Alkhoury, I.; Giménez Pastor, A.; Juan Císcar, A.; Andrés Ferrer, J. (2013). Arabic Printed Word Recognition Using Windowed Bernoulli HMMs. Lecture Notes in Computer Science. 8156:330-339. doi:10.1007/978-3-642-41181-6_34.

Arabic Printed Word Recognition Using Windowed Bernoulli HMMs

Ihab Khoury, Adrià Giménez, Alfons Juan, and Jesús Andrés-Ferrer

DSIC/ITI, Universitat Politècnica de València,
Camí de Vera s/n, 46022 València, Spain,
{ialkhoury,agimenez,ajuan,jandres}@dsic.upv.es

Abstract. Hidden Markov Models (HMMs) are now widely used for off-line text recognition in many languages and, in particular, Arabic. In previous work, we proposed to directly 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 was to by-pass feature extraction and to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. More recently, we extended the column bit vectors by means of a sliding window of adequate width to better capture image context at each horizontal position of the word image. However, these models might have limited capability to properly model vertical image distortions. In this paper, we have considered three methods of window repositioning after window extraction to overcome this limitation. Each sliding window is translated (repositioned) to align its center to the center of mass. Using this approach, state-of-art results are reported on the Arabic Printed Text Recognition (APTI) database.

Keywords: HTR, Bernoulli HMM, APTI, Arabic, Sliding Window, Repositioning

Introduction

Hidden Markov Models (HMMs) are now widely used for off-line text recognition in many languages and, in particular, languages with Arabic scripts [1, 5–7, 4]. Following the conventional approach in speech recognition [9], 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 [9]. However, in the case of text recognition, there is no such standard and,

indeed, very different sets of features are in use today. In [2] 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 was to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. In [3], we improved our basic approach by using a sliding window of adequate width to better capture image context at each horizontal position of the text image. This improvement, to which we refer as *windowed BHMMs,* achieved very competitive results on the well-known IfN/ENIT database of Arabic town names [8]. More recently, very good results on the Arabic Printed Text Image (APTI) database were achieved by using the same approach, which ranked first at the ICDAR 2011 - Arabic Recognition Competition for printed Arabic text [11].

Although windowed BHMMs achieved good results on IfN/ENIT and APTI, it was clear to us that text distortions are more difficult to model with wide windows than with narrow (e.g. one-column) windows. In order to circumvent this difficulty, we have considered new, adaptative window sampling techniques, as opposed to the conventional, direct strategy by which the sampling window center is applied at a constant height of the text image and moved horizontally one pixel at a time. More precisely, these adaptative techniques can be seen as an application of the direct strategy followed by a *repositioning* step by which the sampling window is repositioned to align its center to the center of gravity of the sampled image. This repositioning step can be done horizontally, vertically or in both directions. Although vertical repositioning was expected to have more influence on recognition results than horizontal repositioning, we decided to study both separately, and also in conjunction, so as to confirm this expectation.

In this paper, the repositioning techniques described above are introduced and extensively tested on an Arabic printed database. In particular, we provide new, state-of-art results on the Arabic Printed Text Image (APTI) database, which clearly outperform our previous results without repositioning [3, 11]. In what follows, we briefly review the Bernoulli mixtures HMMs, maximum likelihood parameter estimation and *windowed BHMMs* repositioning techniques. Then, empirical results are reported after a brief description of the APTI database.

Bernoulli mixture HMMs

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:

$$P(O \mid \mathbf{\Theta}) = \sum_{q_1,\ldots,q_T} \prod_{t=0}^{T} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(\mathbf{o}_t)\,, \tag{1}$$

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 the binary feature vector $\mathbf{o}_t$, when $q_t = j$, follows a Bernoulli mixture distribution for the state j

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} \pi_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} (1 - p_{jkd})^{1 - o_{td}} , \tag{2}$$

where o_{td} is the d-th bit of $\mathbf{o}_t$, π_{jk} is the prior of the k-th mixture component in state j, and p_{jkd} is the probability that this component assigns to o_{td} to be 1.

