given stochastic process. If we add the possibility of using more than one feature set to describe the observations, we must modify the classic formal definition of HMMs [17]. For discrete HMMs, we can do this by considering the following parameters:

T: length of the observation sequence O; $O = O_0, O_1, ..., O_{T-1}$, where $O_t = (O_t^0, O_t^1, ..., O_t^p, ..., O_t^{P-1})$, the observation O_t^p at time t being drawn from the p^{th} finite feature set, and $p = 0, 1, ..., P - 1$.

N: number of states in the model.

M_p: number of possible observation symbols for the p^{th} feature set.

$S = \{s_0, s_1, ..., s_{N-1}\}$: set of possible states of the model.

$Q = \{q_t\}$, $t = 0, 1, ..., T - 1$; q_t: state of the process at time t.

$V_p = \{v_1^p, v_2^p, ..., v_M^p\}$ codebook or discrete set of possible observation symbols corresponding to the p^{th} feature set.

$A = \{a_{ij}\}$, $a_{ij} = Pr(q_{t+1}=s_j|q_t=s_i)$: probability of going from state s_i at time t to state s_j at time $t + 1$, and at the same time producing a real observation O_t at time t.

$A' = \{a'_{ij}\} = \{a'_{ij}\}$, $a'_{ij} = Pr(q_t=s_j|q_t=s_i)$: probability of null transition from state s_i at time t to state s_j at time t, producing null observation symbol Φ. Note here that there is no increase over time since no real observation is produced.

$B_p = \{b_{ij}^p(k)\}$, $b_{ij}^p(k) = \Pr(O_t^p = v_k^p \mid q_t=s_i, q_{t+1}=s_j)$: output *pdf* of observing the k^{th} symbol in the p^{th} feature set when a transition from state s_i at time t to state s_j at time $t + 1$ is taken. If we assume the P output *pdfs* are independent (multiple codebooks), we can compute the output probability $b_{ij}(k)$ as the product of the P output probabilities:

$$b_{ij}(k) = \prod_{p=0}^{P-1} b_{ij}^{p}(k) \tag{1}$$

$\Pi = \{\pi_i\}$, $\pi_i = Pr(q_1=s_i)$: initial state distribution. In general, it is more convenient to have predefined initial and final states s_0 and $s_{N\text{-}1}$ that do not change over time. In this case, $\pi_0 = 1$ and $\pi_i = 0$ for $i = 1, 2, \ldots N - 1$.

A, A' and B_p obey the stochastic constraints:

$$\sum_{j=0}^{N} [a_{ij} + a'_{ij}] = 1 \qquad \sum_{k=0}^{M_p-1} b_{ij}^{p}(k) = 1 \qquad p = 0, 1, \ldots, P-1 \tag{2}$$

Given a model, to be represented by the compact notation $\lambda = (A, A', B_p)$ where $p = 0, 1, \ldots, P - 1$, three problems must be solved.

1. Given an observation sequence $O = O_0, O_1, \ldots, O_{T\text{-}1}$ and a model λ, how do we compute the probability of O given λ, $Pr(O|\lambda)$? This is the *evaluation* problem.

2. Given the observation sequence $O = O_0, O_1, \ldots, O_{T\text{-}1}$ and the model λ, how do we find the optimal state sequence in λ that has generated O? This is the *decoding* problem.

3. Given a set of observation sequences and an initial model λ, how can we re-estimate the model parameters so as to increase the likelihood of generating this set of sequences? This is the *training* problem.

2.1. The evaluation problem

To compute $Pr(O|\lambda)$, we modify the well-known *forward-backward* procedure [17] to take into account the assumption that symbols are emitted along transitions, the possibility of null transitions, and the use of multiple codebooks. Hence, we define the *forward* probability $\alpha_t(i)$ as

$$\alpha_t(i) = Pr(O_0, O_1, \ldots, O_{t-1}, q_t = s_i | \lambda) \tag{3}$$

i.e., the probability of the partial observation sequence $O_0, O_1, \ldots, O_{t\text{-}1}$ (until time $t - 1$) and the state s_i reached at time t given the model λ. $\alpha_t(i)$ can be inductively computed as follows:

Initialization

$$\alpha_0(0) = 1.0$$
$$\alpha_0(j) = \sum_{i=0}^{N-1} a'_{ij}\alpha_0(i) \qquad j = 0, 1, \ldots, N-1 \tag{4}$$

given that s_0 is the only possible initial state.

Induction

$$\alpha_t(j) = \sum_{i=0}^{N-1}\left[a_{ij}\left(\prod_{p=0}^{P-1} b_{ij}^{p}(O_{t-1})\right)\alpha_{t-1}(i) + a'_{ij}\alpha_t(i)\right] \tag{5}$$
$$j = 0, 1, \ldots, N-1 \qquad t = 1, \ldots, T$$

by summing over all states that may lead to state s_j, and picking the appropriate time of a transition depending on whether we are dealing with a real observation or a null observation.

Termination

$$Pr(O|\lambda) = \alpha_T(N-1) \tag{6}$$

given that $s_{N\text{-}1}$ is the only possible terminal state. Similarly, we define the *backward* probability $\beta_t(i)$ by

$$\beta_t(i) = Pr(O_t, O_{t+1}, \ldots, O_{T-1}|q_t = s_i, \lambda) \tag{7}$$

i.e., the probability of the partial observation sequence from time t to the end, given state s_i reached at time t and the model λ. $\beta_t(i)$ can also be inductively computed as follows

Initialization

$$\beta_T(N-1) = 1.0$$
$$\beta_T(i) = \sum_{j=0}^{N-1} a'_{ij}\beta_T(j) \qquad i = 0, 1, \ldots, N-1 \tag{8}$$

given that $s_{N\text{-}1}$ is the only possible terminal state.

Induction

$$\beta_t(i) = \sum_{j=0}^{N-1}\left[a_{ij}\left(\prod_{p=0}^{P-1} b_{ij}^{p}(O_t)\right)\beta_{t+1}(j) + a'_{ij}\beta_t(j)\right] \tag{9}$$
$$i = 0, 1, \ldots, N-1 \qquad t = 0, \ldots, T-1$$

Termination

$$Pr(O|\lambda) = \beta_0(0) \tag{10}$$

given that s_0 is the only possible initial state.

2.2. The decoding problem

The decoding problem is solved using a near-optimal procedure, the *Viterbi* algorithm [34], [35], by looking for the best state sequence Q = (q_0, q_1,..., q_T) for the given observation sequence O = (O_0, O_1, ..., $O_{T\text{-}1}$). Again, we modify the classic algorithm [17] in the following way. Let

$$\delta_t(i) = \max_{q_0, q_1, \ldots, q_{t-1}} Pr(q_0, q_1, \ldots, q_t = s_i, O_0, O_1, \ldots, O_{t-1}|\lambda) \tag{11}$$

i.e., $\delta_t(i)$ is the probability of the best path that accounts for the first t observations and ends at state s_i at time t. We also define a function $\Psi_t(i)$, the goal of which is to recover the best state sequence by a procedure called *backtracking*. $\delta_t(i)$ and $\Psi_t(i)$ can be recursively computed in the following way:

Initialization

$$\begin{aligned}
&\delta_0(0) = 1.0\\
&\Psi_0(0) = 0\\
&\delta_0(j) = \max_{0 \le i \le N-1} a'_{ij}\delta_0(i) \qquad j = 0, 1, \ldots, N-1\\
&\Psi_0(j) = \operatorname{argmax}_{0 \le i \le N-1} a'_{ij}\delta_0(i) \qquad j = 0, 1, \ldots, N-1
\end{aligned} \tag{12}$$

given that s_0 is the only possible initial state.

