

# A New OCR System Similar to ASR System

## **Abstract**

Optical character recognition (OCR) system is created using the concepts of automatic speech recognition where the hidden Markov Model is widely used. Results are very encouraging. Recognition accuracy in the range 98 % to 99 % has been achieved.

## **1. Introduction**

Optical character recognition systems are used to convert text in printed documents into machine-readable ASCII codes for the purpose of storage or further processing. The range of applications is wide and includes postal code reading, automatic data entry into large administrative systems, recognition of print and script, automated cartography, banking, and reading devices for blind. It is an integral part of office automation in the form of text processing. The features, which characterize a good optical character recognition system, are accuracy, flexibility, and speed (El-Khaly, and Sid-Ahmed, 1990).

In the last two decades, there has been progress in the field of machine recognition techniques for Latin and Chinese characters. Meanwhile, the machine recognition of Arabic characters has not yet been fully explored. Despite the fact that Arabic characters are used in writing many widespread languages like Arabic, Persian, Urdu, etc. The published literature includes very little information about the computer recognition of Arabic characters. However, these forms of printed text are usually presented as isolated characters. The shape of the character in these forms is independent of its location in the printed word, contrary to Arabic text (Al-Youfafi and Upda, 1992, El-Khaly, and Sid-Ahmed, 1990, and El-Wakil and Shoukry, 1988).

In order to illustrate the difference between Arabic text and various other texts, the main characteristics of Arabic characters are provided next:

- (i) Arabic is written cursively from right to left.
- (ii) Arabic words consist of one or more connected portions.
- (iii) The discontinuities between portions are due to some characters that are not connectable from the left side with the succeeding portion.
- (iv) Every character has more than one shape depending on its position within a connected portion of the word (beginning, middle, end of portion, or isolated character).
- (v) There are a lot of diacritics that may be associated with Arabic text.

Hidden Markov Models (HMMs) have been used successfully for speech recognition applications and they have recently been applied to the problem of on-line handwriting recognition as well. In this thesis we will use the HMMs for solving the problem of off-line automatic recognition of connected Arabic text.

To use the HMM techniques for the application of Arabic connected character recognition, we have to add some modifications to overcome the difference between speech signals and Arabic connected text. My idea is to scan the Arabic documents and to remove noise and discontinuities from the scanned images using preprocessing algorithms. Then we will look into the text-image using a sliding window that moves from right to left through text lines and take features of each window. These features are used to train the Hidden Markov Models of the Arabic patterns.

## **2. Hidden Markov Models Overview:**

Hidden Markov models (HMMs) are a method of modeling systems with discrete, time dependant behavior characterized by common, short time “processes” and transitions between them.

An HMM can be thought of as a *finite state machine* where the transitions between the states are dependant upon the occurrence of some “symbol”. Associated with each state transition is an *output probability distribution*, which describes the probability with which a symbol will occur during the transition, and a *transition probability* indicating the likelihood of this transition.

Since the late 1970’s, when HMMs were first applied to automatic speech recognition (ASR), several analytical techniques have been developed for estimating these probabilities. These techniques enabled HMMs to become more computationally efficient, robust, and flexible. This has permitted HMMs to serve as the basis of many large-vocabulary, speaker-independent systems.

The HMM model describes a stochastic process, which produces a sequence of observed events or symbols. The HMM is called “Hidden” Markov Model because there is an underlying stochastic process that is not observable, but affects the observed sequence of events. The sequence of observations can be described using the notation:

$$O_1^T = O_1, O_2, \dots, O_t, \dots, O_T,$$

where the process has been observed for the T discrete time steps from time  $t=1$  to  $t=T$ . The observations may be any one of  $K$  symbols,  $v_k$ . In speech recognition, these symbols may be “low-level” vector quantization (VQ) codebook entries computed at regular intervals, or “high-level” acoustic descriptors such as phonemes.

The HMM may be thought of as a model of the process which produces the sequence of acoustic events belonging to a specific class, or word, in the vocabulary. Variations between observation sequences of the same class, such as word length and pronunciation, are modeled by the underlying stochastic nature of the HMM. An HMM-based ASR will generally have  $C$  HMMs, where  $C$  is the number of classes. The HMM recognizes the word when the final state is reached.

The key parameters to be determined in an HMM-based ASR system are the number of states per word,  $N_c$ , and the state transition and symbol output probability matrices. Large amounts of training data (per word) are needed to obtain acceptable estimates of these probability density functions. These probability density functions capture the statistics of the talker population and are more robust for speaker-independent recognition than the finite number of word templates used in the dynamic time warping (DTW) algorithm.