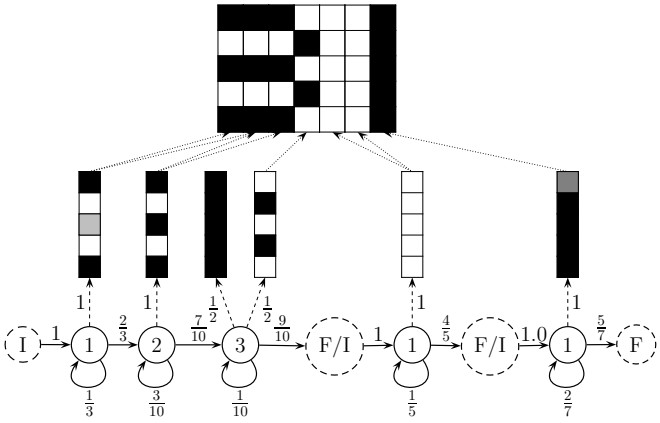

Fig. 1. BHMM examples for the numbers 31, together with binary images generated from it. Bernoulli prototype probabilities are represented using the following color scheme: black=1, white=0,gray=0.5 and light gray=0.1.

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:

$$P(O \mid S, \mathbf{\Theta}) = \sum_{i_1, \ldots, i_{L+1}} \prod_{l=1}^{L} P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) , \tag{3}$$

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

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

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 l-th segment, as given by (1) using the HMM associated with symbol s_l.

An embedded BHMM for the number 31 is shown in Fig. 1, 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. It is worth noting that prototypes do not account for the whole digits 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 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 (3), the probability of generating this image is 0.0004.

Maximum Likelihood Parameter Estimation

Maximum likelihood estimation (MLE) of the parameters governing an embedded BHMM does not differ significantly from the conventional Gaussian case, and it is also efficiently performed using the well-known EM (Baum-Welch) re-estimation formulae [9, 12]. Let $(O_1, S_1), \ldots, (O_N, S_N)$, be a collection of N training samples in which the n-th observation has length T_n, $O_n = (\mathbf{o}_{n1}, \ldots, \mathbf{o}_{nT_n})$, which corresponds to a sequence of L_n symbols ($L_n \leq T_n$), $S_n = (s_{n1}, \ldots, s_{nL_n})$. At iteration r, the E step requires the computation, for each training sample n, of its corresponding forward (α) and backward (β) recurrences (see [9]), as well as

$$z^{(r)}_{nltk}(j) = \frac{\pi^{(r)}_{s_{nl}jk} \prod_{d=1}^{D} {p^{(r)}_{s_{nl}jkd}}^{o_{ntd}} (1 - p^{(r)}_{s_{nl}jkd})^{1-o_{ntd}}}{b^{(r)}_{s_{nl}j}(\mathbf{o}_{nt})}\,, \tag{4}$$

for each t, k, j, l. In (4), $z^{(r)}_{nltk}(j)$ is the probability of $\mathbf{o}_{nt}$ to be generated in the k-th mixture component, given that $\mathbf{o}_{nt}$ has been generated in the j-th state of symbol s_l. The conditional probability function $b^{(r)}_{s_{nl}j}(\mathbf{o}_{nt})$ is analogous to that defined in (2).

In the M step, the Bernoulli prototype corresponding to the k-th component of the state j for character c has to be updated as

$$\mathbf{p}^{(r+1)}_{cjk} = \frac{1}{\gamma_{ck}(j)} \sum_{n} \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi^{(r)}_{nltk}(j) \mathbf{o}_{nt}}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,, \tag{5}$$

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

$$\gamma_{ck}(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,, \tag{6}$$

and $\xi_{nltk}^{(r)}(j)$ is the probability of O_n when the t-th feature vector of the n-th sample corresponds to symbol s_l and is generated by the k-th component of the state j,

$$\xi_{nltk}^{(r)}(j) = \alpha_{nlt}^{(r)}(j) z_{nltk}^{(r)}(j) \beta_{nlt}^{(r)}(j)\,. \tag{7}$$

Similarly, the k-th component coefficient of the state j in the HMM for character c is updated by

$$\pi_{cjk}^{(r+1)} = \frac{1}{\gamma_c(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,, \tag{8}$$

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

$$\gamma_c(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \alpha_{nlt}^{(r)}(j) \beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,. \tag{9}$$