Recursion

$$\delta_t(j) = \max_{0 \le i \le N-1}\left[a_{ij}\left(\prod_{p=0}^{P-1} b_{ij}^{p}(O_{t-1})\right)\delta_{t-1}(i); a'_{ij}\delta_t(i)\right] \tag{13}$$

$$\Psi_t(j) = \operatorname{argmax}_{0 \le i \le N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^{p}(O_{t-1}) \right) \delta_{t-1}(i); a'_{ij} \delta_t(i) \right] \tag{14}$$

$j = 0, 1, \ldots, N-1 \qquad t = 1, 2, \ldots, T$

Termination

$$P^* = \delta_T(N-1) \tag{15}$$

$$q_T^* = N-1 \tag{16}$$

given that $s_{N\text{-}1}$ is the only possible terminal state.

Path recovering: Backtracking procedure

$$q_t^* = \Psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \ldots, 0. \tag{17}$$

As shown above, except for the backtracking procedure, Viterbi and forward (Equations (4) - (6)) procedures are similar. The only difference is that the summation is replaced by a maximization.

2.3. The training problem

The main strength of HMMs is the existence of a procedure called the *Baum-Welch* algorithm [16], [17] that iteratively and automatically adjusts HMM parameters given a training set of observation sequences. This algorithm, which is an implementation of the *EM* (*expectation-maximization*) algorithm [36] in the HMM case, guarantees the model to converge to a local maximum of the probability of observation of the training set according to the *maximum likelihood estimation* (*MLE*) criterion. This maximum depends strongly on the initial HMM parameters.

To re-estimate HMM parameters, we first define $\xi_t^1(i,j)$, the probability of being in state s_i at time t and in state s_j at time $t + 1$, producing a real observation O_t given the model and the observation O, and $\xi_t^2(i,j)$, the probability of being in state s_i at time t and in state s_j at time t, producing the null observation Φ given the model and the observation O.

$$\xi_t^1(i,j) = Pr(q_t = s_i, q_{t+1} = s_j | O, \lambda) \tag{18}$$

$$\xi_t^2(i,j) = Pr(q_t = s_i, q_t = s_j | O, \lambda) \tag{19}$$

The development of these quantities leads to

$$\xi_t^1(i,j) = \frac{\alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t) \right) \beta_{t+1}(j)}{Pr(O|\lambda)} \tag{20}$$

$$\xi_t^2(i,j) = \frac{\alpha_t(i) a'_{ij} \beta_t(j)}{Pr(O|\lambda)} \tag{21}$$

We also define $\gamma_t(i)$ as the probability of being in state s_i at time t, given the observation sequence and the model.

$$\gamma_t(i) = Pr(q_t = s_i | O, \lambda) \tag{22}$$

$\gamma_t(i)$ is related to $\xi_t^1(i,j)$ and $\xi_t^2(i,j)$ by the following equation:

$$\gamma_t(i) = \sum_{j=0}^{N-1} [\xi_t^1(i,j) + \xi_t^2(i,j)] = \frac{\alpha_t(i)\beta_t(i)}{Pr(O|\lambda)} \tag{23}$$

The re-estimations of HMM parameters $\{a_{ij}\}$, $\{a'_{ij}\}$, $\{b_{ij}^p(k)\}$ are

$$\overline{a_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } t+1}{\text{expected number of being in } s_i} \tag{24}$$

$$\overline{a'_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ to } s_j \text{ and observing } \Phi}{\text{expected number of being in } s_i} \tag{25}$$

$$\overline{b_{ij}^p(k)} = \frac{\text{exp. num. of symbols } v_k^p \text{ in transition from } s_i \text{ at time } t \text{ to } s_j \text{ at time } t+1}{\text{exp. num. of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } t+1} \tag{26}$$

Given the definitions of $\xi_t^1(i,j)$, $\xi_t^2(i,j)$ and $\gamma_t(i)$, it is easy to see, when we are using one observation sequence O, that

$$\overline{a_{ij}} = \frac{\sum_{t=0}^{T} \xi_t^1(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t) \right) \beta_{t+1}(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{27}$$

$$\overline{a'_{ij}} = \frac{\sum_{t=0}^{T} \xi_t^2(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T} \alpha_t(i) a'_{ij} \beta_t(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{28}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{t=0}^{T} \delta(O_t^p, v_k^p)\xi_t^1(i,j)}{\sum_{t=0}^{T} \xi_t^1(i,j)} = \frac{\sum_{t=0}^{T} \delta(O_t^p, v_k^p)\alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t)\right) \beta_{t+1}(j)}{\sum_{t=0}^{T} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t)\right) \beta_{t+1}(j)} \tag{29}$$

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

For a set of training sequences $O(0)$, $O(1)$,..., $O(U-1)$ (size U), as is usually the case in real-world applications, the above formulas become

$$\overline{a_{ij}} = \frac{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \xi_t^1(i,j,u)}{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t^p(u))\right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{30}$$

$$\overline{a'_{ij}} = \frac{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \xi_t^2(i,j,u)}{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u) a'_{ij} \beta_t(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{31}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \delta(O_t^p(u), v_k^p)\xi_t^1(i,j,u)}{\sum_{u=0}^{U-1}\sum_{t=0}^{T} \xi_t^1(i,j,u)}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \delta(O_t^p(u), v_k^p)\alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t^p(u))\right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(O_t^p(u))\right) \beta_{t+1}(j,u)} \tag{32}$$

In the above equations, the index u is introduced into α, β, ξ^1, ξ^2 and γ to indicate the observation sequence $O(u)$ currently used. Note that a new quantity $P(u) = Pr(O(u)|\lambda)$ appears, since this term is now included in the summation and cannot be eliminated as before. Training can also be performed using the *segmental k-means* algorithm or Viterbi training [37]. The idea behind this algorithm is that after initializing the model parameters with random values, each word is matched against its associated feature sequence via the Viterbi algorithm. According to the current model, observation sequences of the training set are segmented into states (or transitions) by recovering the optimal alignment path. The re-estimations of the new HMM parameters are then directly obtained by examining the number of transitions between states and the number of observations emitted along transitions. This procedure is repeated (as in Baum-Welch training) until the increase in the global probability of observing training examples falls below a small fixed threshold. Although this algorithm is less optimal than the Baum-Welch algorithm, it generally leads to good results and is faster in computation.

3 Representation of Word Images

In Markovian modeling, each input word image must be represented as a sequence of observations, which should be statistically independent, once the underlying hidden state is known. To fulfill the latter requirement, the word image is first preprocessed by 4 modules: *baseline slant normalization*, *lower case letter area normalization* when dealing with cursive words, *character skew correction*, and finally, *smoothing*.

Indeed, beside the fact that these variabilities are not meaningful to recognition and cause a high writer-sensitivity in classification, thus increasing the complexity in a writer-independent handwriting recognizer, they can introduce dependence between observations. For instance, a word with a highly slanted baseline is likely to give rise to many segments (after the segmentation process) with incorrectly detected descenders. In the absence of the baseline slant, none of these descenders will be detected, hence the idea of dependence between observations when the writing baseline is not normalized. The same thought can be made about the character slant. After preprocessing, we perform segmentation and feature extraction processes to transform the input image into an ordered sequence of symbols (first assumption).

