Arabic Handwritten Word Recognition based on Bernoulli Mixture HMMs

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

Máster en:

Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Ihab Alkhoury

Supervised by:
Alfons Juan-Císcar

Valencia, España. December 2010

Acknowledgements

It is a great pleasure to thank everyone who helped me to complete my master thesis successfully and moreover on time. First, I would like to thank my professors form Bethlehem University, specially Dr. Saleem. and Dr. Suhail for there support in my undergraduate project who encouraged me to continue my master studies.

I am equally grate-full to my adviser, Alfons Juan. He gave me moral support, guided and suggested me the outlines of this project and corrected my doubts. Also a special thanks to my colleagues in the laboratory who always helped me, and specially to Adrian Giménez who never stopped supporting me, and answering my questions.

Last but not the least, I would like to thank my family, who always stayed by my side and encouraged me all the time. I send them all my regards. Of Course I wont forget my friends, in Spain and also in Palestine.

Thanking you,

Ihab Al-Khoury
Valencia, November 30, 2010

Contents

Abstract

This thesis presents new approaches in off-line Arabic Handwriting Recognition based on conventional Bernoulli Hidden Markov models. Until now, the off-line handwriting recognition, in particular, the Arabic handwriting recognition is still far away form being perfect. Hidden Markov Models (HMMs) are now widely used for off-line handwriting recognition in many languages and, in particular, in Arabic. As in speech recognition, they are usually built from shared, embedded HMMs at symbol level, in which state-conditional probability density functions are modeled with Gaussian mixtures. In contrast to speech recognition, however, it is unclear which kind of features should be used and, indeed, very different features sets are in use today. Among them, we have recently proposed to simply use columns of raw, binary image pixels, which are directly fed into embedded Bernoulli (mixture) HMMs, that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. The idea is to by-pass feature extraction and ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In this thesis, we review this idea along with some extensions that are currently providing state-of-the-art results on Arabic handwritten word recognition.

1

Introduction

1.1 Arabic Handwriting Recognition

Computer recognition of characters or word is one of the most successful applications in pattern recognition. A handwriting recognizer is a system that automatically transcribes text image into text. Put it simply, it is an automated process that uses pattern recognition (PR) and machine learning (ML) techniques to recognize characters or words given a lexicon or even an entire dictionary [27].

Fueled by market demand, some Optical Character Recognition (OCR) systems, and also some handwriting systems are now available as commercial products. Despite all these, off-line handwriting recognition is still a major challenge in PR [27, 25]. Offline handwriting recognition is the task of determining what letters or word are present in digital image on handwritten text[15]. The earliest work on handwriting recognition (HR) was carried out in the sixties and seventies. Due to the poor performance achieved by these systems at that time, less research on handwriting recognition took place during the eighties [14]. The ultimate goal of HR is to have systems able to understand any handwriting text. Their training phase should be minimum to automatically adapt themselves to a new user.

The recognition of Arabic handwriting presents unique challenges and benefits and has been approached more recently than the recognition of text in other scripts. Arabic is spoken by 234 million people and important in the culture of many more [15]. It is one of the six United Nations official languages [6, 5, 3, 1]. The characters of Arabic script and similar characters are used by a much higher percentage of the worlds population to write languages such as Arabic, Farsi (Persian), and Urdu. Arabic script differs from Latin scripts in several ways. Unlike English handwriting, Arabic is written from right-to-left and does not distinct between upper or lowercase characters [20].

1.2 Statistical Handwriting Recognition

To discuss the problem of handwriting recognizer design, we need its mathematical formulation based on probabilities. Let $O = \mathbf{o}_1, \ldots, \mathbf{o}_T$ denote an observation sequence of fixes-dimension feature vectors, and $W = \mathbf{w}_1, \ldots, \mathbf{w}_N$ denote the set of possible transcriptions, each belonging to a fixed and known vocabulary. If $p(W \mid O)$ denotes the probability that the transcriptions W is written in the text image, given that the evidence O was observed, then the recognizer will choose the most likely transcription given the observed evidence. In another words, the recognizer should decide in favor of a transcription (word) w^* satisfying,

$$w^* = \arg\max_{w \in W} \; p(W \mid O) \tag{1.1}$$

The well known Bayes formula of probability theory allows us to rewrite the right side of the equation, moreover, since the maximization in equation 1.1 is carried out with observation O fixed, it follows that the recognizer is trying to find the word w^* that maximizes the product $p(W)\, p(O \mid W)$, that is,

$$w^* = \arg\max_{w \in W} \; p(W)\, p(O \mid W)\,, \tag{1.2}$$

where $p(W)$ is the probability that the word W was written, it is usually approximated by an *n-gram language model* [11], and $p(O \mid W)$ is the probability that when the image contains the word W the evidence O will be observed, in another words it is a *text image model*

Following equation 1.2, a handwriting recognition system consists of four main components which are described in detail in the following sections, preprocessing and feature extraction, a text image model, a language model, and a search module.

1.2.1 Preprocessing and Feature Extraction

It is important to know what a data O will be observed. In handwriting of text image case, image has to be transformed into sequence of fixed-dimension feature vectors with which the recognizer will deal. For this, some techniques may be applied on original images depending on its clarity, such as, rescaling, thresholding and background removal, skew correction, binarization, and many others. For details see to [22]. Feature extraction is not only limited for one bit column feature extraction. However a sliding window of fixed width could be centered at each column of the image. Moreover, this sliding window also could be then translated to align its center with its mass center (Repositioning). The binary image under the translated window is read to construct a local binary feature vector and, in this way, the whole input image is transformed into a sequence of fixed-dimension feature vectors. More information in section 3.2

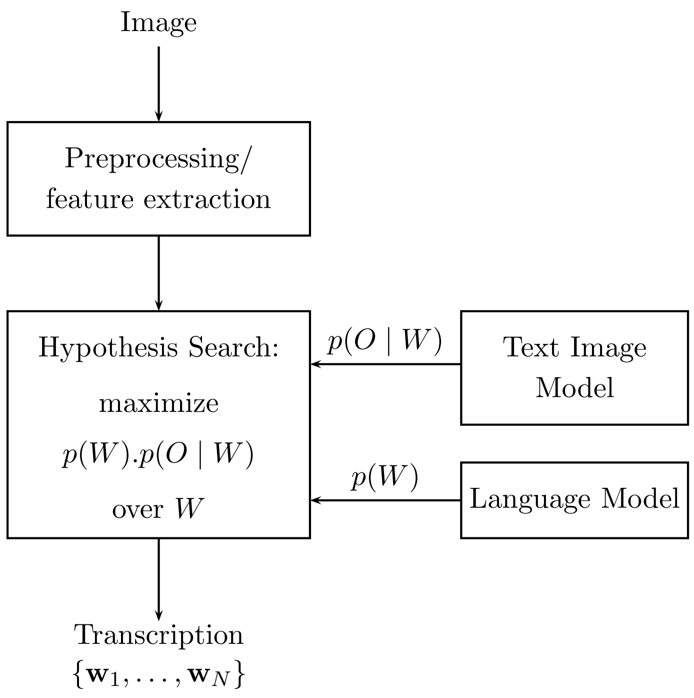


Figure 1.1: Basic architecture of a statistical handwriting recognition system

1.2.2 Text Image Modeling

Returning back to the equation 1.2, the recognizer needs to determine the probability value $p(O \mid W)$ for the realization of a sequence of feature vectors O given a word sequence W. Having in mind that, writings differ significantly from writer to another depending on the writing conditions. To model the writing variations, hidden Markov models (HMM) are used. They have been established as a de-facto standard for speech recognition systems [24], however they are widely used for off-line handwriting recognition in many languages and, in particular, in Arabic [16, 17]. Other models are also used such as those based on artificial neural networks or dynamic time wrapping [12]. In this thesis, we use the Bernoulli (mixture) HMMs (BHMMs), that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. Our models are explained in details in chapter 2.

1.2.3 Language Modeling

Language models (LMs) are used to model text properties like syntax and semantic independently from the morphological models. They are used in many natural language processing applications such as speech recognition, machine translation or handwritten recognition. These models try to capture the properties of a language, and are used to predict the next word in a word sequence [10].

In another words, equation 1.2, requires that we be able to compute for every word W the a priori probability $p(W)$. Bayes formula allows many decompositions,

$$p(W) = p(w_1).\prod_{t=2}^{T} p(w_t \mid w_1^{t-1})\,, \tag{1.3}$$

where $p(w_t \mid w_1^{t-1})$ is the probability of the word w_t when we have already seen the sequence of words $w_1 \ldots w_{t-1}$. The sequence of words prior to w_t is called history. The recognizer must estimate the value of the probability $p(w_t \mid w_1^{t-1})$. In fact estimating this value is difficult and costly since sentences can be arbitarily long. For this reason these models are often approximated using smoothed n-gram models, which obtains surprisingly good performance [10], although they only capture short term dependencies.

The n-gram model is nowadays the most wide-spread model used for language modeling. Besides from being used in HTR, it is also used in machine translation, speech recognition, and practically in all human language technologies [11]. In a n-gram model, the probability $p(w_1, \ldots, w_T)$ of observing the sentence $w_1, \ldots, w_T$ is approximated as,

$$p(W) \approx p(w_1).\prod_{t=2}^{T} p(w_t \mid w_{t-(n-1)}^{t-1})\,, \tag{1.4}$$

where the probability of observing the t^{th} word w_t in the context history of the preceding $t-1$ words can be approximated by the probability of observing it in the shortened context history of the preceding $n-1$ words (n^{th} order Markov property).

The parameter estimation can be easily carried out from a training set using the Maximum Likelihood Estimation (MLE), since no hidden variables are required in n-gram model. The conditional probability can be calculated from n-gram frequency counts,

$$p(w_t \mid w_{t-(n-1)}^{t-1}) = \frac{count(w_{t-(n-1)}, \ldots, w_{t-1}, w_t)}{count(w_{t-(n-1), \ldots, w_{t-1}})} \tag{1.5}$$

As it is known, however, the n-gram probabilities are not derived directly from the frequency counts, this estimation gives a zero probability for all unseen events. This problem is solved by smoothing the model, that is, modifying the original probability distribution in order to obtain similar distribution but without zero probabilities. Various methods are used, from simple "add-one" smoothing (assign a count of 1 to unseen n-grams) to more sophisticated models, such as Good-Turing discounting or back-off models.

1.2.4 Hypothesis Search

Keeping the equation 1.2 in mind, in order to find the desired transcription w^* of the observation O, we must search over all possible words W to find the maximizing one. This search cannot be conducted by brute force since the space of words W is very large. However a method called the *Viterbi algorithm* is used that will not consider the overwhelming number of possible candidates of word W. This method, like all the known others for a large vocabulary, cannot actually be guaranteed to find the most likely W. It gives very good results, however, from the practical point of view.

1.3 Scientific Goals and Document Structure

The main objective of this thesis is to put forward the theoretical framework of the Bernoulli Hidden Markov model, as well as to discuss new approaches in the conventional handwriting recognition system, specially applied to Arabic. In chapter 6, we will see that our objectives has been fulfilled to a great extent that are currently providing state-of-the-art results on Arabic handwritten word recognition. In particular, Our Arabic handwriting recognition system won the award of the first Place Prize in the ICFHR 2010 competition. This thesis must be read in a sequential order.

In chapter 2, we introduce our basic Bernoulli Hidden Markov model applied to the well known IfN/ENIT database of Arabic handwriting Tunisian town names [23], BHMM-based handwriting recognition, maximum likelihood parameter estimation, and empirical results. In chapter 3, our basic extension to plain BHMMs, which will be referred to as *windowed BHMMs*, and much more improved results. In chapter 4, window repositioning is described, and new empirical results. To our knowledge, they are the best results published to date on the IfN/ENIT database. In chapter 5, repositioning in windowed BHMM is applied to the OpenHaRT database [21], and empirical results are shown. Concluding remarks are discusses in chapter 6. A short overview of Arabic Handwriting is given in the Appendix (chapter 7).

2

Bernoulli Hidden Markov Models

2.1 Introduction

Hidden Markov Models (HMMs) are now widely used for off-line handwriting recognition in many languages and, in particular, in Arabic [16, 17]. Arabic is spoken by 234 million people and important in the culture of many more [?]. Given a text (line or word) image, it is first transformed into a sequence of fixed-dimension feature vectors, and then fed into an HMM-based decoder to find on its most probable transcription. Following the conventional approach in speech recognition [24], HMMs at global (line or word) level are built from shared, *embedded* HMMs at character (subword) level, which are usually simple in terms of number of states and topology. In the common case of real-valued feature vectors, state-conditional probability (density) functions are modeled as Gaussian mixtures since, as with finite mixture models in general, their complexity can be easily adjusted to the available training data by simply varying the number of components.

After decades of research in speech recognition, the use of certain real-valued speech features and embedded Gaussian (mixture) HMMs is a de-facto standard [24]. However, in the case of handwriting recognition, there is no such a standard and, indeed, very different sets of features are in use today. In [8], we proposed to by-pass feature extraction and to directly feed columns of raw, binary pixels into *embedded Bernoulli (mixture) HMMs (BHMMs),* that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. The basic idea is to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In this chapter, we discuss this basic idea, where our experiments were carried out on the very popular IfN/ENIT database of Arabic handwritten Tunisian town names [23]. In particular, we describe the plain Bernoulli mixtures and Bernoulli mixture HMMs with examples (Sec. 2.2 and 2.3), BHMM-based Handwriting Recognition, and forward and backward algorithms (Sec. 2.4), maximum likelihood estimation (Sec. 2.5). Then, empirical results are reported in Section 2.6. Concluding remarks are given in Section 2.7.