Finally, it is well-known that MLE tends to overtrain the models. In order to amend this problem Bernoulli prototypes are smoothed by linear interpolation with a flat (uniform) prototype, $\mathbf{0.5}$,

$$\tilde{\mathbf{p}} = (1-\delta)\,\mathbf{p} + \delta\,\mathbf{0.5}\,, \tag{10}$$

where δ is usually optimized in a validation set or fixed to a sensible value such as $\delta = 10^{-6}$

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 MLE, although it would be very helpful to better capture the image context at each horizontal position of the image. As an example, Fig. 2 shows a binary image of 4 columns and 5 rows, which is transformed into a sequence of four 15-dimensional feature vectors (first row) 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 4 would be indistinguishable if, as in our previous approach, they were extracted with no context ($W = 1$).

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 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 Fig. 2 in which the standard method (no repositioning) is compared with the three repositioning methods considered.

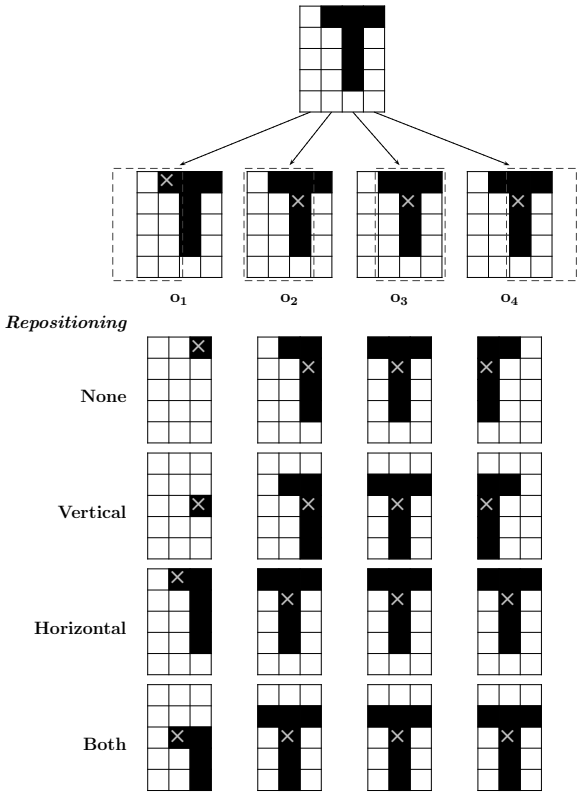

Fig. 2. Example of transformation of a 4×5 binary image (top) into a sequence of 4 15-dimensional binary feature vectors $O = (\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4)$ using a window of width 3. After window extraction (illustrated under the original image), the standard method (no repositioning) is compared with the three repositioning methods considered: vertical, horizontal, and both directions. Mass centers of extracted windows are also indicated.

APTI database

The Arabic Printed Text Image (APTI) database is freely available for non-commercial research [10]. It is a multi-font, multi-size and multi-style database. It was used as a training data in the Arabic Recognition Competition held at ICDAR (Int. Conf. on Document Analysis and Recognition) in 2011 [11]. It comprises 113,284 Arabic words generated in 10 different fonts, 10 different font sizes, and also 4 different styles. For the purpose of evaluating Arabic word recognition systems, APTI is divided into six equilibrated sets labeled as set_1, set_2, ..., set_6. The sixth set is unavailable for the public, and it was used as a testing data in the ICDAR 2011 competition. Each set has different words, but characters are distributed equally.

At the ICDAR 2011 competition, two protocols were defined which differ in the number of fonts used: APTIPC1 and APTIPC2. In APTIPC1, only the Arabic Transparent font was used. In APTIPC2, however, five different fonts were used: Arabic Transparent (Trans), Andalus (Anda), Diwani Letter (Diw), Simplified Arabic (Simp), and Traditional Arabic (Trad). In both protocols, only the *Plain* font style was used, with sizes of 6, 8, 10, 12, 18 and 24. Three systems participated: IPSAR, DIVA-REGIM and UPV-PRHLT (our system).