3.1. Preprocessing

In our system, the preprocessing consists of four steps: *baseline slant normalization*, *lower case letter area* (*upper-baseline*) *normalization* when dealing with cursive words, *character skew correction*, and finally, *smoothing*. The goal of the first two is to ensure a robust detection of ascenders and descenders in our *first* feature set. The third step is required since the *second* feature set shows a significant sensitivity to character slant (Section 3.3). Baseline slant normalization is performed by aligning the minima of the lower contour after having removed those corresponding to descenders and those generated by pen-down movements, using the *least square method* and some thresholding techniques. Upper-baseline normalization is similar, and consists of aligning the maxima of the upper contour after having filtered those corresponding to ascenders or uppercase letters. However, the transformation here is non-linear since it must keep the normalized lower-baseline horizontal. The ratio of the number of filtered maxima over the total number of maxima is used as an *a priori* selector of the writing style: either cursive or mixed if this ratio is above a given threshold (fixed at 0.4 after several trials) or uppercase, in which case no normalization is done. Character skew is estimated as the average slant of elementary segments obtained by sampling the contour of the word image, without taking into account the horizontal and pseudo-horizontal segments. Finally, we carry out a smoothing to eliminate noise appearing at the borders of the word image, and resulting from the application of the continuous transformations associated to the above preprocessing techniques in a discrete space (bitmap). Figure 2 shows an example of the steps of preprocessing. More details on the description of these techniques can be found in [38].

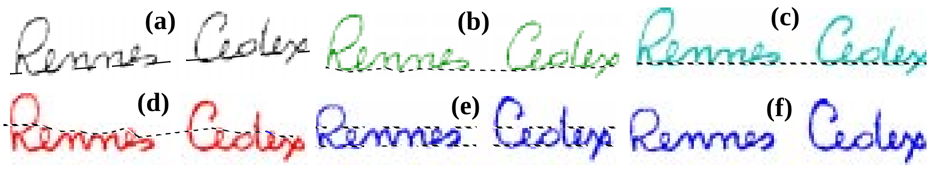

Figure 2. Preprocessing steps applied to word images: **(a)** original image, **(b)** and **(c)** baseline slant normalization, **(d)** character slant normalization, **(e)** lower-case letter area normalization, **(f)** definitive image after smoothing.

3.2. Character segmentation of words

As mentioned earlier, segmentation of words into smaller units is necessary when dealing with large vocabularies. Segmentation techniques used in the framework of HMM-based handwritten word recognition approaches can be divided into *implicit* and *explicit* methods. Implicit methods are inspired by those considered in speech recognition which consist of sampling the speech signal into successive frames with a frequency sufficiently large to separately detect the different phonetic events (for instance, *phonemes*) using minimal supervised learning techniques [16], [39]. In handwriting, they can either work at the pixel column level [6], [20] or perform an *a priori* scanning of the image with sliding windows [19], [40]. Explicit methods, on the other hand, try to find explicitly the segmentation points in a word by using some characteristic points such as upper (or lower) contour minima, intersection points, or spaces. Implicit methods are better than explicit methods in splitting touching characters, for which it is hard to find regularly explicit segmentation points. However, due to the bidimensional nature of off-line handwritten word images, and to the overlap between letters, implicit methods are less efficient here than in speech recognition or on-line handwriting recognition. Indeed, vertical sampling makes it difficult to capture the sequential aspect of the strokes, which is better represented by explicit methods. Moreover, in implicit methods, segmentation points have to be also learned.

On the other hand, when employing explicit methods, the basic units to be segmented are naturally the alphabet letters. Unfortunately, because of the ambiguity encountered in handwritten words, it is impossible to correctly segment a word into characters without resorting to the recognition phase. Indeed, the same pixel representation may lead to several interpretations, in the absence of the context which can be a lexicon or grammatical constraints. In Figure 3, for instance, the group of letters inside the dashed square could be interpreted - in the absence of context given by the word Strasbourg - as "lreur", "lrun", "bour" (correct spelling), "baun", etc.

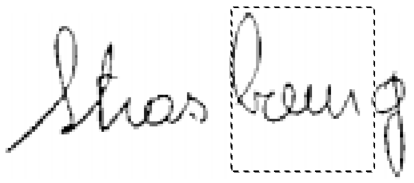

Figure 3. Ambiguity in handwritten words.

In view of the above remark, our concern is to design a segmentation process that tries to detect all the potential segmentation points, instead of only the real ones. This gives rise to several segmentation options, with the optimal one to be implicitly recovered during recognition. Integrating the above ideas, our segmentation algorithm is based on the following two hypotheses:

- There exist natural segmentation points corresponding to disconnected letters.
- The physical segmentation points between connected letters are located at the neighborhood of the image upper contour minima.

The segmentation algorithm makes use of upper and lower contours, loops, and upper contour minima. Then, to generate a segmentation point, a minimum must be located at the neighborhood of an upper-contour point that permits a vertical transition from the upper contour to the lower one without crossing any loop, while minimizing the vertical transition histogram of the word image. This strategy leads to a correct segmentation of a letter, to an undersegmentation of a letter (letter omission), or to an oversegmentation in which case a letter is split into more than one piece. Figure 4 gives an example of the behavior of the segmentation algorithm.

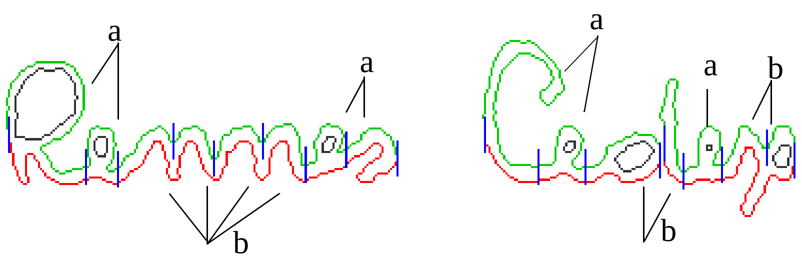

Figure 4. Segmentation of words into letters (a) or pseudo-letters (b).

3.3. Feature Extraction

The aim of the feature extraction phase is to extract in an ordered way, suitable to Markovian modeling, a set of relevant features that reduce redundancy in the word image, while preserving the discriminative information for recognition. Unlike in speech recognition where the commonly used features are obtained using well defined physical and mathematical concepts such as linear predictive coding (*LPC*) [17], there is no such agreement about the optimal features to be extracted from handwritten words. This is why topological features, features based on the pixel level, on the distribution of black pixels and on global transformations (*Fourier*, *Walsh*, *Karhunen-Loeve*, etc.) are often used in handwritten word recognition. Our main philosophy in this phase is that lexicon-driven word recognition approaches do not require features to be very discriminative at the segment (grapheme) level, because other information such as context (particular letter ordering in lexicon words, nature of the segmentation points) and word length, are available and permit high discrimination of words. Thus, our idea is to consider features at the grapheme level with the aim of clustering letters into classes.

Given our segmentation algorithm, a grapheme may consist of a full character, a piece of a character or more than a character. Such features cannot capture fine details of the segments, but this allows on the other hand a description of the segments with less variability, ensuring a better learning of the distribution of the features over the characters.