2.2 Bernoulli Mixture

Let $\mathbf{o}$ be a D-dimensional feature vector. A finite mixture is a probability (density) function of the form:

$$P(\mathbf{o} \mid \mathbf{\Theta}) = \sum_{k=1}^{K} \pi_k \, P(\mathbf{o} \mid k, \mathbf{\Theta}') \ , \tag{2.1}$$

where K is the number of mixture components, π_k is the kth component coefficient, and $P(\mathbf{o} \mid k, \mathbf{\Theta}')$ is the kth component-conditional probability (density) function. The mixture is controlled by a parameter vector $\mathbf{\Theta}$ comprising the mixture coefficients and a parameter vector for the components, $\mathbf{\Theta}'$. It can be seen as a generative model that first selects the kth component with probability π_k and then generates $\mathbf{o}$ in accordance with $P(\mathbf{o} \mid k, \mathbf{\Theta}')$.

A Bernoulli mixture model is a particular case of (2.1) in which each component k has a D-dimensional Bernoulli probability function governed by its own vector of parameters or *prototype* $\mathbf{p}_k = (p_{k1}, \ldots, p_{kD})^t \in [0,1]^D$,

$$P(\mathbf{o} \mid k, \mathbf{\Theta}') = \prod_{d=1}^{D} p_{kd}^{o_d} (1 - p_{kd})^{1-o_d} \ , \tag{2.2}$$

where p_{kd} is the probability for bit d to be 1. Note that this equation is just the product of independent, unidimensional Bernoulli probability functions. Therefore, for a fixed k, it can not capture any kind of dependencies or correlations between individual bits.

Consider the example given in Figure 2.1. Three binary images (**a, b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black=1, white=0, gray=0.5). The prototype has been obtained by averaging images **a** and **c,** and it is the best approximate solution to assign a high, equal probability to these

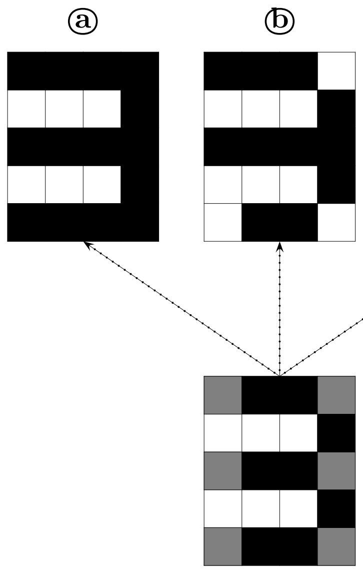


Figure 2.1: Three binary images (**a, b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a gray image (black=1, white=0, gray=0.5).

images. However, as individual pixel probabilities are not conditioned to other pixel values, there are $2^6 = 64$ different binary images (including **a, b** and **c**) into which the whole probability mass is uniformly distributed. It is then not possible, using a single Bernoulli prototype, to assign a probability of 0.5 to **a** and **c,** and null probability to any other image such as **b.** Nevertheless, this limitation can be easily overcome by using a Bernoulli mixture and allowing a different prototype to each different image shape. That is, in our example, a two-component mixture of equal coefficients, and prototypes **a** and **b,** does the job.

2.3 Bernoulli HMM

Let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of feature vectors. An HMM is a probability (density) function of the form:

$$P(O \mid \boldsymbol{\Theta}) = \sum_{q_0,\ldots,q_{T+1}} \prod_{t=0}^{T} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t)\,, \tag{2.3}$$

where the sum is over all possible *paths* (state sequences) $q_0, \ldots, q_{T+1}$, such that $q_0 = I$ (special *initial* or *start* state), $q_{T+1} = F$ (special *final* or *stop* state), and $q_1, \ldots, q_T \in \{1, \ldots, M\}$, being M the number of regular (non-special) states of the HMM. On the other hand, for any regular states i and j, a_{ij} denotes the *transition* probability from i to j, while b_j is the *observation* probability (density) function at j.

A Bernoulli (mixture) HMM (BHMM) is an HMM in which the probability of observing $\mathbf{o}_t$, when $q_t = j$, is given by a Bernoulli mixture probability function for the state j:

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} \pi_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} (1 - p_{jkd})^{1-o_{td}} \,, \tag{2.4}$$

where π_{jk} and $\mathbf{p}_{jk}$ are, respectively, the prior and prototype of the kth mixture component in state j.

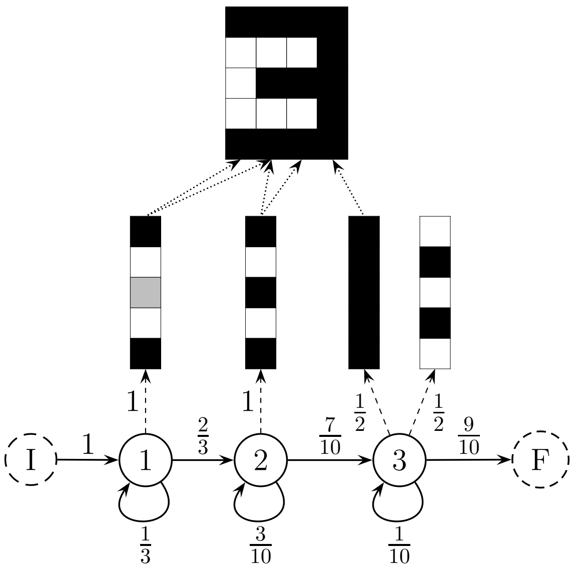
Consider the upper part of Figure 2.2, where a BHMM example for the number 3 is shown, together with a binary image generated from it. It is a three-state model with single prototypes attached to states 1 and 2, and a two-component mixture assigned to state 3. In contrast to the example in Figure 2.1, prototypes do not account for the whole digit realizations, but only for single columns. This column-by-column emission of feature vectors attempts to better model horizontal distortions at character level and, indeed, it is the usual approach in both speech and handwriting recognition when continuous-density (Gaussian mixture) HMMs are used. The reader can check that, by direct application of Eq. (2.3) and taking into account the existence of two different state sequences, the probability of generating the binary image generated from this BHMM example is 0.063.

As discussed in the introduction, BHMMs at global (line or word) level are built from shared, embedded BHMMs at character level. More precisely, let C be the number of different characters (symbols) from which global BHMMs are built, and assume that each character c is modeled with a different BHMM of parameter vector $\mathbf{\Theta}_c$. Let $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_C\}$, and let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of feature vectors generated from a sequence of symbols $S = (s_1, \ldots, s_L)$, with $L \leq T$. The probability of O can be calculated, using embedded HMMs for its symbols, as:

$$P(O \mid S, \mathbf{\Theta}) = \sum_{i_1,\ldots,i_{L+1}} \prod_{l=1}^{L} P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) \,, \tag{2.5}$$

where the sum is carried out over all possible segmentations of O into L segments, that is, all sequences of indices $i_1, \ldots, i_{L+1}$ such that

$$1 = i_1 < \cdots < i_L < i_{L+1} = T + 1;$$



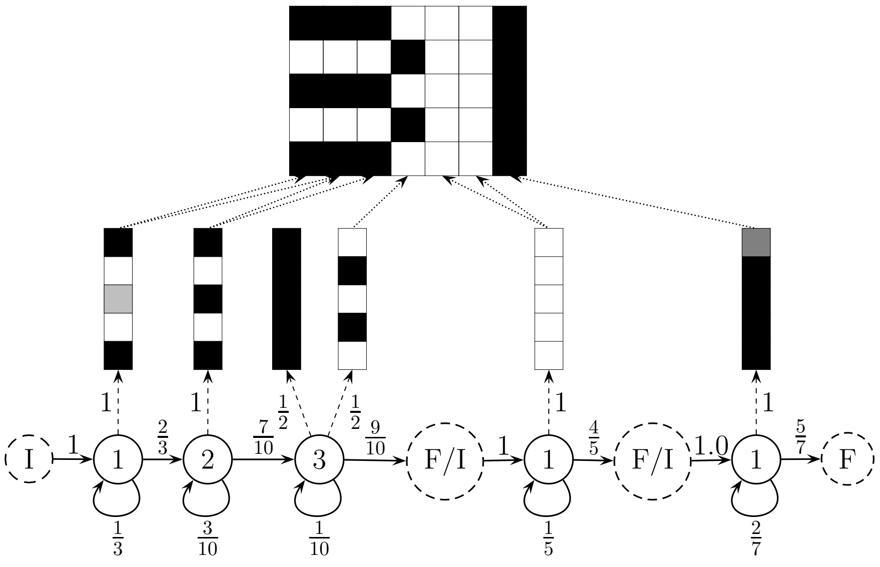


Figure 2.2: BHMM examples for the numbers 3 (top) and 31 (bottom), together with binary images generated from them. Note that the BHMM example for the number 3 is also embedded into that for the number 31. Bernoulli prototype probabilities are represented using the following color scheme: black=1, white=0,gray=0.5 and light gray=0.1.

and $P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l})$ refers to the probability (density) of the lth segment, as given by (2.3) using the HMM associated with symbol s_l.

Consider now the lower part of Figure 2.2. An embedded BHMM for the number 31 is shown, which is the result of concatenating BHMMs for the digit 3, blank space and digit 1, in that order. Note that the BHMMs for blank space and digit 1 are simpler than that for digit 3. Also note that the BHMM for digit 3 is shared between the two embedded BHMMs shown in the Figure. The binary image of the number 31 shown above can only be generated from two paths, as indicated by the arrows connecting prototypes to image columns, which only differ in the state generating the second image column (either state 1 or 2 of the BHMM for the first symbol). It is straightforward to check that, according to (2.5), the probability of generating this image is 0.0004.

2.4 BHMM-based Handwriting Recognition

Given an observation sequence $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$, its most probable transcription is obtained by application of the conventional Bayes decision rule:

$$w^* = \arg\max_{w \in W} \; p(w \mid O) \tag{2.6}$$

$$= \arg\max_{w \in W} \; p(w)\, p(O \mid w)\,, \tag{2.7}$$

where W is the set of possible transcriptions; $p(w)$ is usually approximated by an *n-gram language model* [11]; and $p(O \mid w)$ is a *text image model* which, in this work, is modeled as a BHMM (built from shared, embedded BHMMs at character level), as defined in Eq. (2.5). A particularly interesting case arises when the set of possible transcriptions reduces to a (small) finite set of *words (class labels)*. In this case, $p(w)$ is simply the *prior* probability of word w, while $p(O \mid w)$ is the probability of observing O given that it corresponds to a handwritten version of word w.

2.4.1 The forward algorithm

In order to efficiently compute $p(O \mid w)$ as a BHMM probability of the form given in Eq. (2.5), we use a dynamic programming method known as *forward algorithm* [24, 26]. For each time t, symbol s_l and state j from the HMM for symbol s_l, we define the *forward probability* $\alpha_{lt}(j)$ as:

$$\alpha_{lt}(j) = P(O_1^t, q_t = (l, j) \mid S, \mathbf{\Theta})\,, \tag{2.8}$$

that is, the probability of generating O up to its tth element and ending at state j from the HMM for symbol s_l. This definition includes (2.5) as the particular case in which $t = T$, $l = L$ and $j = F_{s_L}$; that is,

$$P(O \mid S, \boldsymbol{\Theta}) = \alpha_{LT(F_{s_L})} \,. \tag{2.9}$$

To compute $\alpha_{LT(F_{s_L})}$, we must first take into account that, for each position l in S except for the first, the initial state of the HMM for s_l is joined with final state of its preceding HMM, i.e.

$$\alpha_{lt}(I_{s_l}) = \alpha_{l-1t}(F_{s_{l-1}}) \qquad \begin{matrix} 1 < l \le L \\ 1 \le t \le T \end{matrix} \,. \tag{2.10}$$

Having (2.10) in mind, we can proceed at symbol level as with conventional HMMs. For the final states, we have:

$$\alpha_{lt}(F_{s_l}) = \sum_{j=1}^{M_{s_l}} \alpha_{lt}(j)\, a_{s_l j F_{s_l}} \qquad \begin{matrix} 1 \le l \le L \\ 1 \le t \le T \end{matrix} \,, \tag{2.11}$$

while, for regular states, $1 \le j \le M_{s_l}$, we have:

$$\alpha_{lt}(j) = \left[\sum_{i \in \{I_{s_l}, 1, \ldots, M_{s_l}\}} \alpha_{lt-1}(i)\, a_{s_l i j} \right] b_{s_l j}(\mathbf{o}_t) \,, \tag{2.12}$$

with $1 \le l \le L$ and $1 < t \le T$. The base case is for $t = 1$:

$$\alpha_{l1}(i) = \begin{cases} a_{s_1 I_{s_1} i}\, b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \le i \le M_{s_1} \\ 0 & \text{otherwise} \end{cases} \,. \tag{2.13}$$

The forward algorithm uses a dynamic programming table for $\alpha_{lt}(\cdot)$ which is computed forward in time to avoid repeated computations.

