Mono-font Cursive Arabic Text Recognition Using Speech Recognition System

M.S. Khorsheed

Computer & Electronics Research Institute,
King AbdulAziz City for Science and Technology (KACST)
PO Box 6086, Riyadh 11442, Saudi Arabia
mkhorshd@kacst.edu.sa

Abstract. This paper presents a system to recognise cursive Arabic typewritten text. The system is built using the Hidden Markov Model Toolkit (*HTK*) which is a portable toolkit for speech recognition system. The proposed system decomposes the page into its text lines and then extracts a set of simple statistical features from small overlapped windows running through each text line. The feature vector sequence is injected to the global model for training and recognition purposes. A data corpus which includes Arabic text of more than 100 $A4-$size sheets typewritten in Tahoma font is used to assess the performance of the proposed system.

1 Introduction

Among the branches of pattern recognition is the automatic reading of a text, namely, text recognition. The objective is to imitate the human ability to read printed text with human accuracy, but at a higher speed.

Most optical character recognition methods assume that individual characters can be isolated, and such techniques, although successful when presented with Latin typewritten or typeset text, cannot be applied reliably to cursive script, such as Arabic. Previous research on Arabic script recognition has confirmed the difficulties in attempting to segment Arabic words into individual characters [1].

Hidden Markov Models (HMMs) [2] are among other classification systems that are used to recognise character, word or text. They are statistical models which have been found extremely efficient for a wide spectrum of applications, especially speech processing. This success has motivated recent attempts to implement HMMs in character recognition whether on-line [3] or off-line [4]. The HMM provides an explicit representation for time-varying patterns and probabilistic interpretations that can tolerate variations in these patterns. In off-line recognition systems, the general idea is to transform the word image into a sequence of observations. The observations produced by the training samples are used to tune the model parameters whereas those produced by the testing samples are used to investigate the system performance.

HMMs have been also used to Arabic word recognition. Following are the approaches introduced in twofold: the global approach and the analytical approach. The global approach treats the word as a whole. Features are extracted from the unsegmented word and compared to a model [5]. The analytical approach decomposes the word into smaller units, which may correspond to a character or part of a character [6]. Another research [7] proposed a system which depends on the estimation of character models, a lexicon, and grammar from training samples. The training phase takes scanned lines of text coupled with the ground truth, the text equivalent of the text image, as input. Then, each line is divided into narrow overlapping vertical windows from which feature vectors are extracted. The character modelling component takes the feature vectors and the corresponding ground truth and estimates the character models. The recognition phase follows the same step to extract the feature vectors which are used with different knowledge sources estimated in the training phase to find the character sequence with the highest likelihood $P(O|\lambda)$.

This paper presents a HMM–based system to recognise cursive Arabic script offline. Statistical features are extracted from the text line image and fed to the recogniser. The system is built on the Hidden Markov Models Toolkit (HTK) [8]. This is primarily designed for building HMM-based speech processing tools in particular recognisers. The proposed system is lexicon free and it depends on the technique of character models and grammar from training samples.

2 System Overview

Fig 1 shows the block diagram of the proposed system. The global model is a network of interconnected character models. Each character-model represents a letter in the alphabet. The system may be divided into stages. The first stage is performed prior to HTK, and includes: image acquisition, preprocessing and feature extraction. The objective is to acquire the document image, preprocess it and then decompose it into text line images. Each line image is transfered into a sequence of feature vectors. Those features are extracted from overlapping vertical windows along the line image, then clustered into discrete symbols.

Stage two is performed within HTK. It couples the feature vectors with the corresponding ground truth to estimate the character model parameters. The

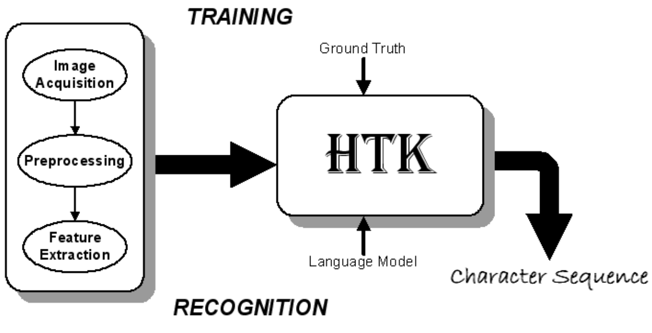


Fig. 1. A block diagram of the HTK-based Arabic recognition system

final output of this stage is a lexicon–free system to recognize cursive Arabic script. During recognition, an input pattern of discrete symbols representing the line image is injected to the global model which outputs a stream of characters matching the text line.

2.1 Feature Extraction

This research implements HMMs to recognise the input pattern. This implies that the feature vector, extracted from the text, is computed as a function of independent variable. In speech, the cepstral features are extracted from the speech signal with respect to time. Similarly, in on–line handwritten recognition a feature vector is computed as a function of time also. In off–line recognition system the case is different; there is no independent variable. Moreover, the whole page image needs to be recognised. In this research, the text line has been chosen as the unit for training and recognition purposes.