In our system, the sequence of segments obtained by the segmentation process is transformed into a sequence of symbols by considering two sets of features. The first set [F_1 in Figure 6] is based on global features such as loops, ascenders and descenders. Ascenders (descenders) are encoded in two ways according to their relative size compared to the height of the upper (lower) writing zone. Loops are encoded in various ways according to their membership in each of the three writing zones (upper, lower, median), and their relative size compared to the sizes of these zones. The horizontal order of the median loop and the ascender (or descender) within a segment is also taken into account to ensure a better discrimination between letters such as "b" and "d" or "p" and "q".

This encoding scheme can be described simply by representing a segment by a binary vector, the components of which indicate the presence or the absence of the characteristics mentioned above. Each combination of these features within a segment is encoded by a distinct symbol. For example, in Figure 6, the first segment is encoded by symbol "*L*", reflecting the existence of a large ascender and a loop located above the core region. The second segment is encoded by symbol "*o*", indicating the presence of a small loop within the core region. The third segment is represented by symbol "-", which encodes shapes without any interesting feature. This scheme leads to an alphabet of 27 symbols.

The second feature set [F_2 in Figure 6] is based on the analysis of the bidimensional contour transition histogram of each segment in the horizontal and vertical directions. After a filtering phase, the histogram values may be 2, 4 or 6. We focus only on the median part of the histogram, which represents the stable area of the segment. In each direction, we determine the dominant transition number (2, 4 or 6). Each different pair of dominant transition numbers is then encoded by a different symbol or class. This coding leads to 3 x 3 = 9 symbols. In Figure 5, for instance, letters "B", "C" and "O", whose pairs of dominant transition numbers are (6,2), (4,2) and (4,4), are encoded by symbols called "*B*", "*C*" and "*O*", respectively. In order to distinguish between the letters "A", "D", "O" and "P", ideally encoded by the same symbol "*A*" (4,4), we added new symbols by a finer analysis of the segments. The subclass ("O", "D") is chosen if the number of points of the lower contour located on the baseline and included in the median (or stable) zone of the segment is greater than a threshold, depending on the width of this zone. The subclass "P" is detected if the dominant number of transitions in the horizontal direction (4, in this case) loses its dominant character when focusing on the lower part of the segment. Similar techniques are used to discriminate between other letter classes, leading to a final feature set of 14 symbols.

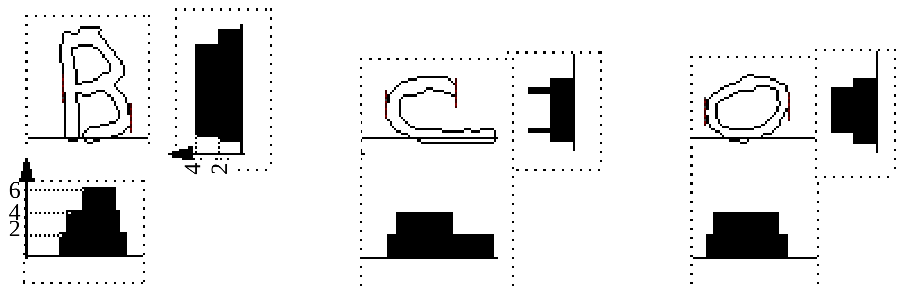

Figure 5. Transition histograms of segmented shapes.

In addition to the two feature sets describing segmented shapes, we also use segmentation features that try to reflect the way segments are linked together. These features consist of three categories. For not connected segments, two configurations are distinguished: if the space width is less than a third of the average segment width (*ASW*), we decide that there is no space and encode this configuration by the symbol "*n*"; otherwise, we validate the space and we encode it in two ways ("@" or "#"), depending on whether the space width is larger or smaller than *ASW*.

For connected segments, the considered feature is the segmentation point vertical position. This feature is encoded in two ways (symbols "*s*" or "*u*") depending on whether the segmentation point is close to or far from the writing baseline. Hence, we obtain 5 segmentation features.

Space features have been considered in order to increase the discrimination between cursive and uppercase letters, while the vertical position of the segmentation or over-segmentation point is taken into account to discriminate between pairs of letters such as ("a","o"), ("u","v"), ("m","w"), ("H","U").

Finally, given that the two sets of shape-features are extracted independently, the feature extraction process represents each word image by two symbolic descriptions of equal length, each consisting of an alternating sequence of symbols encoding a segment shape and of symbols encoding the segmentation point associated with this shape (Figure 6).

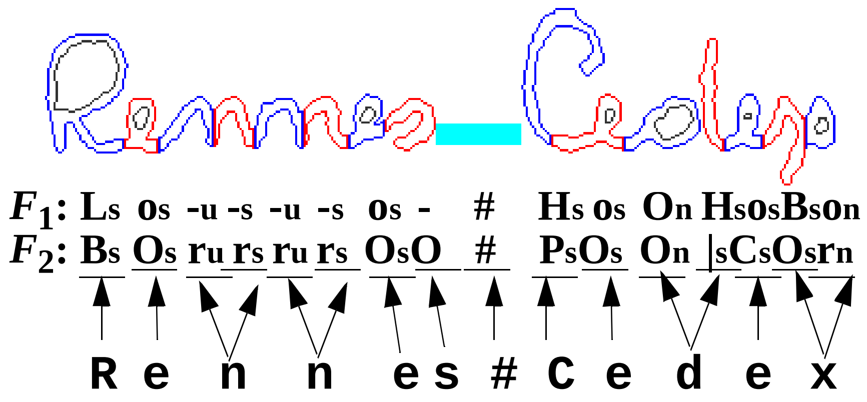
F1: Ls Os -u -s -u -s Os - # Hs Os On HsOsBsOn
F2: Bs Os ru rs ru rs OsO # PsOs On |sCsOsrn
R e n n es # C e d e x

Figure 6. Pair of feature sequences representing a word (or sequence of words) image.

4 Markovian Modeling of Handwritten Words

This section addresses the application of HMMs in handwritten word recognition. We begin first by briefly describing some related works in this field. Then, we give the justifications behind the design of the model we propose, and we detail the steps of learning and recognition as used in our system.

4.1. HMM use in handwritten word recognition

Recently, HMMs have been applied to several areas in handwriting recognition, including noisy printed text recognition [41], isolated character recognition [42], [43], on-line word recognition [44], [45], [46] and off-line word recognition. In the last application, several approaches have been proposed.

Gillies [6] is one of the first to propose an implicit segmentation-based HMM for cursive word recognition. First, a label is given to each pixel in the image according to its membership in strokes, holes and concavities located above, within and below the core region. Then, the image is transformed into a sequence of symbols which result from a vector quantization of each pixel column. Each letter is characterized using a different discrete HMM, the parameters of which are estimated on training data corresponding to hand-segmented letters. The Viterbi algorithm is used in recognition and allows an implicit segmentation of words into letters as a by-product of the word-matching process.

Bunke et al. [26] propose an HMM approach to recognize cursive words produced by cooperative writers. The features used in their scheme are based on the edges of the word skeleton graph. A semi-continuous HMM is considered for each character, with a number of states corresponding to the minimum number of edges expected of this character. The number of codebook symbols (the number of gaussians) was defined by manual inspection of the data, and recognition is performed using a *beam* search-driven Viterbi algorithm.