Figure 2.3 shows an application example of the forward algorithm to the BHMM and observation of Figure 2.2 (bottom). Non-null (and a few null) entries of the dynamic programming table are represented by graph nodes aligned with states (vertically) and time (horizontally). Node borders are drawn in black or gray, depending on whether they are in valid paths (i.e. those from which the observation sequence can be generated) or not. Also, those associated with special states are drawn with dotted lines. Numbers at the top of each node refer to $\alpha_{lt}(\cdot)$ and thus, for instance, the probability of generating O up to the third image column and ending at state 2 of the BHMM for the first symbol is $\alpha_{13}(2) = \frac{10}{450}$. Computation dependencies between nodes are represented by arrows, which are labeled above by, first, the transition probability, and then the observation probability at the target state (see Eq. (2.4)). For instance, the numbers above the arrow pointing to node $\alpha_{13}(4)$ are: $a_{s_1 23} \cdot b_{s_1 3}(\mathbf{o}_4) = \frac{7}{10} \cdot (\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1^5) = \frac{7}{10} \cdot \frac{1}{2}$.

From Figure 2.3, we can clearly see that, as indicated at the end of Section **??**, there are only two paths from which the observation can be generated. They share all nodes drawn with black borders except the two nodes aligned with the second observation vector. In accordance with Eq. (2.9), the probability of the observation sequence is $\alpha_{37}(F) = 0.0004$.

2.4.2 The backward algorithm

The *backward algorithm* is similar to the forward algorithm but, as it name indicates, it uses a dynamic programming table which is instead computed backward in time [24, 26]. The basic definition in this case is the *backward probability:*

$$\beta_{lt}(j) = P(O_{t+1}^T \mid q_t = (l, j), S, \mathbf{\Theta}) \ , \tag{2.14}$$

which measures the probability (density) of generating O_{t+1}^T given that the tth vector was generated in state j of the BHMM for the symbol s_l. Using this definition, Eq. (2.5) can be rewritten as:

$$P(O \mid S, \mathbf{\Theta}) = \sum_{j=1}^{M_{s_1}} a_{s_1 I_{s_1} j} \, b_{s_1 j}(\mathbf{o}_1) \, \beta_{11}(j) \ . \tag{2.15}$$

Taking into account that:

$$\beta_{lt}(F_{s_l}) = \beta_{l+1t}(I_{s_{l+1}}) \qquad \begin{array}{l} 1 \leq l < L \\ 1 \leq t < T \end{array} \ , \tag{2.16}$$

the backward probability for the initial and regular states, $i \in \{I_{s_l}, 1, \ldots, M_{s_l}\}$, can be efficiently computed as:

$$\beta_{lt}(i) = a_{s_{nl} i F_{s_l}} \beta_{lt}(F_{s_l}) + \sum_{j=1}^{M_{s_l}} a_{s_l ij} b_{s_l j}(\mathbf{o}_{t+1}) \beta_{lt+1}(j) \qquad \begin{array}{l} 1 \leq l \leq L \\ 1 \leq t < T \end{array} \ , \tag{2.17}$$

where the base case is defined for $t = T$ as

$$\beta_{lT}(i) = \begin{cases} a_{s_L i F_{s_L}} & l = L, 1 \leq i \leq M_{s_L} \\ 0 & \text{otherwise} \end{cases} \ . \tag{2.18}$$

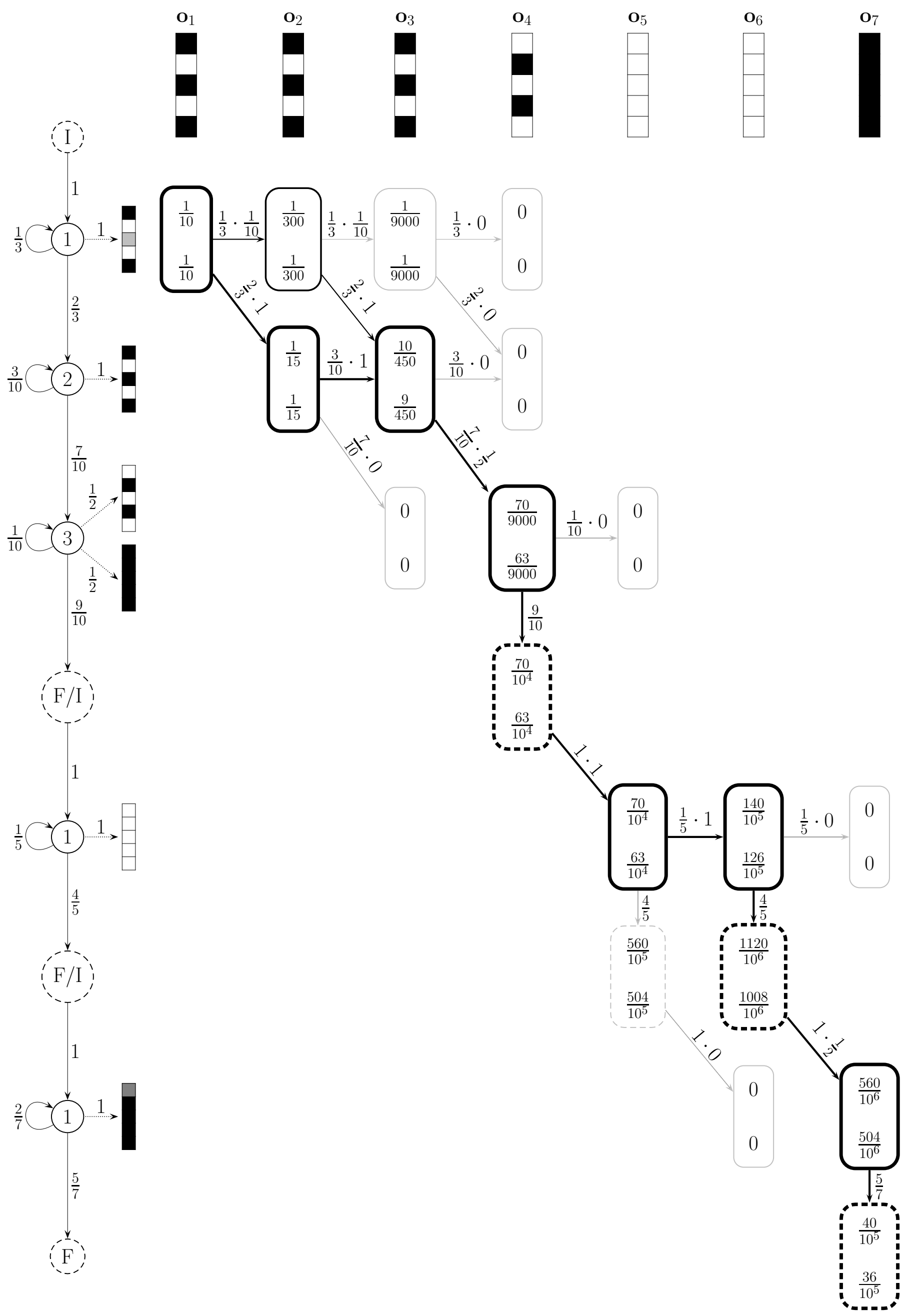


Figure 2.3: Application example of the forward and Viterbi algorithms to the the BHMM and observation of Figure 2.2 (bottom). Numbers at the top of the nodes denote forward probabilities, while those at the bottom refer to Viterbi scores.

2.4.3 The Viterbi algorithm

Although the forward and backward algorithms efficiently compute the exact value of $P(O \mid S, \mathbf{\Theta})$, it is common practice to approximate it by the so-called *Viterbi* or *maximum approximation,* in which the sums in Eqs. (2.3) and (2.5) are replaced by the max operator, i.e.

$$P(O \mid S, \mathbf{\Theta}) \approx \max_{\substack{i_1,\dots,i_{L+1}\\ q_1,\dots,q_T}} \prod_{l=1}^{L} \hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l})\,, \tag{2.19}$$

where the $\hat{P}$ is defined as:

$$\hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) = a_{s_l I_{s_l} q_{i_l}} \cdot \prod_{t=i_l}^{i_{l+1}-2} a_{s_l q_t q_{t+1}} \cdot a_{s_l q_{i_{l+1}-1} F_{s_l}} \cdot \prod_{t=i_l}^{i_{l+1}-1} b_{s_l q_t}(\mathbf{o}_t)\,. \tag{2.20}$$

In contrast to the exact definition, this approximation allows us to identify a single, best state sequence or *path* associated with the given observation sequence. The well-known *Viterbi algorithm* efficiently computes this approximation, using dynamic programming recurrences similar to those used by the forward algorithm. Formally, we need to compute the probability $Q(l,t,j)$ of the most likely path up to time t that ends with the state j from the BHMM for symbol s_l. For the specials states, it can be computed as:

$$Q(l,t,I_{s_l}) = Q(l-1,t,F_{s_{l-1}}) \qquad \begin{matrix} 1 < l \le L \\ 1 \le t \le T \end{matrix}\,. \tag{2.21}$$

$$Q(l,t,F_{s_l}) = \max_{1\le j\le M_{s_l}} Q(l,t,j)\, a_{s_l j F_{s_l}} \qquad \begin{matrix} 1 \le l \le L \\ 1 \le t \le T \end{matrix}\,, \tag{2.22}$$

while, for the regular states with $1 \le l \le L$ and $1 < t \le T$, we have:

$$Q(l,t,j) = \left[\max_{i\in\{I_{s_l},1,\dots,M_{s_l}\}} Q(l,t-1,i)\, a_{s_l ij}\right] b_{s_l j}(\mathbf{o}_t)\,, \tag{2.23}$$

The base case is for $t = 1$:

$$Q(l,1,i) = \begin{cases} a_{s_1 I_{s_1} i}\, b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \le i \le M_{s_1} \\ 0 & \text{otherwise} \end{cases}\,. \tag{2.24}$$

Clearly, the Viterbi algorithm can be seen as a minor modification of the forward algorithm in which only the most probable is considered in each node computation. Indeed, the application example shown in Figure 2.3 is used both, for the forward and

Viterbi algorithms. Now, however, the relevant numbers are those included at the bottom of each node, which denote $Q(l,t,j)$; i.e., at row 2 and column 3, we have $Q(1,3,2)=\frac{9}{450}$. Consider the generation of the third observation vector at the second state (for the first symbol). It occurs after the generation of the second observation vector, either at the first or the second states, but we only take into account the most likely case. Formally, the corresponding Viterbi score is computed as:

$$Q(1,3,2)=\max\left\{\frac{1}{15}\cdot\frac{3}{10}\cdot 1,\frac{1}{300}\cdot\frac{2}{3}\cdot 1\right\}=\max\left\{\frac{9}{450},\frac{1}{450}\right\}=\frac{9}{450}$$

Note that forward probabilities do not differ from Viterbi scores up to $Q(1,3,2)$, since it corresponds to the first (and only) node with two incoming paths. The Viterbi approximation to the exact probability of generating the observation sequence is obtained at the final node: $Q(3,7,F)=0.00036$. The most likely path, drawn with thick lines, is retrieved by starting at this node and moving backwards in time in accordance with computation of Viterbi scores. As usual in practice, the final Viterbi score in this example (0.00036) is a tight lower bound of the exact probability (0.00040).

2.5 Maximum likelihood parameter estimation

Maximum likelihood estimation of the parameters governing an embedded BHMM does not differ significantly from the conventional Gaussian case, and it can be carried out using the well-known EM (Baum-Welch) re-estimation formulae [24, 26]. Let $(O_1,S_1),\dots,(O_N,S_N)$, be a collection of N training samples in which the nth observation has length T_n, $O_n=(\mathbf{o}_{n1},\dots,\mathbf{o}_{nT_n})$, and was generated from a sequence of L_n symbols ($L_n\leq T_n$), $S_n=(s_{n1},\dots,s_{nL_n})$. At iteration r, the E step requires the computation, for each training sample n, of their corresponding forward and backward probabilities (see (2.8) and (2.14)), as well as the expected value for its tth feature vector to be generated from kth component of the state j in the HMM for symbol s_l,

$$z_{nltk}^{(r)}(j)=\frac{\pi_{s_{nl}jk}^{(r)}\prod_{d=1}^{D}p_{s_{nl}jkd}^{(r)}{}^{o_{ntd}}(1-p_{s_{nl}jkd}^{(r)})^{1-o_{ntd}}}{b_{s_{nl}j}^{(r)}(\mathbf{o}_{nt})},$$

for each t, k, j and l.

In the M step, the Bernoulli prototype corresponding to the kth component of the state j in the HMM for character c has to be updated as:

$$\mathbf{p}_{cjk}^{(r+1)} = \frac{1}{\gamma_{ck}(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j) \mathbf{o}_{nt}}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}, \tag{2.25}$$

where $\gamma_{ck}(j)$ is a normalization factor,

$$\gamma_{ck}(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}, \tag{2.26}$$

and $\xi_{nltk}^{(r)}(j)$ the probability for the tth feature vector of the nth sample, to be generated from the kth component of the state j in the HMM for symbol s_l,

$$\xi_{nltk}^{(r)}(j) = \alpha_{nlt}^{(r)}(j) z_{nltk}^{(r)}(j) \beta_{nlt}^{(r)}(j). \tag{2.27}$$

Similarly, the kth component coefficient of the state j in the HMM for character c has to be updated as:

$$\pi_{cjk}^{(r+1)} = \frac{1}{\gamma_c(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}, \tag{2.28}$$

where $\gamma_c(j)$ is a normalization factor,

$$\gamma_c(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \alpha_{nlt}^{(r)}(j) \beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}. \tag{2.29}$$

To avoid null probabilities in Bernoulli prototypes, they can be smoothed by linear interpolation with a flat (uniform) prototype, $\mathbf{0.5}$,

$$\tilde{\mathbf{p}} = (1-\delta)\,\mathbf{p} + \delta\,\mathbf{0.5}, \tag{2.30}$$

where, for instance, $\delta = 10^{-6}$.