Each HMM consists of a number of *states*. When the model is in state  $s_i$ , the process may be measured and one of the symbols  $v_k$  may be produced, according to an *observation probability distribution*,  $b_i(v_k)$ . At each time step  $t$ , the model will undergo a transition to a new state,  $s_j$ , according to a *transition probability distribution*, given by  $a_{ij}$ . These transitions may take place between any two states, including self-transitions, as long as the transition probability distribution is non-zero. Therefore, the HMM is a *Markov process* because the probability of being in a particular state at time  $t+1$ , given the state sequence prior to time  $t$ , depends only on the state at time  $t$ . An illustration of a typical HMM is provided in Figure 1.

A particular model may be characterized by the number of states  $N_c$ , the number of observations  $K$ , and the three probability densities  $\pi = \{\pi_i\}$ ,  $A = \{a_{ij}\}$ , and  $B = \{b_i(v_k)\}$ , where  $\pi_i$  is the initial probability distribution across states. The model of the  $c^{th}$  word,  $M^c$ , can be characterized by the set of  $M^c = \{\pi^c, A^c, B^c\}$ . Pattern recognition with an HMM is equivalent to selecting the single model  $M^{c*}$  from the set  $\{M^1, M^2, \dots, M^C\}$  which maximizes the probability of observation sequence  $o_1^T$ :

$$C^* = \operatorname{argmax}_{1 < c < C} [Pr(o_1^T / M^c)].$$

For simplicity,  $Pr(o_1^T / M^c)$  will simply be referred to as  $Pr(O/M)$ . The following sections describe how these three probability

densities can be calculated and how an efficient implementation can be realized.

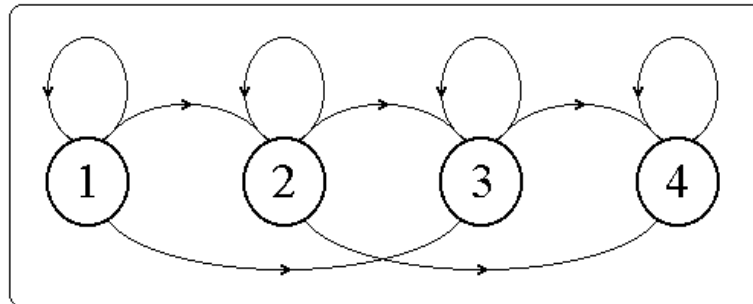


Figure 1: An illustration of a left-to-right hidden Markov model.

### Constructing an HMM

The selection of the optimal number of states, which properly describe the observed sequence of events for a given word is a somewhat empirical process. For discrete words, one might select a number of states, which roughly correspond to the number of phonemes in the word to be modeled, with additional states corresponding to beginning and ending silences.

When constructing an HMM, the probability distributions  $A$  and  $B$  cannot be determined using a direct analytical solution. However, an iterative procedure, known as forward-backward algorithm, or *Baum-Welch algorithm*, may be used for estimating these distributions. It has been shown that the Baum-Welch recursive algorithm can compute a new word model,  $M^{c'}$  =  $\{\pi^{c'}, A^{c'}, B^{c'}\}$ , from an initial model  $M^c$  =  $\{\pi^c, A^c, B^c\}$ , which either increases the probability of the sequence given the model, or leaves the probability estimates unchanged ( $M^c = M^{c'}$ ). If the probabilities remain unchanged, then  $M^c$  is at a limiting point of the likelihood function.

The forward-backward algorithm is a computationally efficient method for determining the model parameters. This iterative procedure uses the *forward probabilities*,  $\alpha_i(t)$ , and *backward probabilities*,  $\beta_j(t)$ , to update the observation probabilities,  $b_i(v_k)$ , and transition probabilities  $a_{ij}$ . It is dependant upon the computation of the probability of a partial

observation at time  $t < T$  of  $O_t^t$  and state  $s_i$  for a specific model  $M^c$  with  $N_c$  states. In general, the forward probability is given by:

$$\alpha_i(t) = Pr(O_t^t, s_i / M),$$

and is computed by induction using an initial estimate of  $a_{ij}$  and  $b_i(v_k)$ :

1.  $\alpha_i(0) = \pi_i, 1 \leq i \leq N$ ;
2. and for  $1 \leq t \leq T - 1$  and  $1 \leq j \leq N$ :

$$\alpha_i(t) = \left( \sum_{j=1}^N \alpha_j(t-1) a_{ji} \right) b_i(O_t).$$

From the forward probabilities, the probability of the complete observation sequence at time  $T$  can be calculated:

$$Pr(O / M) = \sum_{i=1}^N \alpha_i(T).$$

Thus, the forward algorithm calculates the probability that an observable sequence was generated by model  $M$ .