Chen et al. [47] use an explicit segmentation-based continuous density variable duration HMM for unconstrained handwritten word recognition. In this approach, observations are based on geometrical and topological features, pixel distributions, etc. Each letter is identified with a state which can account for up to 4 segments per letter. The statistics of the HMM (transition, observation and state duration probabilities) are estimated using the lexicon and the manually labeled training data. A modified Viterbi algorithm is applied to provide several outputs, which are post-processed using a general string editing method.

Cho et al. [40] use an implicit segmentation-based HMM to model cursive words. The word image is first split into a sequence of overlapping vertical grayscale bitmap frames, which are then encoded into discrete symbols using *principal component analysis* and vector quantization. A word is modeled by an interconnection network of character and ligature HMMs, with a number of states depending on the average sequence length of corresponding training samples. A clustering of ligature samples is performed to reduce the number of ligature models so as to ensure a reliable training. To improve the recognition strategy, several combinations of Forward and Backward Viterbi are investigated.

Finally, Mohamed and Gader [20] use an implicit segmentation-based continuous HMM for unconstrained handwritten word recognition. In their approach, observations are based on the location of black-white and white-black transitions on each image column. A 12-state left-to-right HMM is designed for each character. The training of the models is carried out on hand-segmented data, where the character boundaries are manually identified inside word images.

4.2. The proposed model

As shown above, several HMM architectures can be considered for handwritten word recognition. This stems from the fact that the correct HMM architecture is actually not known. The usual solution to overcome this problem is to first make structural assumptions, and then use parameter estimation to improve the probability of generating the training data by the models. In our case, the assumptions to be made are related to the behavior of the segmentation and feature extraction processes. As our segmentation process may produce a correct segmentation of a letter, a letter omission, or an oversegmentation of a letter into two or three segments, we built an eight-state HMM having three paths to take these configurations into account (Figure 7). In this model, observations are emitted along transitions. Transition t_{07}, emitting the null symbol Φ, models the letter omission case. Transition t_{06} emits a symbol encoding a correctly segmented letter shape, while transition t_{67} emits a symbol encoding the nature of the segmentation point associated with this shape. Null transition t_{36} models the case of oversegmentation into only 2 segments. Transitions t_{01}, t_{23} and t_{56} are associated with the shapes of the first, second and third parts of an oversegmented letter, while t_{12} and t_{45} model the nature of the segmentation points that gave rise to this oversegmentation.

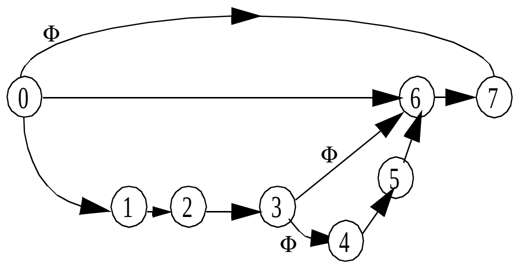

Figure 7. The character model.

This architecture is somewhat similar to that of other approaches such as [4], [47], but with some differences. In our method, the first segment presented to a character model is produced by two different transitions depending on whether it corresponds to the entire shape of a correctly segmented character (t_{06}) or to the first part of an oversegmented character (t_{01}), while in [4], [47] for example, the same transition is shared between these two configurations. The architecture proposed here allows the transitions of the model to be fed by homogeneous data sources, leading to less variability and higher accuracy (for example, the first part of an oversegmented "d" and a correctly segmented "d", which are very different, would be presented to different kinds of transitions (t_{01} and t_{06}, respectively). In other words, the variability coming from the inhomogeneity in the source data, since it is known *a priori*, is eliminated by separate modeling of the two data sources. We should also add that as each segment is represented in the feature extraction phase by two symbols related to our two feature sets, two symbols are independently emitted along transitions modeling segment shapes (t_{06}, t_{01}, t_{23} and t_{56}). In addition, we have a special model for inter-word space, in the case where the input image contains more than one word (Figure 1).

This model simply consists of two states linked by two transitions, modeling a space (in which case only the symbols corresponding to spaces "@" or "#" can be emitted) or no space between a pair of words (Figure 8).

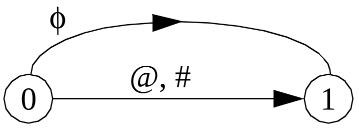

Figure 8. The inter-word space model.

4.3. The learning phase

The goal of the learning phase is to estimate the best parameter values of the character models, given a set of training examples and their associated word labels. Since the exact orthographic transcription (labeling) of each training word image is available, the word model is made up of the concatenation of the appropriate letter models; the final state of an HMM becomes the initial state of the next one, and so on (Figure 9).

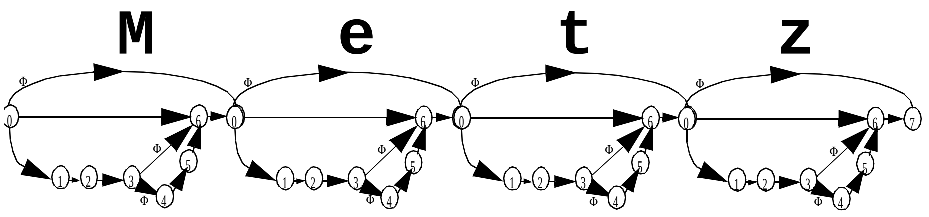

Figure 9. Training model for the French word `Metz`.

The training is performed using the variant of the Baum-Welch procedure described in Section 2. Note here that we consider a *minimum supervised* training (given the exact transcription of words) in which the units (segments) produced by the segmentation algorithm need not be manually labeled by their associated letters or pseudo-letters. This is an important consideration for two reasons: first, labeling a database is a time-consuming and very expensive process, and is, therefore, not desirable; second, supervised training allows the recognizer to capture contextual effects, and permits segmentation of the sequence of units into letters and re-estimation of the transitions associated with these units to optimize the likelihood of the training database. Thus, the recognizer decides for itself what the optimal segmentation might be, rather than being heavily constrained by *a priori* knowledge based on human intervention [39]. This is particularly true if we bear in mind the inherent incorrect assumptions made about the HMM structure. From an implementation point of view, given a word composed of L letters, a new parameter corresponding to the index of the currently processed letter is added to the quantities involved in the Baum-Welch algorithm. Then, the results of the final forward (initial backward) probabilities at the last (initial) state of the elementary HMM associated with a letter are moved forward (backward) to become the initial forward (final backward) probabilities at the initial (last) state of the elementary HMM associated with the following (previous) letter. If $\alpha_t^l(i)$ (or $\beta_t^l(i)$) denotes the standard forward (or backward) probability associated with the letter of index l, then this process is carried out according to the following equations:

$$\alpha_t^{l+1}(0) = \alpha_t^l(N-1) \qquad l = 0, \ldots, L-2 \qquad t = 0, 1, \ldots, T-1 \tag{33}$$

$$\beta_t^{l-1}(N-1) = \beta_t^l(0) \qquad l = 1, \ldots, L-1 \qquad t = 0, 1, \ldots, T-1 \tag{34}$$

s_0 and $s_{N\text{-}1}$ being the initial and final states of elementary HMMs associated with letters. Similar modifications can be made to $\delta_t(i)$ if we want to use Viterbi training.