2.6 Experiments

Experiments reported here were carried out on the very popular IfN/ENIT database of Arabic handwritten Tunisian town names [23]. More details about this database are in section 2.6.1. Each image was rescaled in height to a given dimension D (10, 15, 20, 25, 30,35), while keeping the original aspect ratio, and then binarized using Otsu binarization. In the results reported below, however, only height 30 is considered since, in a series of preliminary informal tests, it led to better results than other heights.

2.6.1 IfN/ENIT database

IfN/ENIT database is an Arabic handwritten text database which contains handwritten Tunisian town/villages names [23]. It is then a database for isolated word recognition. In last years this database has been used in several Arabic handwritten competitions, see [19, 16, 17, 18], becoming a reference in the Arabic handwritten area. The database consists of 946 Tunisian town/villages. It is written by 411 writers. They were asked to fill 5 forms with 12 names from the possible names with their corresponding postcodes. Forms were made guarantying that each word appears at least 3 times in the database, and each character shape occur at minimum more than 200 times. The only aid to writing was the printing of dark light rectangles in the backside of the form to indicate where to write the words. Figure 2.5 shows two examples of forms.

Forms were scanned with 300 dpi and, binarised and automatically segmented. Using a semi-automatically process segmented images were labeled with the postcode, the Arabic word in codes as "ISO 8859-6" with a sequence of Arabic character shapes from 306 different shapes, since each letter appears in four different forms depending of its position in the word (Begin, Middle, End or Isolated form). It is important to note that "ISO 8859-6" does not encode the shape information.

The resulting database is composed by 32492 different images divided into 5 sets (a,b,c,d and e). The first four sets are the original sets of the database, while the set e was used as test set in the ICDAR 2005 competition see [19], being late released. Thus it is a common practice to public results doing a cross validation experiment with the first four sets, and a final experiment training with sets a,b,c,d and testing the set e. Note that while the number of classes is 946 (postcodes), the size of the lexicon is greater since names are written in different forms. Table 2.1 shows some statistics for the five sets, and figure 2.4 shows some samples of images.

Table 2.1: Some statistics of the IfN/ENIT-database sets

	No. of words	Lexicon
a	6537	1588
b	6710	1634
c	6477	1498
d	6735	1564
e	6033	733

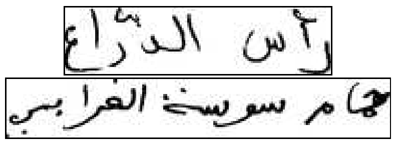


Figure 2.4: Samples of the IfN/ENIT database

2.6.2 Evaluation Metrics

Results in general are obtained following a very popular evaluation metrics, we describe two of them in this thesis. First, the classification error rate, which is the conventional metric used in pattern recognition for classification problems. It is used for the isolated words recognition, where each word indicates a class label. It is calculated as the percentage of errors in test set,

$$Error = \frac{NE}{N} \times 100 \tag{2.31}$$

where NE is the number of errors, and N is the number of samples in the test set. This metric could also be used in continuous HTR (line condition), in which a sequence of words is obtained as a result of the recognition process.However, it is somewhat a strict metric, since an erroneous word on the sequence implies that the sequence is considered all erroneous.

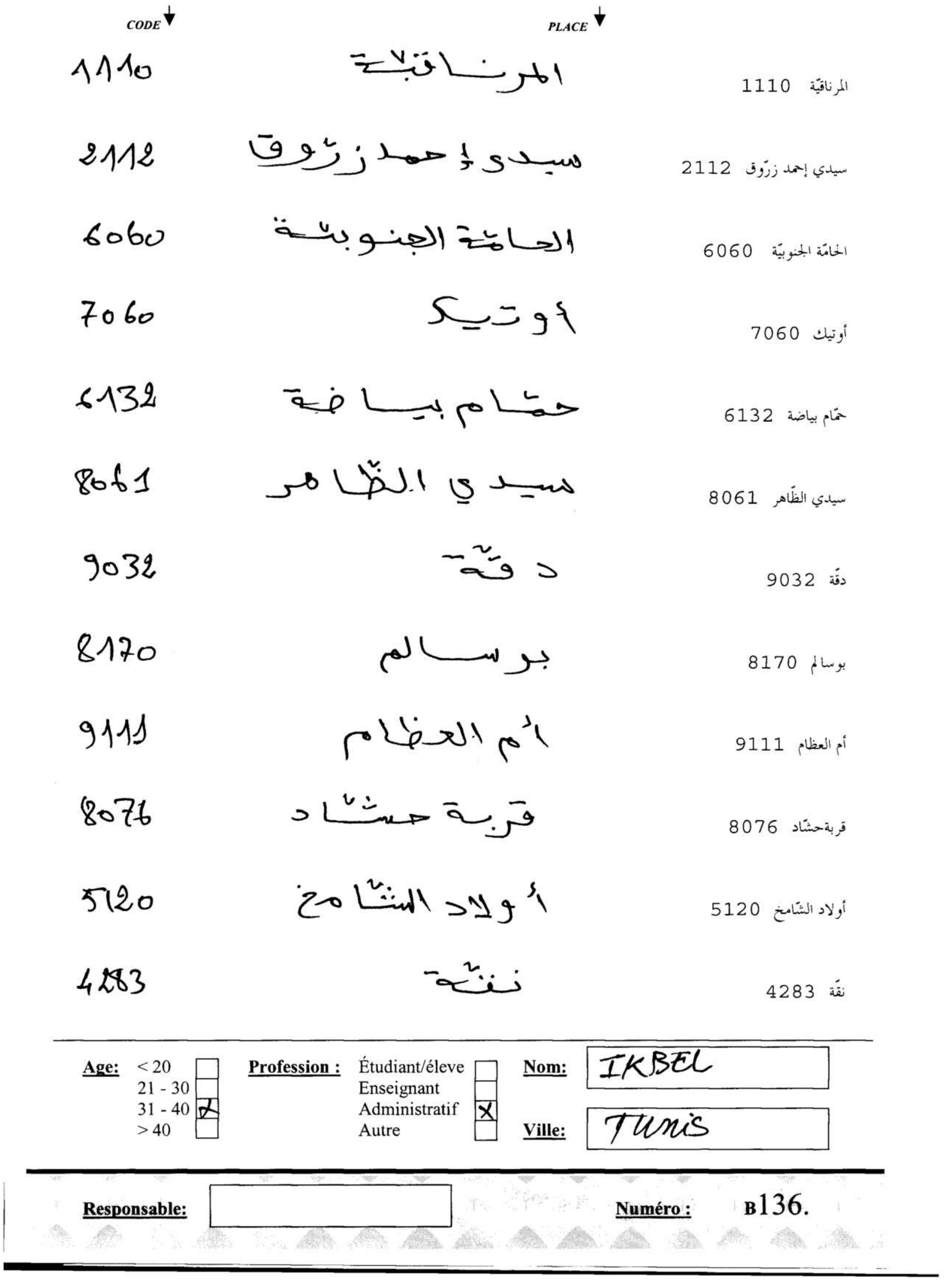
The second metric is the Word Error Rate (WER), is a common metric of the performance of a speech recognition or machine translation system. This metrics tends to calculate the error at word level instead of at sentence level. That is, for each predicted sentence, the minimum number of insertions, deletions, and substitutions is calculated.

$$WER = \frac{NI + ND + NS}{NE}, \tag{2.32}$$

where NI, ND, and NS, are respectively the total number if needed insertions, deletions, and substitutions, while NW is the total number of words in the reference.

2.6.3 Experiments

Experiments were carried out by trying different number of states, $Q \in \{2, 4, 6, 8\}$, and also different number of mixture components per state, $K \in \{1, 4, 16, 64, 128\}$. For



CODE	PLACE		
1110	المرناقيّة	1110	المرناقيّة
2112	سيدي إحمد زروق	2112	سيدي إحمد زرّوق
6060	الحامّة الجنوبيّة	6060	الحامّة الجنوبيّة
7060	أوتيك	7060	أوتيك
6132	حمّام بياضة	6132	حمّام بياضة
8061	سيدي الظّاهر	8061	سيدي الظّاهر
9032	دقّة	9032	دقّة
8170	بوسالم	8170	بوسالم
9111	أم العظام	9111	أم العظام
8076	قربة حشّاد	8076	قربةحشّاد
5120	أولاد الشّامخ	5120	أولاد الشّامخ
4283	نقّة	4283	نقّة

Age: < 20 ☐ 21 - 30 ☐ [✓ 31 - 40] > 40 ☐

Profession : Étudiant/éleve ☐ Enseignant ☐ [✓ Administratif] Autre ☐

Nom: IKBEL

Ville: TUNIS

Responsable:

Numéro : B136.

Figure 2.5: Example of two handwritten forms from the IfN/ENIT database

$K = 1$, the HMMs were initialized by first segmenting the training set with a "neutral" model, and then using the resulting segments to perform a Viterbi initialization. The initialized HMMs were trained with 4 EM iterations. For $K > 1$, the HMMs were initialized by splitting the mixture components of the models trained with $K/4$ (or $K/2$) mixture components per state. Again, after initialization, HMMs were trained with 4 EM iterations. On the other hand, recognition of test images was performed by using the Viterbi algorithm.

Figure 2.6 shows the Word Error Rate (WER%), as a function of the number of states, for varying number of components. Each WER estimate (plotted point) was obtained by cross-validation with the first 4 standard folds (a, b, c and d).

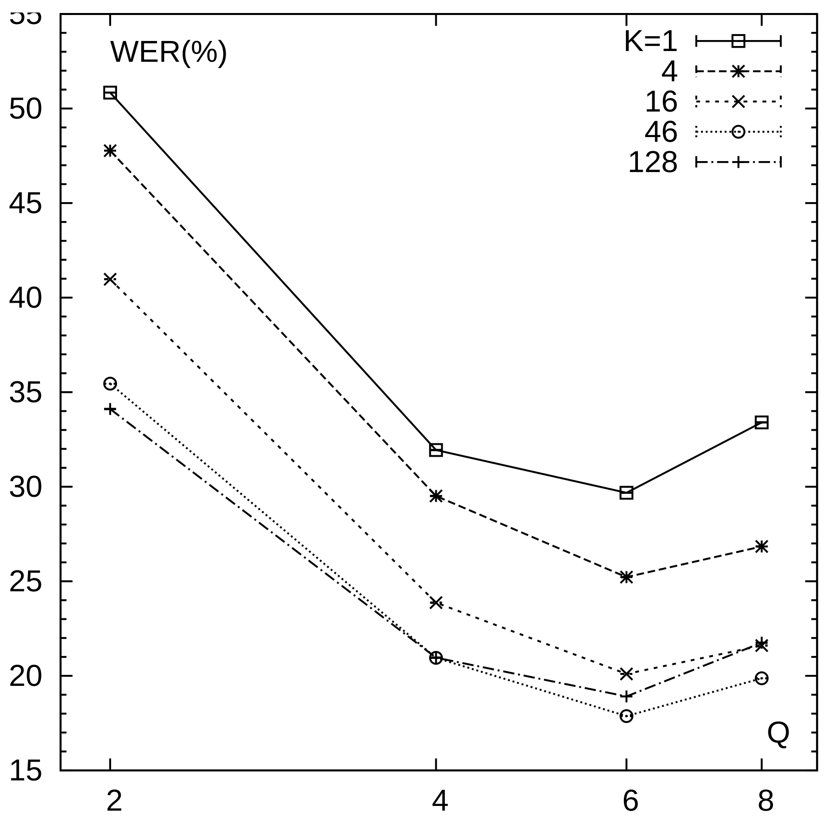


Figure 2.6: Classification error-rate (%) as a function of the number of states, for varying number of components (K). Cross-validation using sets (a,b,c,d).

From the results in Fig. 2.6, it seems that an appropriate value for the number of states is 6, and also an appropriate value for the number of mixture components per state is 64. Using these values, two additional experiments were carried out by using the training-test partitions abcd-e and abcde-e. The resulting WER values are included in Table 2.2 together with those obtained in the other training-test combinations involved in the 4-fold cross-validation experiment performed previously.

Table 2.2: Word Error Rate (WER%), for $Q = 6$ and $K = 64$, in different training-test combinations of the a, b, c, d and e folds.

Training	Test	WER%
abc	d	17.6
abd	c	17.3
acd	b	19.0
bcd	a	17.5
abcd	e	34.3
abcde	e	24.3

From the results in Table 2.2, we can see that the results for the first four folds are very similar, in the range 17.3% – 19.0%, while those for fold e (34.3% and 24.3%) are significantly higher. This might be due to the different age and profession distribution of the writers that contributed to fold e, as compared with those of the first four folds [19]. On the other hand, when compared with the results on fold e (abcde-e) at the ICDAR 2007 competition, our 24.3% would rank in the middle part of the list, far from the best results, but nonetheless above many participating systems. We think that this is a relatively good result since our Bernoulli mixture HMM-based system is still at a basic state of development and, therefore, there is significant room for improvement.

2.7 Concluding Remarks

Embedded Bernoulli mixture HMMs have been applied to Arabic handwriting recognition and, more precisely, they have been tested on the very popular IfN/ENIT database of Arabic handwritten Tunisian town names.