An analogous backward probability can also be computed and is essential for updating the estimates of  $a_{ij}$  and  $b_j(v_k)$ . The probability of the partial sequence  $O_{t+1}^T$ , given state  $s_j$  at time  $t$  is given by:

$$\beta_i(t) = Pr(O_{t+1}^T / s_i, M).$$

This backward probability is computed in a similar fashion to the forward probability:

1.  $\beta_j(T) = 1, 1 \leq j \leq N$ ;
2. and for  $T-1 \geq t \geq 1$  and  $1 \leq i \leq N$ :

$$\beta_i(t) = \sum_{j=1}^N \beta_j(t+1) a_{ij} b_j(O_{t+1}).$$

The probability of the complete sequence given the model can be computed with either the forward probabilities alone or from both the forward and backward probabilities:

$$Pr(O / M) = \sum_{i=1}^N \sum_{j=1}^N \alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1).$$

Once the forward and backward probabilities are calculated, the word model  $M' = \{\pi', A', B'\}$ , can be reestimated. For a specific model  $M$ , the reestimation formulas are given by:

$$\pi_i' = \gamma_i(1),$$

where  $\gamma_i(t)$ , the probability of state  $s_i$  at time  $t$ , given the sequence  $O$ , is given by:

$$\gamma_i(t) = \alpha_i(t) \beta_i(t) / Pr(O | M).$$

Likewise,  $A$  can be updated,

$$a'_{ij} = \sum_{t=1}^{T-1} \zeta_{ij}(t) / \sum_{t=1}^{T-1} \gamma_i(t),$$

where  $\zeta_{ij}(t)$ , the probability of the state  $s_i$  at time  $t$  and a transition to state  $s_j$  at time  $t+1$ , is given by:

$$\zeta_{ij}(t) = \alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1) / Pr(O | M).$$

Finally,  $B$  can be updated,

$$b'_j(v_k) = \sum_{t \in (O_t=v_k)}^T \gamma_j(t) / \sum_{t=1}^T \gamma_j(t),$$

where the sum in the numerator is over all instances where the observation at  $t$  was the symbol  $v_k$ . This type of iterative procedure is called an "estimate-maximize" algorithm, and is characterized by an efficient "hill-climbing" procedure for determining the word-model parameters (Rabinar, 1989 and Morgan and Scofield, 1992).

### **HMM Recognition: The Viterbi Algorithm**

HMM recognition can begin once the word-model parameters are determined for all  $C$  words. During recognition, the input symbols generate a particular sequence of states which are visited by the HMM in producing the observation sequence. The state sequence essentially represents the segmentation of the words modeled by the HMM. However, to ensure that the optimal state sequence with the highest *posteriori* probability is selected, the *Viterbi algorithm* is employed. The Viterbi algorithm is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time Markov process. It is essentially a dynamic programming procedure, which may be used to compute the state sequence with the highest priority.

The Viterbi algorithm consists of the following three steps:

(1) Initialize the probability,  $\phi$ , of state  $s_j$  and observation  $O_1$ , and let  $\psi_j(t)$  represent the state at time  $t$ , for which, a maximization over previous states has been performed:

$$\begin{aligned}\phi_j(1) &= \pi_j b_j(O_1), \text{ for } 1 \leq j \leq N; \\ \psi_j(1) &= 0.\end{aligned}$$

(2) For  $2 \leq t \leq T$  and  $1 \leq j \leq N$ , recursively compute new values:

$$\begin{aligned}\phi_j(t) &= \max_{1 < i < N} [\phi_i(t-1) a_{ij}] b_j(O_t), \\ \psi_j(t) &= \operatorname{argmax}_{1 < i < N} [\phi_i(t-1) a_{ij}] b_j(O_t).\end{aligned}$$

(3) Then the single path with the highest probability can be computed and maximum likelihood state sequences can be recovered from  $\psi_j^{*T}(t)$ :

$$\begin{aligned}Pr(O | M) &= \max_{1 < j < N} [\phi_j(T)], \\ j_T^* &= \operatorname{argmax}_{1 < j < N} [\phi_j(T)].\end{aligned}$$