We should also add that the analysis of the segmentation process shows that splitting a character into three pieces is a rather rare phenomenon. Thus, the associated parameters are not likely to be reliably estimated, due to the lack of training examples that exhibit the desired events. The solution to this problem is to share the transitions involved in the modeling of this phenomenon (t_{34}, t_{36}, t_{45}, t_{56}) over all character models, by calling for the *tied states* principle. Two states are said to be tied when there exists an equivalence relationship between them (transitions leaving each of these two states are analogous and have equal probabilities) [16]. Nevertheless, this procedure is not carried out for letters M, W, m or w for which the probability of segmentation into 3 segments is high, and therefore, there are enough examples to train separately the parameters corresponding to the third segment for each of these letters. A further improvement consisted of considering *context-dependent* models for uppercase letters depending on their position in the word: first position whether in an uppercase or cursive word, or any different position in an uppercase word. The motivation behind this is that features extracted from these two categories of letters can be very different, since they are based on global features such as ascenders which strongly depend on the writing style by way of the writing baselines.

In addition to the learning set, we use a validation set in training on which the re-estimated model is tested after each iteration of the training algorithm. At the end of training, reached when the increase in the probability of generating the learning set by the models falls below a given threshold, the stored HMM parameters correspond to those obtained at the iteration, maximizing the likelihood of generating the validation set (and not the learning set) by the models. This strategy allows the models to acquire a better generalization over unknown samples.

4.4. The recognition phase

The task of the recognition problem is to find the word *w* (or word sequence) maximizing the *a posteriori* probability that *w* has generated an unknown observation sequence *O*:

$$Pr(\hat{w}|O) = \max_w Pr(w|O) \tag{35}$$

Applying Bayes' rule to this definition, we obtain the fundamental equation of pattern recognition,

$$Pr(w|O) = \frac{Pr(O|w)Pr(w)}{Pr(O)} \tag{36}$$

Since *Pr*(*O*) does not depend on *w*, the decoding problem becomes equivalent to maximizing the joint probability,

$$Pr(w, O) = Pr(O|w)Pr(w) \tag{37}$$

Pr(*w*) is the *a priori* probability of the word *w* and is directly related to the language of the considered task. For large vocabularies, *Pr*(*w*) is very difficult to estimate due to the lack of sufficient training samples. The estimation of *Pr*(*O*|*w*) requires a probabilistic model that accounts for the shape variations *O* of a handwritten word *w*. We assume that such a model consists of a global Markov model created by concatenating letter HMMs. The architecture of this model remains basically the same as in training. However, as no information in recognition is available on the style (orthographic transcription) in which an unknown word has been written, a letter model here actually consists of two models in parallel, associated with the upper and lower case modes of writing a letter (Figure 10). As a matter of fact, an initial state (*I*) and a final state (*F*) are considered, and two consecutive letter models are now linked by four transitions associated with the various ways two consecutive letters may be written: uppercase-uppercase (*UU*), uppercase-lowercase (*UL*), lowercase-uppercase *(LU*) and lowercase-lowercase (*LL*). The probabilities of these transitions are estimated by their occurrence frequency from the same learning database which served for HMM parameter estimation. The probabilities of beginning a word by an uppercase (0*U*) or lower-case letter (0*L*) are also estimated in the same way.

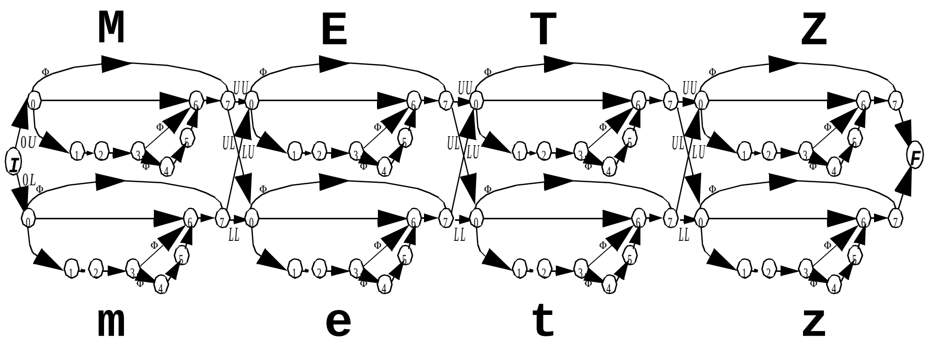

Figure 10. Global recognition model for lexicon word METZ.

This architecture is more efficient than usually adopted methods which generate *a priori* two or three possible ASCII configurations of words (fully uppercase, fully lower-case or lower-case word beginning with an uppercase letter). Indeed, these methods quickly become tedious and time consuming when dealing with a word sequence rather than a single word, besides the fact that they cannot handle the problem of mixed handwritten words (e.g. Figure 1e). The proposed model elegantly avoids these problems, while the computation time increases only linearly. Recognition is performed using the variant of the Viterbi algorithm described in Section 2, allowing an implicit detection during recognition of the writing style which can be recovered by the backtracking procedure.

5 Experiments

Experiments were carried out on unconstrained handwritten French city name images located manually on real mail envelopes. The sizes of the learning, validation and test databases were 11,106, 3,610 and 4,280, respectively. To simulate the address recognition task, we assume that the city names in competition are independent for a given list of corresponding postal code hypotheses. Under this assumption, for each image in the test set, we choose $N - 1$ city names from a vocabulary of 9,313 city names. The prior probabilities were assumed to be equal, so that the average perplexity was maximum and equal to N. In our tests, the values chosen for N were 10, 100 and 1,000. These values have a physical meaning since they simulate the case where 1, 2 or 3 digits in the postal code are ambiguous (1, 2 or 3 ambiguous digits give rise to 10, 100 or 1,000 possible postal codes). Recognition was carried out using the logarithmic version of the Viterbi procedure described in Section 2, while for training we used the Baum-Welch algorithm. Results of the tests are reported in Table 1, in which *RR(k)* corresponds to the proportion of the correct answers among the *k* best solutions provided by the recognition module. Figure 11 shows some well-recognized images by our approach.

TABLE 1. Recognition rates obtained on various lexicon sizes.

Lexicon	*RR(1)*	*RR(2)*	*RR(3)*	*RR(4)*	*RR(5)*
10	99.0%	99.8%	99.9%	99.9%	100.0%
100	96.2%	98.2%	98.8%	99.2%	99.4%
1,000	88.5%	93.4%	94.8%	95.8%	96.4%

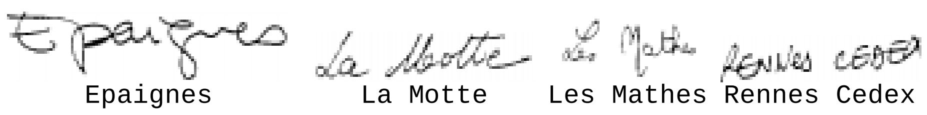

Figure 11. Some examples of well-recognized images.

The results shown in Table 1 prove that HMMs can be successfully used for designing a high-performance handwritten word recognition system. As mentioned in Section 3.3, even though the features used are not very discriminative at the grapheme level and do not capture fine details of the letters, the association (sequence) of these features to describe words in an ordered way is very discriminative, thanks to redundancy. This discrimination is as high as the length of the feature sequence extracted from the unknown word image. Furthermore, the description of the graphemes with features which do not take into account details, make these features more visible in the training set, thus ensuring a reliable estimation of the probability of observation of these features. Preprocessing and segmentation features also have been proven to significantly contribute to recognition accuracy [38]. It is not obvious to compare our approach with other works since we are not using the same databases, and also because our system recognizes actual French city names which may consist of one or several words. However, our results seem to compare favorably with others in the literature [40], [47]. Confusions in our system come mainly from poor images (Figure 12a), words with overlapping and touching characters (Figure 12b), words with truncated characters (Figure 12c), images with underline or with line above them (Figure 12d), or the lack of examples to reliably estimate some model parameters.