3

Windowed BHMMs

3.1 Introduction

In this chapter, we provide new, much more improved results on the IfN/ENIT database. In contrast to our basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, now we use a sliding window of adequate width to better capture image context at each horizontal position of the word image. It must be noted that the use of sliding windows for HMM-based handwriting recognition is not new and, indeed, the best Word Error Rate (WER) reported on the standard *abcd-e* training-test partition of IfN/ENIT, 14.6%, has been obtained using *windowed* Gaussian mixture HMMs [7]. In this chapter, however, we show that our windowed BHMMs approach leads to an even better WER of 12.3%.

In what follows, we first describe windowed BHMMs formally, with intuitive examples. Then, the new empirical results are reported in Section 3.3. Concluding remarks are given in Section 3.4.

3.2 Windowed BHMMs

Given a binary image normalized in height to H pixels, we may think of a feature vector $\mathbf{o}_t$ as its column at position t or, more generally, as a concatenation of columns in a window of W columns in width, centered at position t. This generalization has no effect neither on the definition of BHMM nor on its maximum likelihood estimation, though it might be very helpful to better capture image context at each horizontal position of the image. As an example, Figure 3.1 shows a binary image of 4 columns and 5 rows, which is transformed into a sequence of 4 15-dimensional feature vectors by application of a sliding window of width 3. For clarity, feature vectors are depicted as 3×5 subimages instead of 15-dimensional column vectors. Note that feature vectors at positions 2 and 3 would be indistinguishable if, as in our previous approach, they were extracted with no context ($W = 1$).

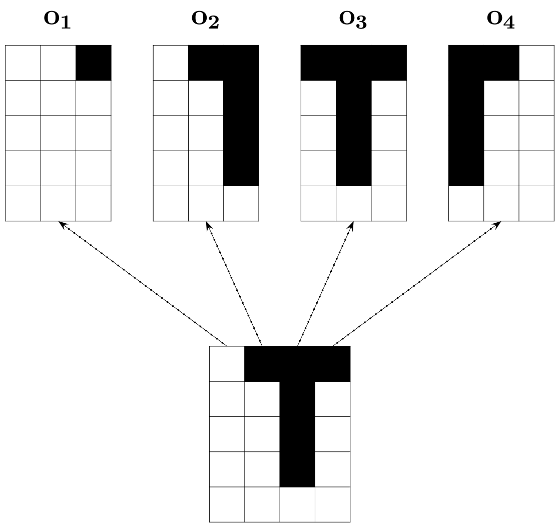


Figure 3.1: Example of transformation of a 4×5 binary image (bottom) into a sequence of 4 15-dimensional binary feature vectors $O = (\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4)$ using a window of width 3.

3.3 Experiments

As described before, experiments have been carried out using the well-known IfN/ENIT database of Arabic handwritten Tunisian town names [23]. More precisely, we have used the IfN/ENIT database in version 2.0, patch level 1e (v2.0p1e), which is exactly the version used as training data in the Arabic handwriting recognition competition held at ICDAR (Int. Conf. on Document Analysis and Recognition) in 2007 and 2009 [16, 17]. It comprises 32492 Arabic words written by more than 1000 different writers, from a lexicon of 937 Tunisian town/village names. A standard partition is defined which consists of five folds labeled as a, b, c, d and e. Each image was scaled in height to 30 pixels and then binarized using Otsu's method.

We tried different values for the sliding window width, $W \in \{1, 3, 5, 7, 9, 11\}$ and also different different values for number of mixture components per state, $K \in$

$\{1, 2, 4, 8, 16, 32, 64\}$. However, taking into account our previous, preliminary results in [13], we only tried BHMMs of 6 states. For $K = 1$, BHMMs were initialized by first segmenting the training set with a "neutral" model, and then using the resulting segments to perform a Viterbi initialization. Initialized HMMs were trained with 4 EM iterations. For $K > 1$, the HMMs were initialized by splitting the mixture components of the models trained with $K/2$ mixture components per state. Again, after initialization, HMMs were trained with 4 EM iterations. On the other hand, recognition of test images was performed by using the Viterbi algorithm. Figure **??** shows the Word Error Rate (WER%) as a function of the number of mixture components, for varying sliding window widths. Each WER estimate (plotted point) was obtained by cross-validation with the first 4 standard folds (abcd).

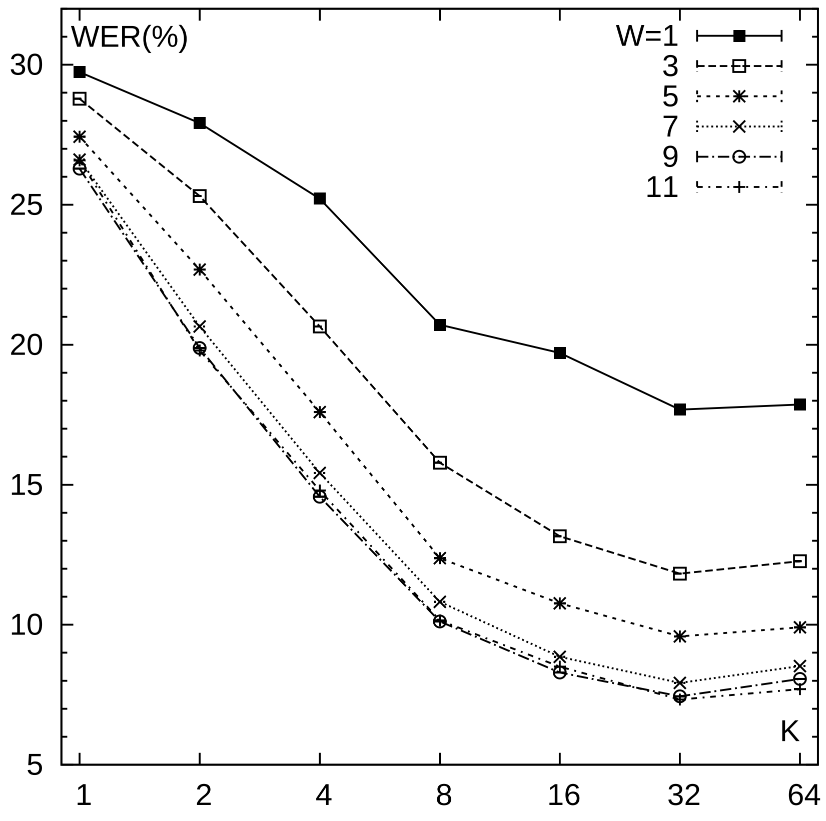


Figure 3.2: WER(%) as a function of the number of mixture components (K) for varying sliding window widths (W).

From the results in Figure 3.2 it becomes clear that the use of a sliding window improves the results to a large extent. In particular, the best result, 7.4%, is obtained for $W = 9$ and $I = 32$, though very similar results are also obtained for $W = 7$ and $W = 11$. It is worth noting that the best result achieved with no sliding windows ($W = 1$) is 17.7%.

To get some insight into the behavior of our windowed BHMMs, the model for character خ, trained from folds abc with $W = 9$ and $K = 32$, is (partially) shown in Figure 3.3 (bottom) together with its Viterbi alignment with a real image of the character خ, extracted from sample *de05_007* (top). As in Figure **??**, Bernoulli prototypes are represented as grey images where the grey level of each pixel measures the probability of its corresponding pixel to be black (white = 0 and black = 1). From these prototypes, it can be seen that the model works as expected, i.e. each state from right to left accounts for a different local part of خ, as if the sliding window was moving smoothly from right to left. Also, note that the main stroke of the character خ appears almost neatly drawn in most prototypes, whereas its upper dot appears blurred, probably due to a comparatively higher variability in window position.

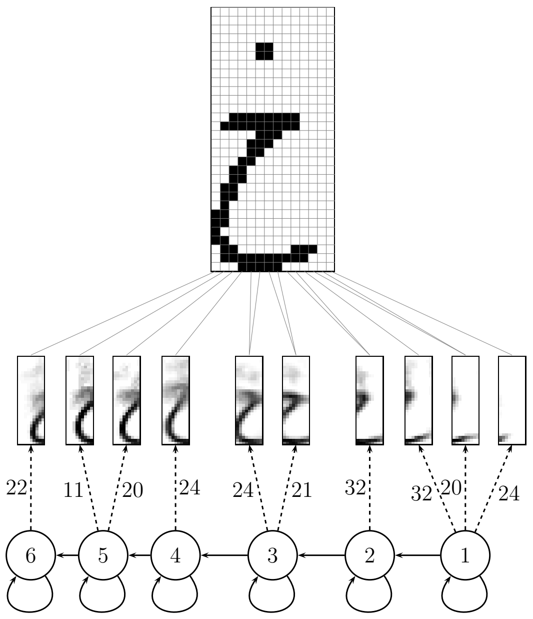


Figure 3.3: BHMM for character خ, trained from folds abc with $W = 9$ and $K = 32$ (bottom), together with its Viterbi alignment with a real image of the character خ, extracted from sample *de05_007* (top).

As discussed in [7], letters in Arabic script differ significantly in length, and thus it might not be appropriate to model all of them using BHMMs of identical number of states. With this idea in mind, a new experiment was carried out, similar to that described above, but with fixed sliding window of $W = 9$ and variable number of states per character. To decide the number of states for each character, we first Viterbi-segmented all training data using BHMMs of 4 states, and then computed the average length of the segments associated with each character. Given an average segment length for character c, $\bar{T}_c$, its number of states was set to $F \cdot \bar{T}_c$, where F is a *factor* measuring the average number of states that are required to emit a feature vector. Thus, its inverse, $\frac{1}{F}$, can be interpreted as a *state load,* that is, the average number of feature vectors that are emitted in each state. For instance, $F = 0.2$ means that only a fraction of 0.2 states is required to emit a feature vector or, alternatively, that $\frac{1}{0.2} = 5$ feature vectors are emitted on average in each state. Figure 3.4 shows the WER obtained as a function of F, $F \in \{0.2, 0.3, 0.4, 0.5\}$, for varying values of the number of mixture components. The best result achieved is a WER of 7.3%, using $F = 0.4$ and $K = 32$, which is slightly better than the 7.4% obtained with 6 states per character.

In Figure 3.5, the sample *dm33_037* has been recognized using BHMMs with $W = 9$, $K = 32$ and both, 6 states (top) and variable number of states, with $F = 0.4$ (bottom). In both cases, the recognized word has been Viterbi-aligned at character level (background color) and state level (bottom and upper ticks). Although the BHMMs of 6 states produce a recognition error, النفَاتية (top), the BHMMs of variable number of states are able to recognize the correct word, الدخّانية (bottom). Note that there are two letters, 'ﻟ' and 'ﺩ', that are written at the same vertical position, or better to say, at a specific column, and thus it is very difficult for our BHMMs to recognize them as two different letters. On the other hand, the incorrectly recognized word (top) is not very different in shape from the correct one; e.g. the characters 'ﻧ' and 'ﺗ' are very similar (type B [19]).

A final experiment was carried out using the four folds of the previous experiments, abcd, as training data, and the fifth fold, e, as test data. Taking into the results obtained above, we tried BHMMs with $W = 9$, $K = 32$, and both, $Q = 6$ states and variable number of states with $F = 0.4$. The WER achieved in both cases is included in Table 3.1, together with WER results for other training-test partitions, including the 4 partitions involved in the 4-fold cross-validation experiments described above.

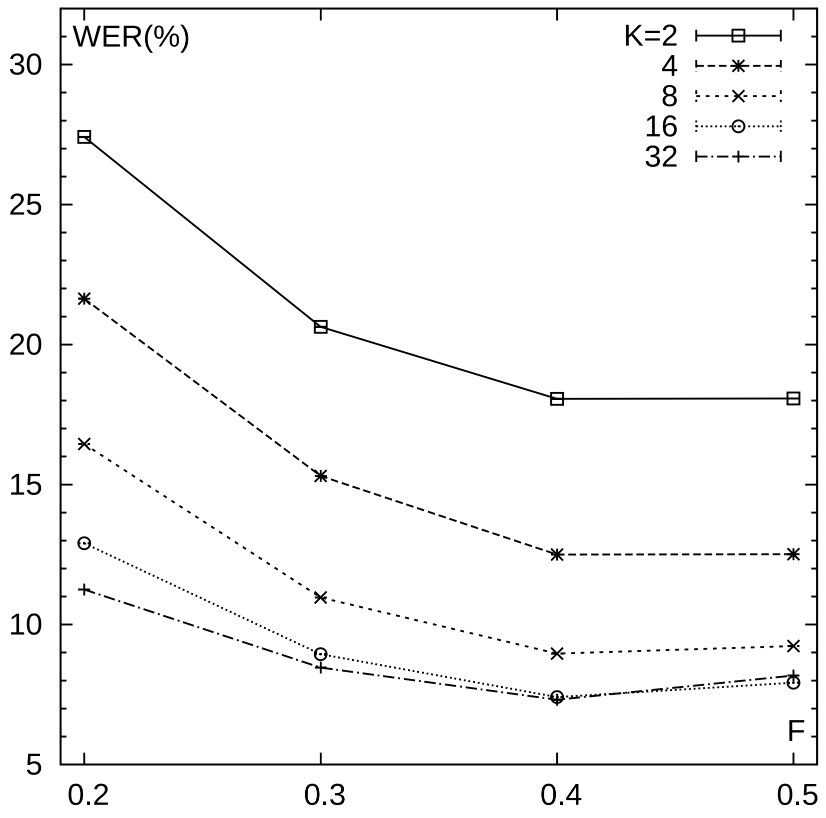


Figure 3.4: WER(%) as a function of the factor F for varying values of the number of mixture components (K).