Experiments

As indicated above, experiments were carried out on the public part of the APTI database [10]. As the public part of APTI does not include set_6, we could not re-run exactly the experiments carried out at the ICDAR 2011 competition. Instead, we defined two new protocols: UPVPC1 and UPVPC2. UPVPC1 is similar to the APTIPC1 protocol described above though, as set_6 was not available, we randomly drew a number of images from all available sets for testing. More precisely, UPVPC1 uses 10000 images for training, 2000 for validation, and 3000 for testing. The UPVPC2 protocol was designed to approximate the ICDAR 2011 competition strategy reported in [11]. It uses sets set_1 to set_4 for training and set_5 for testing. It is worth noting that APTI was divided into six equilibrated sets to allow for flexibility in the composition of development and test sets [10]. Thus, we assume that the results on set_5 should not be very different from those on set_6.

For both protocols, UPVPC1 and UPVPC2, text images were scaled in height to a given dimension of D pixels (for 10 different values of D from 30 to 54) while keeping the aspect ratio. They were then binarized by means of the Otsu algorithm. For each tried value of D, we also tried different values of the sliding window width W (from 1 to 13).

The UPVPC1 protocol was first used to find out, for each font size, appropriate values for the dimension D, window width W, number of states per character Q, and number of mixture components per state K. To this end, a number of experiments were carried out using different values for these parameters. For all experiments, we used a 5-grams language model at character level instead of the

conventional class priors, due to the huge amount of classes. Table 1 shows the best Character Error Rate (CER%) obtained for each size, together with the corresponding system values. From Table 1, it is clear that appropriate values for the parameters are $D \in \{40, 42\}$, $W = 9$, $Q = 7$ and $K = 32$. It is worth noting that the results with sliding windows ($W > 1$) are much better than those obtained with $W = 1$. For instance, for size 6, the best result we had with no window is a CER of 18.4% while, as can be seen in Table 1, it is reduced to a 3.5% with $W = 9$.

Table 1. Character Error Rate (CER%) for different font sizes, using the UPVPC1 protocol. For each CER% reported, the corresponding system values for D, W, Q and K are also provided.

Size	CER%	D	W	Q	K
6	3.5	38	9	7	32
8	2.4	42	9	7	32
10	2.6	40	9	6	32
12	2.3	40	9	6	32
18	1.9	40	11	6	32
24	1.8	42	11	7	32

In the experiments reported above, we did not try window repositioning after window extraction but, as previously mentioned, we think that many recognition errors of our BHMM-based recognizer might be due to its limited capability to properly model vertical image distortions. In order to study the effect of repositioning on the recognition accuracy, the standard method (no repositioning) was compared with the three repositioning methods considered in this work: vertical, horizontal, and both directions. This was done for font size 6, $D = 42$, $W = 9$ and $Q = 7$, with the data partition used in the previous experiments. The resulting CER% is shown in Fig 3, as a function of the grammar scale factor (a weight on the language model to adjust their importance with respect to word-conditional likelihoods). Clearly, vertical (or both) window repositioning outperforms the standard method or horizontal repositioning alone. It is worth noting that the CER% values obtained with no repositioning are approximately halved when using vertical repositioning. In particular, for $G = 70$, the standard method achieves a 3.8% of CER while vertical repositioning reduces this figure to 2.2%. These results are included in Table 2 together with the results obtained using font sizes other than 6. As can be seen in Table 2, the use of vertical repositioning leads to huge CER reductions in relative terms.

In order to compare our results with those reported in the ICDAR 2011 competition, a final experiment was carried out using the UPVPC2 protocol. In this experiment, we used a different recognizer for each font (font-dependent). Each test sample was recognized according to the font-dependent recognizer of higher maximum posterior probability. The results at character and word level

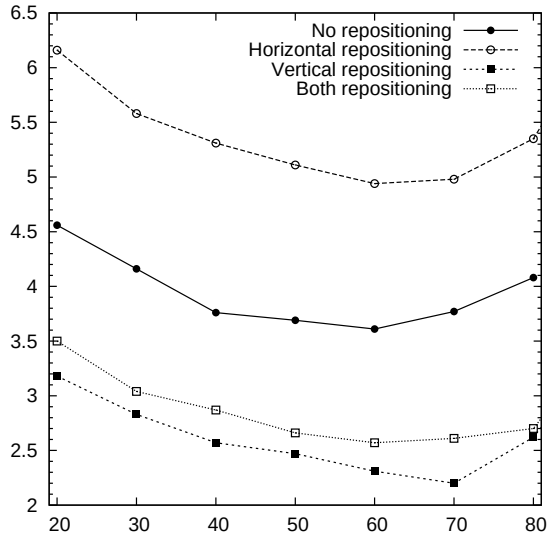

Fig. 3. CER(%) as a function of the grammar scale factor (G) for the different repositioning techniques.