### Practical Application of HMMs

In the preceding formulas for computing model parameters, the forward and backward probabilities  $\alpha_i(t)$  and  $\beta_i(t)$  appear. However, Levinson et al. note that these probabilities tend to zero exponentially as  $T \rightarrow \infty$ , and in practice these values must be scaled to avoid under-flow. The scaling procedure involves multiplication of  $\alpha_i(t)$  and  $\beta_i(t)$  by a scaling coefficient  $\lambda(t)$ , selected such that  $\lambda(t) \alpha_i(t)$  and  $\lambda(t) \beta_i(t)$  remain non-zero in the computation, and

$$\lambda(t) = \left[ \sum_{i=1}^N \alpha_i(t) \right]^{-1}.$$

After reestimating the model parameters, the scale factors may be factored out and canceled. Unfortunately, the scale factors may not be factored out of the computation of the model probability, given by:

$$Pr(O | M) = \sum_{i=1}^N \alpha_i(T).$$

However, under re-scaling:

$$\sum_{i=1}^N \alpha_i(T) \rightarrow \left( \prod_{t=1}^T \lambda(t) \right) \sum_{i=1}^N \alpha_i(T) = 1,$$

$$\prod_{t=1}^T \lambda(t) Pr(O | M) = 1,$$

$$Pr(O | M) = 1 / \prod_{t=1}^T \lambda(t).$$

The log of the probability can then be expressed as:

$$\log(Pr(O | M)) = -\sum_{t=1}^T \log[\lambda(t)].$$

Rabiner and Juang have noted that in ASR applications, either the forward-backward procedure or the Viterbi algorithm may be used to determine  $Pr(O/M)$ . The summation in the forward-backward procedure is usually dominated by the single term, which the maximization step in the Viterbi algorithm selects. As a result, the scaling procedure may be avoided since the Viterbi algorithm can be modified to compute  $\log(Pr(O/M))$  directly. In this case the initial (log) probability is given by:

$$\phi_j(1) = \log(\pi_i b_j(O_1)),$$

And the recursion becomes:

$$\phi_j(t) = \max_{1 < i < N} [\phi_i(t-1) + \log(a_{ij})] + \log(b_j(O_t)).$$

Then the log probability computed with the Viterbi algorithm for each word is:

$$\log(Pr(O/M)) = \max_{1 < j < N} [\phi_j(T)].$$

The implementation of  $\log$  values permits the computation of word probabilities to be additive rather than multiplicative, which avoids computer “underflow” and speeds up the recognition process considerably. In general, the HMM computational complexity is  $O(N^2)$  in the number of states searched by the Viterbi algorithm. Finally, it should be noted that there are a considerable number of HMM variants, which have been formulated to deal with specific continuous speech recognition (CSR) problems (Rabinar, 1989 and Morgan and Scofield, 1992).

### 3. OCR System Architecture

The OCR system is divided into two subsystems: trainer and recognizer. The trainer subsystem is used to initialize and train hidden Markov models using many scanned bitmaps and their corresponding text files. The block diagram of the trainer subsystem is shown in Figure 2.





with our OCR system, but invariant moments described by AlKhaly et al. [AlKhaly88] provided the best performance.

The vector quantizer module provides a series of observations to the labeler module using the input train of feature vectors and a codebook constructed from many images of the training set of documents representing the Arabic font.

The labeler module provides a series of observations with their associated sequence of patterns to the HMM trainer. The HMM trainer uses these inputs to adapt the values of the transition probability matrices and the observation probability matrices of models of these patterns to maximize the probability of the observation sequences of these patterns as described in the HMM section.

The recognizer subsystem uses the trained HMMs to recognize the unseen documents of the same font used for training the OCR system. The block diagram of the recognizer subsystem is shown in Figure 6.