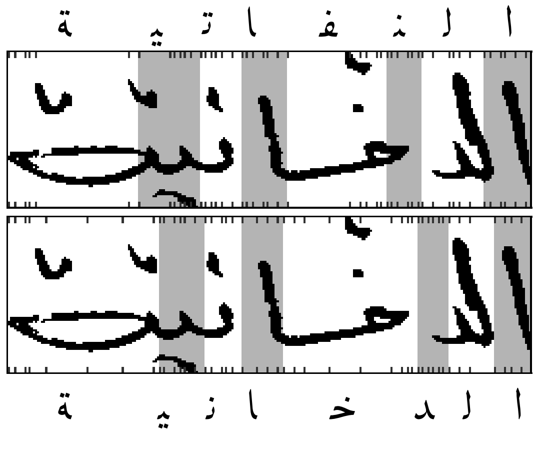


Figure 3.5: Sample *dm33_037* incorrectly recognized with BHMMs of 6 states (top), but correctly recognized with BHMMs of variable number of states (bottom). In both cases, the recognized word has been Viterbi-aligned at character level (background color) and state level (bottom and upper ticks).

Table 3.1: Word Error Rate (WER%) in different training-test combinations of the abcde folds, for BHMMs with $W = 9$, $K = 32$, and both, $Q = 6$ states and variable number of states with $F = 0.4$.

		WER%	
Training	Test	$Q = 6$	$F = 0.4$
abc	d	8.0	7.5
abd	c	6.6	6.9
acd	b	7.4	7.7
bcd	a	7.8	7.6
abcd	**e**	**13.7**	**12.3**
abcde	e	5.4	4.0

In Table 3.1, it can be seen that the results for the first 4 folds are very similar, in the range 6% – 9%, while those for fold e (13.7% and 12.3%) are higher. This might be due to the different age and profession distribution of the writers that contributed to fold e, as compared with those of the first 4 folds [19]. On the other hand, when compared with the results on fold e (abcde-e) at the ICDAR 2007 competition, our 4% outperforms the three best results. If the comparison is done with the more recently results of the ICDAR 2009 competition, our result would rank in the top of the list. In both cases, however, the results must be interpreted with caution since fold e is used both for training and testing. Nevertheless, on the standard abcd-e partition, our WER of 12.3% outperforms the 14.6% reported in [7] which, to our knowledge, is the best result known to date.

3.4 Concluding Remarks

Windowed Bernoulli mixture HMMs (BHMMs) have been defined and tested for Arabic Handwritten Word Recognition on the the well-known IfN/ENIT database of handwritten Tunisian town names. In contrast to our previous basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, we have used a sliding window of adequate width to better capture image context at each horizontal position of the word image. Very good results have been reported on IfN/ENIT and, in particular, a WER of 12.3% has been achieved on the standard abcd-e partition.

4

Repositioning in Windowed BHMMs

4.1 Introduction

In this chapter, again, more improved results on the IfN/ENIT database. Our new approach does not differ a lot from our previous approach, where a sliding window of fixed width on the binarized image is first centered at each column, and then translated to align its center with its mass center. The binary image under the translated window is read to construct a local binary feature vector and, in this way, the whole input image is transformed into a sequence of binary feature vectors. In this chapter, we show that our windowed BHMMs with repositioning approach leads to an even better WER of 6.1%.

In what follows, we first describe repositioning in windowed BHMMs, with intuitive examples. Then, empirical results are reported in Section 4.3. Concluding remarks are given in Section 4.5.

4.2 Window Repositioning

As we have seen in chapter 3, given a binary image normalized in height to H pixels, we may think of a feature vector $\mathbf{o}_t$ as a concatenation of columns in a window of W columns in width, centered at position t. Although one-dimensional, "horizontal" HMMs for image modeling can properly capture non-linear horizontal image distortions, they are somewhat limited when dealing with vertical image distortions, and this limitation might be particularly strong in the case of feature vectors extracted with significant context. To overcome this limitation, we have considered three methods of window *repositioning* after window extraction: *vertical, horizontal,* and *both.* The basic idea is to first compute the compute the center of mass of the extracted window, which is then repositioned (translated) to align its center to the center of mass. This is done in accordance with the chosen method, that is, horizontally, vertically, or in both directions. Obviously, the feature vector actually extracted is that obtained after repositioning. An example of feature extraction is shown in Figure 4.1 in which the the standard method (no repositioning) is compared with the three methods repositioning methods considered.

To illustrate the effect of repositioning with real data, Figure 4.2 shows the sequence of feature vectors extracted from a real sample of the IfN/ENIT database, with and without (both) repositioning. As intended, (vertical or both) repositioning has the effect of normalizing vertical image distortions, especially translations.

4.3 Experiments

In the experiments described in previous chapters, we have not tried window repositioning after window extraction but, as discussed in Sec. 3.2, many recognition errors of our BHMM-based classifier might be due to its limited capability to properly model vertical image distortions. In order to study the effect of repositioning on the classification accuracy, the standard method (no repositioning) was compared with the three repositioning methods described in Sec. 3.2: vertical, horizontal, and both directions. This was done with $W = 9$, $K = 32$, and $F = 0.4$, for the four partitions considered in the previous experiments (abc-d, abd-c, acd-b, and bcd-a), and also for the partitions abcd-e and abcde-e, which are commonly used to compare classifiers in the IfN/ENIT task, abcd-e especially. The results are included in Table 4.1.

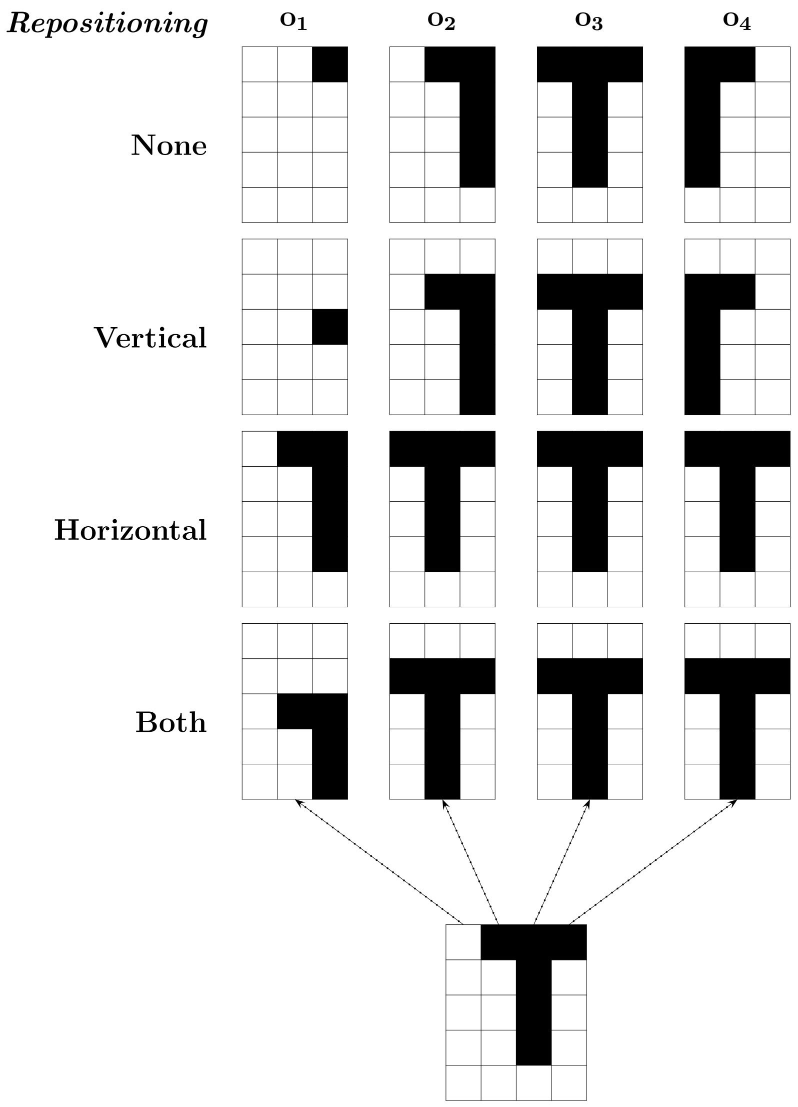


Figure 4.1: Example of transformation of a 4×5 binary image (bottom) into a sequence of 4 15-dimensional binary feature vectors $O = (\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4)$ using a window of width 3. The standard method (no repositioning) is compared with the three repositioning methods considered: vertical, horizontal, and both directions.

As expected, from the results in Table 4.1 it becomes clear that vertical (or both) window repositioning improves very much the results obtained with the standard method or horizontal repositioning alone. To our knowledge, the result obtained for the abcd-e partition with vertical (or both) repositioning, **6.1**%, is the best result reported on this partition to date. Indeed, it represents a 50% relative error reduction with respect to

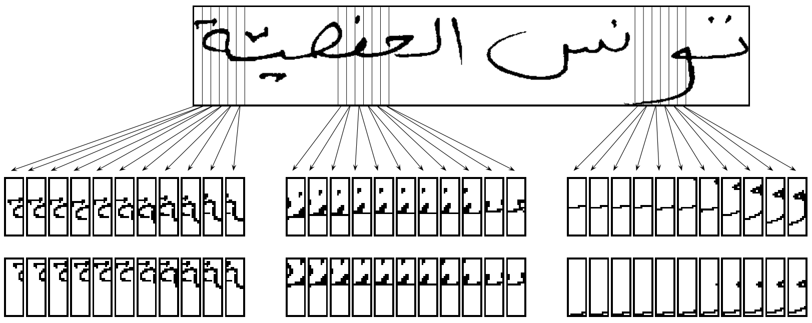


Figure 4.2: Original sample *pf069_011* from IfN/ENIT database (top) and its sequence of feature vectors produced with and without (both) repositioning (center and bottom, respectively).

Table 4.1: Word Error Rate (WER%) of the BHMM-based recognizer (with $W = 9$, $K = 32$, and $F = 0.4$) in different training-test combinations of the abcde folds, for four repositioning methods: none, vertical, horizontal, and both directions.

		WER%			
Training	Test	None	Vertical	Horizontal	Both
abc	d	7.5	4.7	8.4	4.8
abd	c	6.9	3.6	7.7	3.8
acd	b	7.7	4.5	8.1	4.4
bcd	a	7.6	4.4	8.2	4.6
abcd	**e**	**12.3**	**6.1**	**12.4**	**6.1**
abcde	e	4.0	2.2	3.9	2.0

the 12.3% of WER obtained without repositioning which, to our knowledge, was the best result published until now [9].

4.4 ICFHR 2010 competition results

In this section, we summarize the most recent published results in the ICFHR 2010 competition [18] on the well-known IfN/ENIT database, which presents the current stat-of-art for Arabic Handwriting Recognition.

The competition was held in Kolkata (India), where 4 groups with 6 systems participated. The systems were compared based on the recognition rate. For the test purpose, two new datasets was introduced, set f and s. Set f was collected in Tunisia, while set s was collected in the United Arab Emirates.

According to [18], our system with window repositioning (UPV-BHMM2) obtained the best results on set s with 84.62%, and second best results on set f with 92.20%. It is good to know, that all previous sets were collected in Tunisia including set f, but not set s which is more general from the writing variety, that makes sense why recognition rates for set s is a bit different. Our system won the award of the first Place Prize in the IAPR sponsored Arabic Handwriting Recognition Competition organized during ICFHR 2010.

4.5 Concluding Remarks

We have considered three methods of window repositioning after window extraction so as to help our BHMM-based recognizer in dealing with vertical image distortions. As expected, the best results have been obtained with an adequate adjustment of the window width, number of states, number of mixture components and, what it seems even more important, (vertical) window repositioning after window extraction. A WER of 6.1% has been achieved on the standard abcd-e partition which outperforms the best result known to date. Moreover, our participation in the ICFHR 2010 competition fulfilled with success and we won the award of the first Place Prize.

5

OpenHaRT Experiments

In this chapter, we extend the empirical results reported in previous chapters to the NIST OpenHaRT database [21]. For more details about databases, please review section 5.1. The IfN/ENIT database is small in size, so we used it first to find out some parameters such as best image width, number of states, etc. Adding to that, the NIST OpenHaRT database is very huge. It is a very costly process to try all possible parameter values. Therefore we used those valuess that obtained best results on IfN/ENTI database to train the models on the OpenHaRT database. The NIST OpenHaRT database contains lines of handwritten text. Therefore, the task to be performed on this database is not only a word recognition task, but also a line recognition task. Due to the varying number of words per line, the error rate for each predicted sentence is calculated by finding the minimum number of insertions, deletions and substitutions 2.32, not only by calculating substitutions.

5.1 NIST OpenHaRT database

The National Institute of Standards and Technology (NIST) Open Handwriting Recognition and Translation (OpenHaRT) evaluation is a public evaluation of image-to-text transcription and translation, similar to the tasks evaluated by NIST for the DARPA Multilingual Automatic Document Classification Analysis and Translation (MADCAT) Program, see [21]. The 2010 evaluation focuses on recognition and translation of images containing primary Arabic handwritten script. Note that in this competition, no previous results were published on the available data.

The data being used for OpenHaRT 2010 was created by the Linguistic Data Consortium (LDC) and has been used in previous MADCAT evaluations. This data was created in a controlled environment where known scribes copied Arabic source texts that were previously used in the DARPA GALE program. The source text was originally in electronic format. A corresponding document image was created by instructing literate native Arabic writers to produce handwritten copies of chosen passages using various writing conditions. Each passage was copied by at least two scribes. The handwritten copies were then scanned at 600 dpi to create the document images in TIFF format. A writing factor is considered, the writing instrument, surface and speed. Please refer to the NIST evaluation plan for details [21].