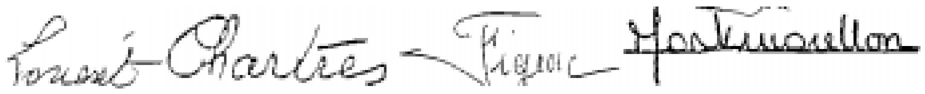
(a) `Lorient` (b) `Chartres` (c) `Figeac` (d) `Montmorillon`

Figure 12. Some examples of misrecognized images.

6 Rejection

Usually, systems designed for real tasks are required to have *confusion rates* (*CR*) lower than some threshold depending on economical criteria. A typical value of accepted *CR* in postal applications is 1%, while a much lower value is required in the case of bank check processing. Therefore, it is necessary to consider in our approach a *rejection* criterion. In this perspective, we must go back to the Bayes formula in equation (36) to compute *Pr*(*O*). When *O* is known to belong to the lexicon, as in our previous experiments, *Pr*(*O*) can be obtained simply by

$$Pr(O) = \sum_{w} Pr(O|w) \times Pr(w) \qquad (38)$$

Then rejection can be established by requiring *Pr*(*w*|*O*) to be greater than a given threshold. Table 2 shows, when considering a dynamic lexicon of size 100, the evolution of the recognition rate and reliability (defined as the proportion of correct answers among the accepted images) as a function of the rejection rate by varying the threshold value, when the correct answer is guaranteed to belong to the lexicon.

TABLE 2. Recognition rate (*RC*) and reliability (*RL*) as a function of rejection rate (*RJ*) when the word image is guaranteed to belong to the lexicon.

RJ	0.0	1.9	4.3	7.0	7.7	8.6	9.0	9.8	10.9
RC	96.2	95.4	94.0	92.0	91.5	90.8	90.4	89.8	88.8
RL	96.2	97.2	98.2	98.9	99.1	99.3	99.4	99.5	99.7

In real applications, however, the processed word image is not guaranteed to belong to the lexicon, since it can be the result of a city name mislocation or a wrong dynamic generation of the lexicon (due to an important error in postal code recognition). To cope with this problem, we have randomly generated lexicons, *half* of which do not contain the correct answers. In this case, we express Bayes probability as proposed in [9] by

$$Pr(w|O) = \frac{p_{in} \times Pr(O|w) \times Pr(w)}{p_{in} \times \sum_{w} Pr(O|w) \times Pr(w) + p_{out} \times Pr(O|out)} \qquad (39)$$

where p_{in} = 0.5 is the *a priori* probability that w belongs to the lexicon and p_{out} is its complement; $p_{out} = 1 - p_{in}$. We approximated the term $Pr(O|out)$ by the output of an ergodic HMM trained using the Baum-Welch algorithm on the same set used to train the character models (although a more accurate set should also have included words that do not correspond to city names). The number of states of this HMM was set to 14 after several trials. Table 3 shows, for a dynamic lexicon of size 100, the evolution of the recognition rate and reliability as a function of the rejection rate, when the correct answer is not guaranteed to belong to the lexicon. Note that in this experiment, the recognition rate cannot exceed 50%, since p_{in} = 0.5.

TABLE 3. Recognition rate (*RC*) and reliability (*RL*) as a function of rejection rate (*RJ*) when the word image is not guaranteed to belong to the lexicon.

RJ	0.0	24.9	29.0	34.2	41.4	48.2	51.6	53.6	57.3
RC	48.0	47.5	47.1	46.7	45.8	44.0	42.8	42.0	39.6
RL	48.0	63.2	66.4	70.9	78.2	84.9	88.6	90.6	92.8

7 Summary

In this chapter, we described a complete system designed to recognize unconstrained handwritten words. The results obtained show that our approach achieves good performance, given that the data come from real-word images and that the writers were not aware that their words were to be processed by computer. One of the main strengths of our system lies in its training phase, which does not require any manual segmentation of the data. Due to the large size of the vocabulary, our Markovian modeling is carried out at the character level. Character HMMs model not only segmented shapes, but also segmentation points, leading to better discrimination between letters. By building the word model as a sequence of character models, each consisting of a pair of associated uppercase and lower-case HMMs, the writing style is implicitly detected during recognition. An error analysis shows that our system can still be improved in most of its components. The segmentation algorithm should be optimized so as to be able to systematically split all the characters, particularly the overlapping and touching characters. Indeed, it is better to have pieces of characters which can be gathered during recognition, than segments containing more than one character. We also need more relevant features, since uppercase letters, lower-case letters, numerals and other special characters may be encountered in free-handwriting. This increases the level of ambiguity of the shapes generated by our segmentation algorithm, and therefore, a high level of description of these shapes is required. Moreover, given that our model assumes the independency between different feature sets (see Section 2), adding new feature sets does not increase the complexity from the training point of view and increases only linearly the required amount of memory to store the parameter values. A solution to avoid the inherent loss of information when generating our codebooks or feature sets is to replace discrete HMMs by semi-continuous HMMs. This can be particularly beneficial for our histogram-based features and segmentation features.

Acknowledgments

This work was supported by the Service de Recherche Technique de La Poste (SRTP) at Nantes, France, l'École de Technologie Supérieure (ETS) at Montréal, Canada, and the Centre of Pattern Recognition and Machine Intelligence (CENPARMI) at Montréal, Canada.

References

[1] M. Gilloux and M. Leroux (1992), "Recognition of cursive script amounts on postal cheques," *Proceedings of the 5th U.S. Postal Service Advanced Technology Conference*, pp. 545-556.

[2] T. Paquet and Y. Lecourtier (1993), "Recognition of Handwritten Sentences Using A Restricted Lexicon," *Pattern Recognition*, Vol. 26, No. 3, pp. 391-407.

[3] N.D. Gorski (1994), "Experiments with Handwriting Recognition using Holographic Representation of Line Images," *Pattern Recognition Letters*, Vol. 15, No. 9, pp. 853-859.

[4] S. Knerr, O. Baret, D. Price, and J.C. Simon (1996), "The A2iA Recognition System for Handwritten Checks," *Proceedings of Document Analysis Systems*, pp. 431-494.

[5] C.Y. Suen, L. Lam, D. Guillevic, N.W. Strathy, M. Cheriet, J.N. Said, and R. Fan (1996), "Bank Check Processing System," *International Journal of Imaging Systems and Technology*, Vol. 7, pp. 392-403.

[6] A.M. Gillies (1992), "Cursive Word Recognition Using Hidden Markov Models," *Proceedings of the 5th U.S. Postal Service Advanced Technology Conference*, pp. 557-562.

[7] M.Y. Chen, A. Kundu, and J. Zhou (1994), "Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 5, pp. 481-496.

[8] E. Cohen, J.J. Hull, and S.N. Srihari (1994), "Control Structure for Interpreting Handwritten Addresses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 10, pp. 1049-1055.

[9] M. Gilloux, M. Leroux, and J.M. Bertille (1995), "Strategies for Cursive Script Recognition Using Hidden Markov Models," *Machine Vision and Applications*, Vol. 8, No. 4, pp. 197-205.