Table 2. Character Error Rate (CER%) using the standard method (no repositioning) and vertical repositioning, for different font sizes.

Repositioning	6	8	10	12	18	24
No	3.8	2.4	2.6	2.3	1.9	1.8
Vertical	2.2	1.0	0.9	0.7	0.2	0.3
Relative reduction %	42	58	65	70	89	83

are shown in Table 3, together with the results obtained at the ICDAR 2011 competition. The system presented in this work would rank first.

Table 3. Character Error Rate (CER%) and Word Error Rate (WER%) for all participated systems in the ICDAR 2011 - Arabic Recognition Competition. Results are shown for different fonts.

System		Trans	Anda	Diw	Simp	Trad
IPSAR system	CER	9.8	13.0	22.5	10.2	25.1
	WER	41.2	46.2	63.0	41.0	64.5
UPV-PRHLT-REC	CER	3.6	3.3	17.2	2.8	15.4
	WER	16.6	16.2	56.1	13.1	54.6
DIVA-REGIM(APTIPC1)	CER	0.9	-	-	-	-
	WER	5.2	-	-	-	-
UPV (this work)	CER	**0.3**	**0.5**	**8.3**	**0.4**	**3.0**
	WER	**1.7**	**2.4**	**37.0**	**1.9**	**13.2**

Concluding Remarks

Windowed Bernoulli mixture HMMs (BHMMs) for Printed Arabic word recognition have been described and improved by the introduction of window *repositioning* techniques. In particular, we have considered three techniques of window *repositioning* after window extraction: *vertical, horizontal,* and *both.* The only differ in the way in which extracted windows are shifted to align mass and window centers (only in the vertical direction, horizontal direction or in both directions). These techniques have been carefully described and extensively tested on the APTI database for multi-font, multi-size and multi-style Arabic Printed text images. In all cases, state-of-art results were obtained with vertical repositioning.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 287755. Also supported by the Spanish Government (Plan E, iTrans2 TIN2009-14511 and AECID 2011/2012 grant).

References

1. M. Dehghan et al. Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM. *Pattern Recognition*, 34(5):1057–1065, 2001.
2. A. Giménez and A. Juan. Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In *ICDAR 2009*, pages 896–900, Barcelona (Spain), july 2009.
3. Adrià Giménez, Ihab Khoury, and Alfons Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *ICFHR 2010*, pages 533–538, Kolkata (India), November 2010.
4. Emmanuèle Grosicki and Haikal El Abed. ICDAR 2009 Handwriting Recognition Competition. In *ICDAR '09*, pages 1398 – 1402, Barcelona (Spain), july 2009.
5. Simon Günter et al. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.
6. Volker Märgner and Haikal El Abed. ICDAR 2007 - Arabic Handwriting Recognition Competition. In *ICDAR 2007*, pages 1274–1278, Curitiba (Brazil), sep 2007.
7. Volker Märgner and Haikal El Abed. ICDAR 2009 Arabic Handwriting Recognition Competition. In *ICDAR 2009*, pages 1383–1387, Barcelona (Spain), July 2009.
8. M. Pechwitz et al. IFN/ENIT - database of handwritten Arabic words. In *CIFED '02*, pages 21–23, Hammamet (Tunis), oct 2002.
9. L. Rabiner and B. Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.
10. F. Slimane et al. A new arabic printed text image database and evaluation protocols. In *ICDAR 2009.*, pages 946 –950, july 2009.
11. F. Slimane et al. ICDAR 2011 - arabic recognition competition: Multi-font multi-size digitally represented text. In *ICDAR 2011 - Arabic Recognition Competition*, pages 1449–1453. IEEE, September 2011.
12. S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.