The database comprises 39056 Arabic image documents. It was written by more than 100 different writers. It contains a lexicon of 101515. A standard partition is defined which consists of two sets for training which consists of 37611 documents, in total of 6000 passages, and another two sets for development which consists of 1445 documents, in total of 100 passages. Table 5.1 explores more details.

	Docs	**Lines**	**Words**	**Lexicon**
Training set	37611	663177	3734356	
Dev set	1445	23746	141870	
SmallCorpus1	260	4336	23000	101515
SmallCorpus2	2600	41743	242385	101515
All	39056	686923	3876226	101515

Table 5.1: Database statistics extracting words and lines

For the evaluation process, both tasks are paired with segmentation conditions to explore the relationship between system performance and the system's ability to segment the data. Segmentation is represented as a series of polygon coordinates indicating the locations of the text segments within the image. The two segmentation conditions are referred to as word segmentation and line segmentation.

- **Word segmentation** is created manually. Human annotators mark the word boundaries using the GEDI tool. The input to GEDI is a document image.

- **Line segmentation** is the primary segmentation condition. It is defined as a bounding box that surrounds a line of text and is derived algorithmically from the word segmentations by creating polygons that minimize the amount of text overlap between the lines.

The database was divided into several partitions 5.1. A small partition of the data-set has been used to find the best parameters. It is called “smallcorpus1”. This partition contains of 200 documents for training, which equivalent to 20000 words, and 60 documents for evaluation which equivalent to 3000 words. Another small partition of the database was used. It is called “smallcorpus2”, it consists of 2000 documents for training, and 600 documents for evaluation which is equivalent to 32570 lines for training, and 9173 lines for evaluation, and also equivalent to 184433 words for training, and 57952 words for testing.

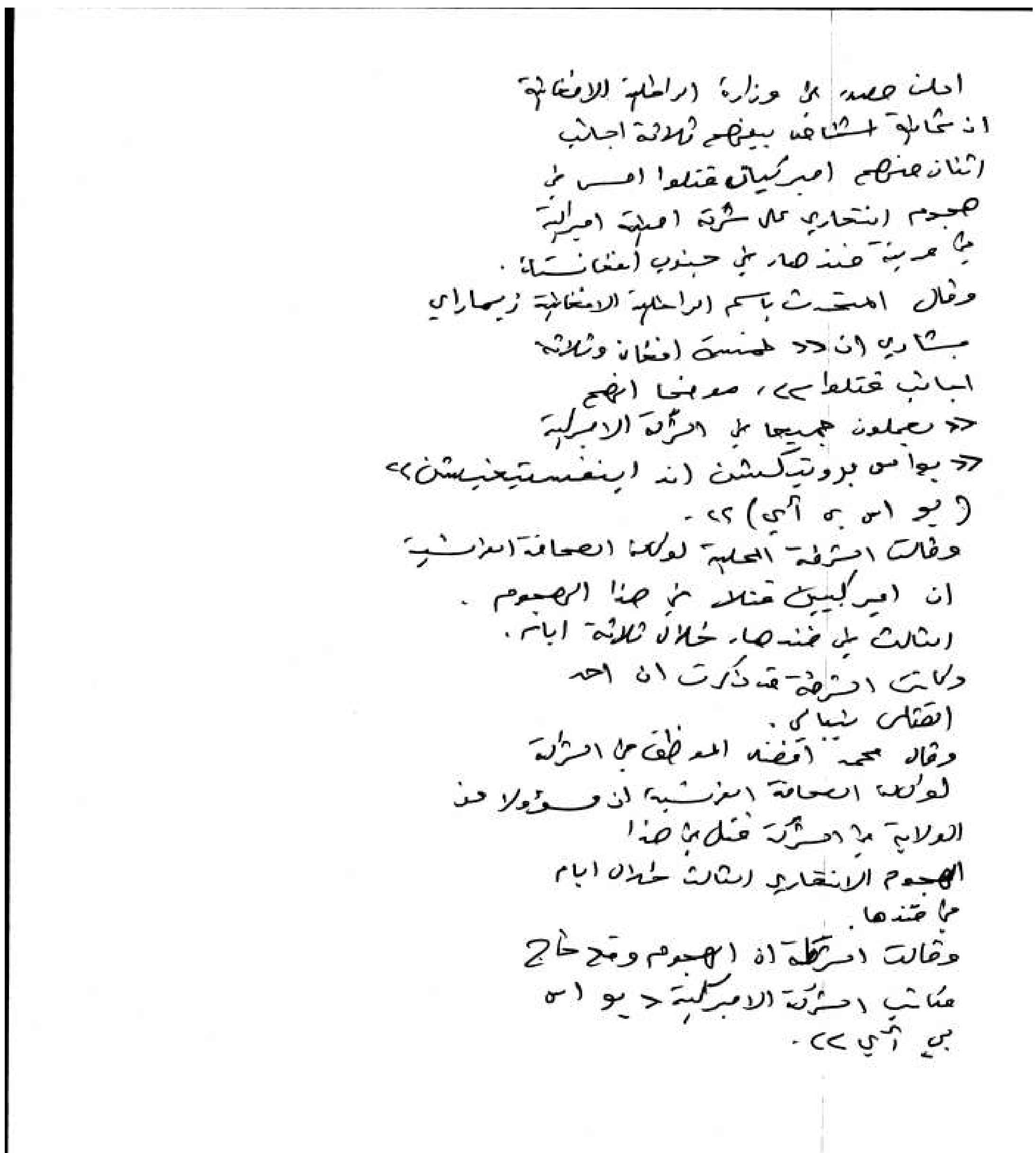
5.2 Tri-character approach

Following The standard procedure that was used in IfN/ENIT, each transcription hypothesis is modeled as an HMM in which emission probabilities are modeled as Bernoulli mixture distributions (BHMMs). To keep the number of independent parameters low, the BHMM at sentence level (transcription hypothesis) is built from BHMMs at character level. Nevertheless, with this database, the BHMM at sentence level is build from BHMMs at character level which depend on their surrounding characters, that is, following a *tri-character model ling approach*.

5.3 Experiments

Experiments in this section were carried out on the NIST OpenHaRT database [21]. Each document image was segmented to fit the two conditions proposed in the NIST OpenHart competition (Word and Line conditions). Word segmentations were supported and they were used to find the line segmentations. Each resulting image was scaled in height to 30 pixels and then binarized using Otsu’s method. Following our strategy, no preprocessing nor feature extraction was applied on images.

Keeping the tri-character approach 5.2 in mind, a list of tri-characters was obtained by taking the first $N \in \{50, 100, 200, 500\}$ frequent ones, that is, if a tri-character T appears more than N time, it will be selected. Selected ones were replaced with those of uni-characters to avoid duplication. This approach improved our results since we



اعلن مصدر في وزارة الداخلية الافغانية
ان ثمانية اشخاص بينهم ثلاثة اجانب
اثنان منهم امريكيان قتلوا امس في
هجوم انتحاري على شركة امنية امريكية
في مدينة قندهار في جنوب افغانستان .
وقال المتحدث باسم الداخلية الافغانية زمراي
بشاري ان << خمسة افغان وثلاثة
اجانب قتلوا >> ، موضحا انهم
<< يعملون جميعا في الشركة الامريكية
<< يو اس بروتيكشن اند اينفستيغيشن >>
(يو اس بي آي) >> .
وقالت الشرطة المحلية لوكالة الصحافة الفرنسية
ان امريكيين قتلا في هذا الهجوم .
الثالث في قندهار خلال ثلاثة ايام .
وكانت الشرطة قد ذكرت ان احد
القتلى نيبالي .
وقال محمد افضل الموظف في الشركة
لوكالة الصحافة الفرنسية ان مسؤولا في
الولايات من الشركة قتل في هذا
الهجوم الانتحاري الثالث خلال ايام
في قندهار .
وقالت الشرطة ان الهجوم وقع خارج
مكاتب الشركة الامريكية < يو اس
بي آي >> .

Figure 5.1: NIST OpenHart example document

model for the character and it's surrounding two characters. We used $N = 500$ to do our experiments since it was the best in results, number of characters, and time consuming. On the other hand, from our previous, preliminary results, some good parameters are used, they are, the window width $W = 9$, and variable number of states with loading factor $F = 0.4$. For the two conditions (word and line), we tried different mixture components per state, $K \in \{4, 16, 32, 64\}$. Training and recognition of test images was performed by using the Viterbi algorithm. Figure 5.2 shows the Word Error Rate (WER%) as a function of segmentation conditions, for varying values of the number of mixture components per state

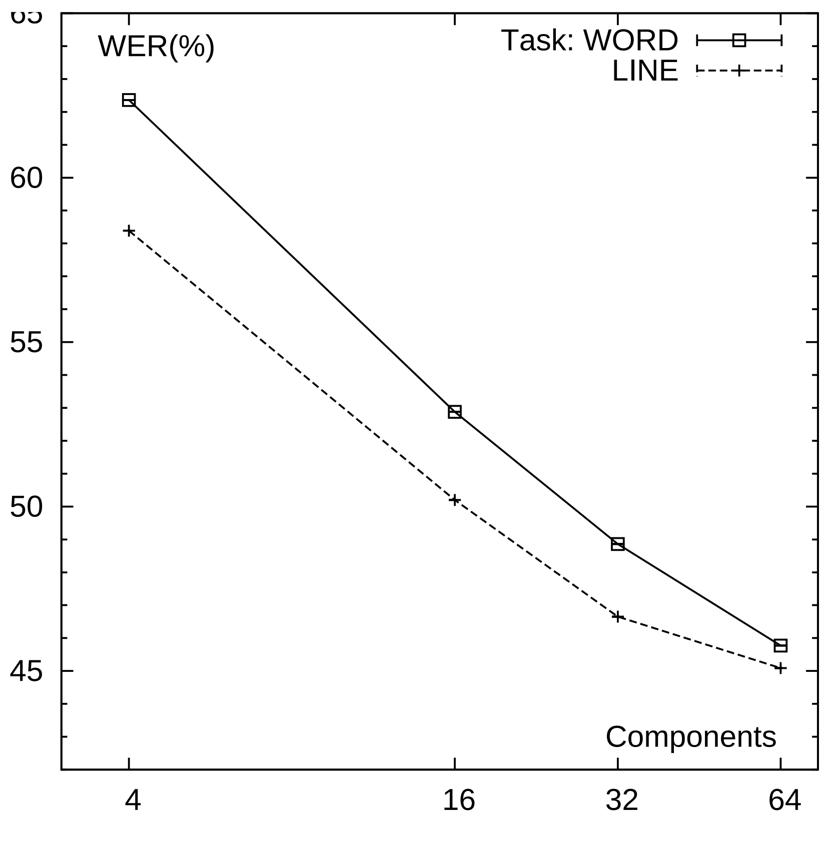


Figure 5.2: WER(%) as a function of the segmentation conditions for varying values of the number of mixture components (K).

From the results in Figure 5.2 it becomes clear that with line segmentation condition WER rate is lower, this improvement could due the use of the n-gram language model. Moreover, the model's complexity is adjusted to the training data by using more mixture components per state. In particular, best result obtained for $K = 64$ on line segmentation condition is 45.09, and on word segmentation condition is 45.77. In the NIST OpenHaRT competition, we could not train for $K > 32$ for the whole data-set, however we got the first place in the line condition with accuracy of 52.54, and second place in word condition with accuracy of 51.06. More details are in the NIST technical report [21].

In order to analyze the high rates of WER in this database, we generated a confusion matrix that shows the very frequent wrong recognized words. Table 7.5 in Appendix section explores more details.

6

Concluding and Remarks

6.1 Conclusions

Embedded Bernoulli HMMs (BHMMs) have been described and tested for Arabic Handwriting Recognition on the the well-known IfN/ENIT database of handwritten Tunisian town names. Apart from our previous basic approach, in which narrow, one-column slices of binary pixels are fed into BHMMs, we have used a sliding window of adequate width to better capture image context at each horizontal position of the word image. Also, we have considered three methods of window repositioning after window extraction so as to help our BHMM-based recognizer in dealing with vertical image distortions. The experiments reported have carefully studied the effects of the window width, the number of states, and repositioning. As expected, the best results have been obtained with an adequate adjustment of the window width, number of states, number of mixture components and, what it seems even more important, (vertical) window repositioning after window extraction. A WER of 6.1% has been achieved on the standard abcd-e partition which, to our knowledge, outperforms the best result known to date.

On the other hand, our new approach, repositioning in windowed BHMMs was tested on the NIST OpenHaRT database for Arabic Handwriting Recognition. This database was introduced by the National Institute of Standards and Technology (NIST) in their 2010 evaluation. Unlike IfN/ENIT database, this database, contains lines of handwritten text. Therefore, the task to be performed is a continuous word recognition task (line condition) and needed a n-gram language model. A WER of 45.09% has been achieved for the line condition on the evaluation set, and a WER of 45.77% for the word condition.

6.2 Scientific Contributions

Parts of this thesis have been published in international workshops, and conferences. In this section, we review these publications pointing out their ranking, and their relation with this thesis.