[10] A. El-Yacoubi, J.M. Bertille, and M. Gilloux (1995), "Conjoined Location and Recognition of Street Names Within a Postal Address Delivery Line," *International Conference on Document Analysis and Recognition*, Vol. 2, pp. 1024-1027.

[11] R.M. Bozinovic and S.N. Srihari (1989), "Off-Line Cursive Word Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No.1, pp. 68-83.

[12] J.T. Favata and S.N. Srihari (1992), "Cursive Word Recognition Using Hidden Markov Models," *Proceedings of the 5th U.S. Postal Service Advanced Technology Conference*, pp. 237-251.

[13] F. Kimura, M. Shridhar, and Z. Chen (1993), "Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words," *International Conference on Document Analysis and Recognition*, pp. 18-22.

[14] L.R. Rabiner and B.H. Juang (1986), "An Introduction to Hidden Markov Models," *IEEE Signal Processing Magazine*, Vol. 3, pp. 4-16.

[15] A.B. Poritz (1988), "Hidden Markov Models: A Guided Tour," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 7-13.

[16] L. Bahl, F. Jelinek, and R. Mercer (1983), "A Maximum Likelihood Approach to Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 179-190.

[17] L.R. Rabiner (1989), "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286.

[18] K.F. Lee, H.W. Hon, M.Y. Hwang, and X. Huang (1990), "Speech Recognition Using Hidden Markov Models: A CMU Perspective," *Speech Communication* 9, Elsevier Science Publishers B.V., North-Holland, pp. 497-508.

[19] T. Caesar, J.M. Gloger, A. Kaltenmeier, and E. Mandler (1993), "Recognition of Handwritten Word Images by Statistical Methods," *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*, pp.409-416.

[20] M. Mohamed and P. Gader (1996), "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 5, pp. 548-554.

[21] L.R. Rabiner (1989), "High Performance Connected Digit Recognition Using Markov Models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 8, pp. 1214-1224.

[22] A. Averbuch, L. Bahl, R. Bakis, P. Brown, G. Daggett, S. Das, K. Davies, S. De Gennaro, P. V. De Souza, E. Epstein, D. Fraleigh, F. Jelinek, B. Lewis, R. Mercer, J. Moorhead, A. Nadas, D. Nahamoo, M. Picheny, G. Shichman, P. Spinelli, D. Van Compernolle and H. Wilkens (1987), "Experiments with the Tangora 20,000 word speech recognizer," *IEEE International Conference on Acoustics, Signal and Speech Processing*, pp. 701-704.

[23] Y.L. Chow, M.O. Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, S. Roucos, and R.M. Schwartz (1987), "BIBLOS: The BBN Continuous Speech Recognition System," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 89-92.

[24] F. Jelinek, R.L. Mercer, and S. Roukos (1992), "Principles of Lexical Language Modeling for Speech Recognition," *Advances in Speech Signal Processing*, edited by Sadaoki Furui and M. Mohan Sondhi, pp. 651-699.

[25] A. Kundu, Y. He, and P. Bahl (1989), "Recognition of Handwritten Word: First and Second Order Hidden Markov Model Based Approach," *Patter Recognition*, Vol. 22, No. 3, pp. 283-297.

[26] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini (1995), "Off-Line Cursive Handwriting Recognition using Hidden Markov Models," *Pattern Recognition*, Vol.28, No.9, pp. 1399-1413.

[27] L. Yang, B.K. Widjaja, and R. Prasad (1995), "Application of Hidden Markov Models for Signature Verification," *Pattern Recognition*, Vol. 28, No. 2, pp. 161-170.

[28] J. Yang, Y. Xu, and S. Chen (1997), "Human Action Learning via Hidden Markov Models," *IEEE Transactions on Systems, Man, and Cybernetics*-Part A: Systems and Humans, Vol. 27, No. 1, pp. 34-44.

[29] P. Smyth (1994), "Hidden Markov Models for Fault Detection in Dynamic Systems," *Pattern Recognition*, Vol. 27, No. 1, pp. 149-164.

[30] K.H. Fielding and D.W. Ruck (1995), "Recognition of Moving Light Displays Using Hidden Markov Models," *Pattern Recognition*, Vol. 28, No. 9, pp. 1415-1421.

[31] R.M. Gray and Y. Lind (1982), "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," *IEEE Transactions on Communications*, Vol. COM-30, No. 2, pp. 381-389.

[32] L.A. Liporace (1982), "Maximum likelihood estimation for multivariate observation of Markov sources," *IEEE Transactions on Information Theory*, Vol. IT-28, No. 5, pp. 729-734.

[33] X.D. Huang and M.A. Jack (1989), "Semi-Continuous Hidden Markov Models for Speech Signals," *Computer Speech and Language*, Vol. 3, pp. 239-251, 1989.

[34] G.D. Forney (1973), "'The Viterbi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, pp. 268-278.

[35] H.L. Lou (1995), "Implementing the Viterbi Algorithm," *IEEE Signal Processing Magazine*, pp. 42-52.

[36] T.K. Moon (1996), "The Expectation-Maximization Algorithm," *IEEE Signal Processing magazine*, pp. 47-60.

[37] L.R. Rabiner and B.H. Juang (1993), "Fundamentals of Speech Recognition," Englewood Cliffs, NJ: Prentice Hall, pp. 382-384.

[38] A. El-Yacoubi, J.M. Bertille, and M. Gilloux (1994), "Towards a more effective handwritten word recognition system," *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition*, pp. 378-385.

[39] J. Picone (1990), "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Signal Processing Magazine*, pp. 26-41.

[40] W. Cho, S.W. Lee, and J.H. Kim (1995), "Modeling and Recognition of Cursive Words with Hidden Markov Models," *Pattern Recognition*, Vol. 28, No. 12, pp.1941-1953.

[41] A.J. Elms and J. Illingworth (1998), "The Recognition of Noisy Polyfont Printed Text Using Combined HMMs," *International Journal on Document Analysis and Recognition (IJDAR)*, Vo. 1, No. 1, pp. 3-17.

[42] R. Nag, K.H. Wong, and F. Fallside (1986), "Script Recognition Using Hidden Markov Models," *IEEE International Conference on Acoustics Signal and Speech Processing*, pp.2071-2074.

[43] H.J. Kim, K.H. Kim, S.K. Kim, and J.K. Lee (1997), "On-Line Recognition of Handwritten Chinese Characters Based on Hidden Markov Models," *Pattern Recognition*, Vol. 30, No. 9, pp. 1489-1500.

[44] S. Bercu and G. Lorette (1993), "On-Line Handwritten Word Recognition: An Approach Based on Hidden Markov Models," *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*, pp. 385-390.

[45] J.Y. Ha, S.C. Oh, and J.H. Kim (1995), "Recognition of Unconstrained Handwritten English Words With Character and Ligature Modeling," *International Journal on Pattern Recognition and Artificial Intelligence*, Vol. 9, No. 3, pp. 535-556.

[46] J. Hu, M.K. Brown, and W. Turin (1996), "HMM Based On-Line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No.10, pp. 1039-1045.

[47] M.Y. Chen, A. Kundu, and S.N. Shrihari (1995), "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Transactions on Image Processing*, Vol. 4, No. 12, pp. 1675-1687.

[48] A. El-Yacoubi, R. Sabourin, M. Gilloux and C.Y. Suen (1998), "Improved Model Architecture and Training Phase in an Off-line HMM-based Word Recognition System," *Proceedings of the 14th International Conference on Pattern Recognition*, pp. 1521-1525.