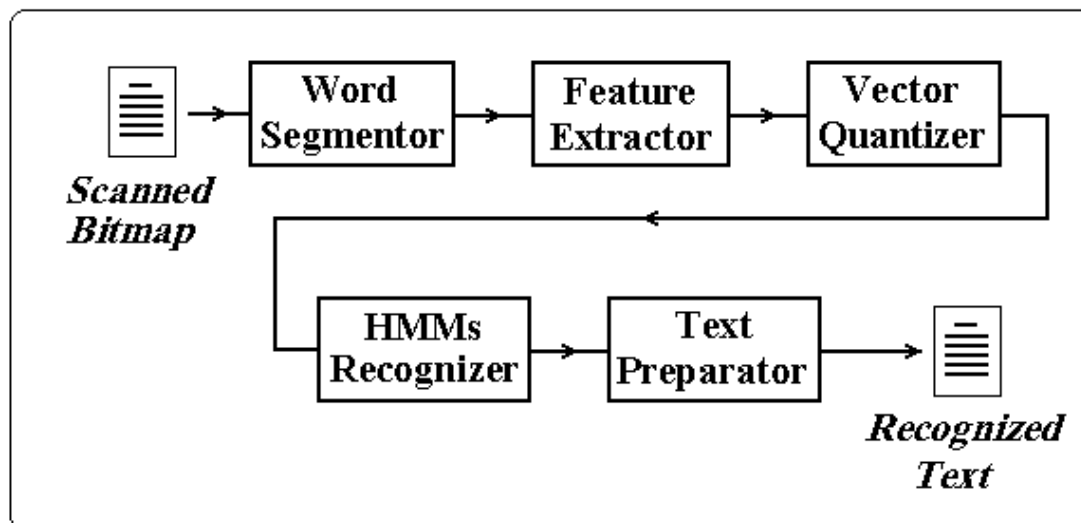


Figure 6: The OCR Recognizer Subsystem.

The Segmentor module segments the scanned bitmap into words, while the Feature Extractor and Vector Quantizer

modules generate a series of observations representing the segmented words as described in the training cycle. The HMMs Recognizer uses the trained HMMs of the selected Arabic font to generate a series of patterns that maximizes the probability of the observation sequence. Finally, the text preparator module converts the series of recognized patterns into their equivalent ligatures and stores these ligatures in a text file.

#### 4. Experimental Results

The OCR system was built using C++ language, and used for training the Simplified Arabic font using the set of ligatures shown in Figure 3. Twenty-five text pages containing these ligatures were printed using a laser printer and scanned using a monochrome scanner with 300 DPI resolution. 21 pages of this set are used for training and four pages are used for testing the system. The following table represents the system recognition accuracy for different codebook sizes and different number of states per model.

Codebook Size	Number of States	Recognition Accuracy
32	10	98.85 %
32	11	98.54 %
32	12	98.94 %
32	13	<b>99.11 %</b>
32	14	98.76 %
64	10	99.07 %
64	11	<b>99.11 %</b>
64	12	98.94 %
64	13	99.07 %
64	14	98.45 %
96	10	98.76 %
96	11	98.80 %
96	12	98.94 %
96	13	<b>99.03 %</b>
96	14	98.72 %

**Table 1: Test Results of Simplified Arabic Font**

The system was also used for training the Traditional Arabic font using the set of ligatures shown in Figure 4 and provided the following results:

Codebook Size	Number of States	Recognition Accuracy
32	8	97.27 %
32	9	97.22 %
32	10	<b>97.51 %</b>
32	11	97.08 %
32	12	95.95 %
64	8	97.22 %
64	9	<b>98.21 %</b>
64	10	98.07 %
64	11	97.88 %
64	12	96.05 %
96	8	97.36 %
96	9	97.93 %
96	10	97.98 %
96	11	<b>98.35 %</b>
96	12	96.66 %

Table 2: Test Results of Traditional Arabic Font

## 5. Conclusion and Future work

In this paper, I used HMM techniques for building a connected Arabic OCR system, and tested this system using the Simplified Arabic and Traditional Arabic fonts of the Microsoft Word application. The system provided a very high accuracy (99.11 % for the Simplified Arabic font and 98.35 % for the Traditional Arabic font). The first font provided better results because it contains less number of ligatures.

In the future, the OCR system will be modified to be used for recognizing Connected Handwritten Arabic text. A robust segmentor has to be designed, and another set of features has to be used for this job. Also a language model has to be added for increasing the recognition accuracy of the system. The system also is slow, the recognition rate is about 10 words/sec. On PII, 333 MHz configuration. More experiments have to be done to increase the recognition speed.

**References:**

- Al-Yousefi, H. and Upda, S.: “*Recognition of Arabic Characters*”, IEE Trans. Pattern Analysis and Machine Intelligence, pp. 853-857, Vol. 14, No. 8, 1992.
- David P. Morgan and Christopher L. Scofield, “*Neural Networks and Speech Processing*”, Kluwer Academic Publishers, 1992, pp. 119-127.
- El-Khaly, F., and Sid-Ahmed, M., “*Machine Recognition of Optically Captured Machine Printed Arabic Text*”, Pattern Recognition, Vol. 23, No. 11, pp. 1207-1214, 1990.
- El-Wakil, M.S., and Shoukry, A.A.: “*On-Line Recognition of Handwritten Isolated Arabic Characters*”, Pattern Recognition, Vol.22, No. 2, pp. 97-105, 1989.
- Rabinar, L.R., “*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*”, Proceedings of the IEEE, vol. 77 (2), February 1989.