- *The 3rd Palestinian International Conference on Computer and Information Technology (PICCIT 2010) (research, innovation and entrepreneurship) in East Hebron, Palestine*
  - Ranking: No core
  - Relation with chapter 2
  - Publication: **I. Khoury**, A. Giménez, and A. Juan. Arabic Handwritten Word Recognition Using e Bernoulli Mixture HMMs. In Proc. of the 3rd Palestinian Int. Conf. on Computer and Information Technology (PICCIT 2010), Hebron (Palestine), Mar. 2010.
- *The International Conference on Frontiers in Handwriting Recognition (ICFHR 2010)*
  - Ranking: core B
  - Relation with chapter 3
  - Publication: A. Giménez, **I. Khoury**, and A. Juan. Windowed Bernoulli Mixture HMMs for Arabic e Handwritten Word Recognition. In Proc. of the Int. Conf. on Frontiers in Handwriting Recognition (ICFHR 2010), Kolkata (India), Nov. 2010.
- *The International Competition on Frontiers in Handwriting Recognition*
  - Relation with chapter 3 and 4
  - First Place Prize in the IAPR sponsored Arabic Handwriting Recognition Competition organized during ICFHR 2010 Authors: A. Giménez, **I. Khoury** and A. Juan November 2010

- Springer book: Contributing in writing a chapter of book (not published yet), due the winning of the first Place Prize in the ICFHR competition.
  - Relation with chapter 4
  - Publication: Volker Märgner and Haikal El Abed, Eds., ”Guide to OCR for Arabic Scripts - Development, Evaluation and Improvement”, Advances in Pattern Recognition. Springer Verlag, 2011.
- *The 2010 NIST Open Handwriting Recognition and Translation Evaluation (OpenHaRT 2010)*
  - Ranking: International technical report
  - Relation with chapter 5
  - Competition: Our results were classified as in first place for the line condition HTR, and a second place for the word condition HTR. More details are in `http://www.nist.gov/itl/iad/mig/hart2010.cfm`

6.3 Future Work

Results have shown that Bernoulli HMMs which are fed of a fixed-dimension feature vectors, have obtained better results that those of Gaussian's which are fed of real-valued feature vectors. Working with Bernoulli HMMs is very promising, though still more research work has to be done. For future work, our models could be tested on more databases especially for Arabic. Further improvements can be expected by applying the discriminative training, as well as trying more methods in modeling the variations in the Arabic handwritten text.

7

Appendix

7.1 Arabic Handwriting

Arabic is spoken by 234 million people and important in the culture of many more [15]. It is one of the six United Nations official languages [6, 5, 3, 1]. The characters of Arabic script and similar characters are used by a much higher percentage of the worlds population to write languages such as Arabic, Farsi (Persian), and Urdu. Arabic script differs from Latin scripts in several ways. Unlike English handwriting, Arabic is written from right-to-left and does not distinct between upper or lowercase characters [20].

Although Arabic inscriptions are most common after the birth of Islam (7th century CE), the origin of the Arabic alphabet lies deeper in time. The Nabataeans, which established a kingdom in what is modern-day Jordan from the 2nd century BCE, were Arabs. They wrote with a highly cursive Aramaic-derived alphabet that would eventually evolve into the Arabic alphabet. The Nabataeans endured until the year 106 CE, when they were conquered by the Romans, but Nabataean inscriptions continue to appear until the 4th century CE, coinciding with the first inscriptions in the Arabic alphabet (which is also found in Jordan) [4]. There are many Arabic dialects. Classical Arabic, which is the language of the Qur'an that was originally the dialect of Mecca in what is now Saudi Arabia. An adapted form of this, known as Modern Standard Arabic, is used in books, newspapers, on television and radio, in the mosques, and in conversation between educated Arabs from different countries.

The Arabic alphabet contains 28 consonants. They are annotated according to their position. Characters may come alone (A), or may come connected to another character. They could be connected at the Beginning (B), Middle (M), or even at the End (E). A list of Arabic characters following the IfN/ENTI database (Sec. 2.6.1) are shown in table 7.4. Like other Proto-Sinaitic-derived scripts, Arabic doesn't have letters for vowels. However, there is a system to marking vowels. Short vowels are represented by diacritics above or below a letter, they are: damma, fat-ha,and kasra. Long vowels are represented by using the short-vowel diacritics plus the letters alif, wa:w, and ya: to represent the sounds [a:], [u:], and [i:], respectively. Table 7.1, shows an example of short and large vowels applied to the character meem (م).

Table 7.1: Short vowels: fat-ha, kasra, and damma respectively starting from the left, and long Vowels in Arabic language

Short vowels			Long vowels		
مَ	مِ	مُ	مَا	مِي	مُو
[ma]	[mi]	[mu]	[ma:]	[mi:]	[mu:]

In addition to the vowel markers, Arabic also has several other diacritics. The hamza, which looks like C, denotes the glottal stop . Please note that the letter alef used to represent the glottal stop, but has become more of a placeholder for initial vowel words. The hamza requires a "seat" letter such as alef but also wa:w and ya: to anchor onto. Another diacritic is the suku:n, which looks like a circle and is placed on top of a letter to denote the absence of any vowel. One more diacritic is the madda, which may comes instead of a hamza on the alef letter. Finally, the diacritic shadda, which resembles W, represents the doubling of a consonant. See table 7.2. And finally, Arabic numbers that is also called Indian numbers. See table 7.3

Table 7.2: Examples of different Arabic diacritics

hamza	alef with		yaa	waaw with	meem with	
alone	hamza	madda	hamza	hamza	shadda	sukun
ء	أ إ	آ	يء	وء	مّ	مْ

Table 7.3: Arabic (Indian) numbers

٠	١	٢	٣	٤	٥	٦	٧	٨	٩

7.1.1 Root system in Arabic

Arabic words are constructed from three-letter "roots" which convey a basic idea [2]. For example, k-t-b conveys the idea of writing. Addition of other letters before, between and after the root letters produces many associated words: not only "write" but also "book", "office", "library", and "author".

7.1.2 Advanced Arabic annotation:

Annotation of Arabic characters is very helpful, where more information about the position of each character is given. Annotation of the IfN/ENIT database is not necessary since they are supported with the database. But when using another database, like OpenHaRT 5.1, that is only Arabic characters is given, annotation is needed. We have created a system to annotate all Arabic letters that takes into account the same way of annotating the IfN/ENIT database but, more characters were considered. Experiments shown that by using the annotated text, results were improved.

Table 7.4: Arabic characters (Char) and their annotations (Anno) according to the IfN/ENIT database (see section 2.6.1)

Name	**Alone(A)**		**Begin(B)**		**Middle(M)**		**End(E)**	
	Char	Anno	Char	Anno	Char	Anno	Char	Anno
Alef	ا	aaA					ـا	aaE
Baa	ب	baA	بـ	baB	ـبـ	baM	ـب	baE
Taa	ت	taA	تـ	taB	ـتـ	taM	ـت	taE
Thaa	ث	thA	ثـ	thB	ـثـ	thM	ـث	thE
Jeem	ج	jaA	جـ	jaB	ـجـ	jaM	ـج	jaE
Haa	ح	haA	حـ	haB	ـحـ	haM	ـح	haE
khaa	خ	khA	خـ	khB	ـخـ	khM	ـخ	khE
Daal	د	daA					ـد	daE
Dhaal	ذ	dhA					ـذ	dhE
Raa	ر	raA					ـر	raE
Zaay	ز	zaA					ـز	zaE
Seen	س	seA	سـ	seB	ـسـ	seM	ـس	seE
Sheen	ش	shA	شـ	shB	ـشـ	shM	ـش	shE
Saad	ص	saA	صـ	saB	ـصـ	saM	ـص	saE
Daad	ض	deA	ضـ	deB	ـضـ	deM	ـض	deE
Tta	ط	toA	طـ	toB	ـطـ	toM	ـط	toE
Dhaa	ظ	zaA	ظـ	zaB	ـظـ	zaM	ـظ	zaE
Ayn	ع	ayA	عـ	ayB	ـعـ	ayM	ـع	ayE
Ghyn	غ	ghA	غـ	ghB	ـغـ	ghM	ـغ	ghE
Faa	ف	faA	فـ	faB	ـفـ	faM	ـف	faE
Qaaf	ق	kaA	قـ	kaB	ـقـ	kaM	ـق	kaE
Kaaf	ك	keA	كـ	keB	ـكـ	keM	ـك	keE
Laam	ل	laA	لـ	laB	ـلـ	laM	ـل	laE
Meen	م	maA	مـ	maB	ـمـ	maM	ـم	maE
Noon	ن	naA	نـ	naB	ـنـ	naM	ـن	naE
Haa	ه	heA	هـ	heB	ـهـ	heM	ـه	heE
Waaw	و	waA					ـو	waE
Yaa	ي	eeA					ـي	eeE

7.2 Confusion Matrix

	”	)	،	ألاق	اللل	الله	حن	ف	فى	في	لا	للـ	متى	ممـن	من
”	26	0	4	0	0	0	0	0	1	0	0	0	0	0	0
)	0	5	4	0	0	0	0	0	0	0	0	0	0	0	0
،	0	1	55	0	0	0	0	0	0	0	0	0	0	0	1
آلاف	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
الله	0	0	0	0	5	13	0	0	0	0	0	0	0	0	0
فى	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
في	0	0	0	0	0	0	1	9	8	30	1	0	0	0	1
لا	0	0	0	0	0	0	0	0	0	0	2	4	0	0	0
من	0	0	0	0	0	0	4	0	0	0	0	0	4	3	38

Table 7.5: Confusion Matrix for the very frequent wrong recognized words

References

[1] I. S. I. Abuhaiba, S. A. Mahmoud, and R. J. Green. Recognition of handwritten cursive arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16:664–672, June 1994. 3, 51

[2] Al-bab. The arabic language @ONLINE. 53

[3] B. Al-Badr and S. A. Mahmoud. Survey and bibliography of arabic optical text recognition. *Signal Processing*, 41(1):49 – 77, 1995. 3, 51

[4] AncientScripts. Arabic @ONLINE. 51

[5] M. Cheriet. Visual recognition of arabic handwriting: challenges and new directions. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, SACH'06, pages 1–21, Berlin, Heidelberg, 2008. Springer-Verlag. 3, 51

[6] S. J. David Doermann, David Scott Doermann. *Arabic and Chinese handwriting recognition.* Springer-Verlag Berline Heidelberg, 2008. 3, 51

[7] P. Dreuw, G. Heigold, and H. Ney. Confidence-Based Discriminative Training for Model Adaptaion in Offline Arabic Handwriting Recognition. *ICDAR*, pages 596–600, 2009. 27, 30, 33

[8] A. Giménez and A. Juan. Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 896–900, Barcelona (Spain), July 2009. 9

[9] A. Giménez, I. Khoury, and A. Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition (ICFHR 2010)*, Kolkata (India), Nov. 2010. 38

[10] V. R. Gómez. Multimodal Interactive Transcription of Handwritten Text Images, jun 2010. 6

[11] J. T. Goodman. A bit of progress in language modeling. Technical report, 2001. 4, 6, 14

[12] F. Jelinek. *Statistical methods for speech recognition.* Cambridge, MA: The MIT Press (Language, speech, and communication series), 1997. 6

[13] I. Khoury, A. Giménez, and A. Juan. Arabic Handwritten Word Recognition Using Bernoulli Mixture HMMs. In *Proc. of the 3rd Palestinian Int. Conf. on Computer and Information Technology (PICCIT 2010)*, Hebron (Palestine), Mar. 2010. 29

[14] S.-W. Lee. *Advances in Handwriting Recognition.* World Scientific Publishing Co. Pte. Ltd., 1999. 3

[15] L. Lorigo and V. Govindaraju. Offline Arabic Handwriting Recognition: A Survey. 28(5):712–724, May 2006. 3, 51

[16] V. Märgner and H. E. Abed. ICDAR 2007 - Arabic Handwriting Recognition Competition. In *Proc. of the 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1274–1278, Curitiba (Brazil), sep 2007. 6, 9, 21, 28

[17] V. Märgner and H. E. Abed. ICDAR 2009 Arabic Handwriting Recognition Competition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 1383–1387, Barcelona (Spain), July 2009. 6, 9, 21, 28

[18] V. Märgner and H. E. Abed. ICFHR 2010 Arabic Handwriting Recognition Competition. In *Proc. of the 12th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR 2010)*, pages 709–714, Kolkata (India), Nov. 2010. 21, 38, 39

[19] V. Märgner, M. Pechwitz, and H. E. Abed. ICDAR 2005 Arabic Handwriting Recognition Competition. In *Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR 2005)*, volume 1, pages 70–74, Seoul (Korea), 2005. 21, 25, 31, 33

[20] S. J. Matrikelnummer. Improved Modeling in Handwriting Recognition, jun 2009. 4, 51

[21] NIST. NIST 2010 Open Handwriting Recognition and Translation. Technical report, The National Institute of Standards and Technology (NIST), 2010. 8, 41, 42, 43, 46

[22] M. Pastor. *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Dep. de Sistemes Informàtics i Computació, València, Spain, Oct 2007. 5

[23] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IFN/ENIT - DATABASE OF HANDWRITTEN ARABIC WORDS. In *7th Colloque International Francophone sur l'Ecrit et le Document, CIFED*, pages 21–23, Hammamet (Tunis), oct 2002. 7, 10, 20, 21, 28

[24] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993. 6, 9, 14, 16, 19

[25] V. Romero, A. Giménez, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 539–546. Springer-Verlag, Girona (Spain), June 2007. 3

[26] S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995. 14, 16, 19

[27] R. b. Zhi Qiang Liu, Jinhai Cai. *Handwriting recognition: soft computing and probabilistic approaches*. Springer, 2003. 3