Now, assuming that the horizontal position along the text line is the independent variable, a sliding window is scanning the line from right to left. A set of simple features is extracted from pixels falling within that window. The resulting feature vector is mapped against predefined codebook vectors, and replaced with the symbol representing the nearest codebook vector. This step transfers the text line image into a sequence of discrete symbols.

Each line image is divided into overlapped narrow windows, see Fig 2. These windows are then vertically divided into cells where each cell includes a predefined number of pixels. Those cells are used for feature extraction.

Features extracted from the text could be structural [9], spectral [10] or as per here statistical. Statistical features are easy to compute and script independent. They avoid any segmentation at word or character level. These features are:

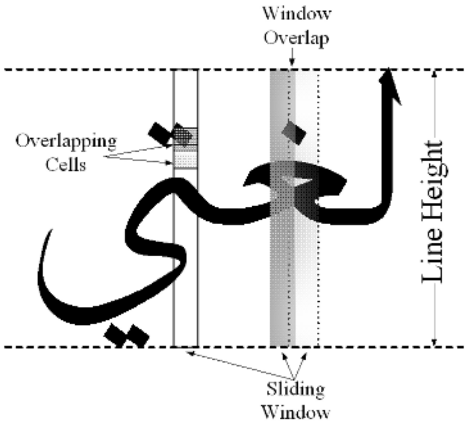


Fig. 2. Dividing the line into windows and cells

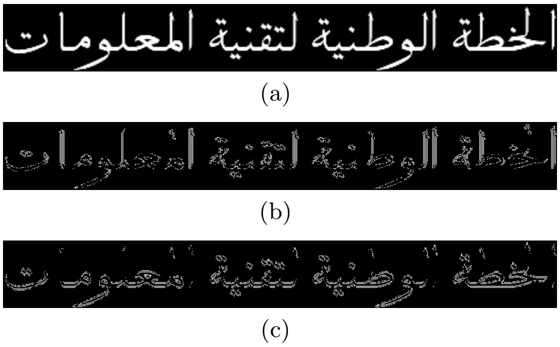


Fig. 3. Feature extraction: a)original line image. b)vertical derivative of (a). c)horizontal derivative of (a).

intensity, intensity of horizontal derivative and *intensity of vertical derivative*, see Fig 3.

The intensity feature represents the number of ones in each cell. The intensity of horizontal derivative detects the edge through X–axis, then computes the number of ones in the resulting cell. The intensity of vertical derivative detects the edge through Y–axis, then computes the number of ones in the resulting cell. The feature vector of one narrow window is built by stacking features extracted from each cell in that window.

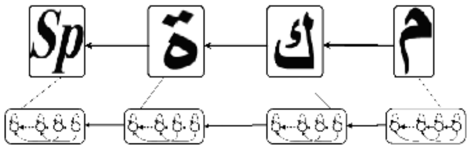


Fig. 4. HMM structure of the word *Makkah* "¡mkT¿", *sp* denotes space character

2.2 HTK Inference Engine

The hidden Markov model toolkit (HTK) [8] is a portable toolkit for building and manipulating hidden Markov models. It is primarily designed for building HMM–based speech recognition systems. HTK was originally developed at the Speech Vision and Robitics Group of the Cambridge University Engineering Department (CUED).

Much of the functionality of HTK is built into the library modules available in C source code. These modules are designed to run with the traditional command line style interface, so it is simple to write scripts to control HTK tools execution. The HTK tools are categorized into four phases: data preparation, training, testing and result analysis tools.

The data preparation tools are designed to obtain the speech data from data bases, CD ROM or record the speech manually. These tools parametrize the speech data and generate the associated speech labels. For the current work, the task of those tools is performed prior to the HTK, as previously explained, then the result is converted to HTK data format.

HTK allows HMMs to be built with any desired topology using simple text files. The training tools adjusts HMM parameters using the prepared training data, representing text lines, coupled with the data transcription. These tools apply the Baum–Welch re–estimation procedure [2] to maximise the likelihood probabilities of the training data given the model.

HTK provides a recognition tool to decode the sequence of observations and output the associated state sequence. The recognition tool requires a network to describe the transition probabilities from one model to another. The dictionary and language model can be input to the tool to help the recogniser to output the correct state sequence.

The result analysis tool evaluates the performance of the recognition system by matching the recogniser output data with the original reference transcription. This comparison is performed using dynamic programming to align the two transcriptions, the output and the ground truth, and then count the number of: *substitution* (S) and *deletion* (D).

The optimal string match calculates a score for matching the output sample with respect to the reference line. The procedure works such that identical labels match with score 0, a substitution carries a score of 10 and a deletion carries a score of 7. The optimal string match is the label alignment which has the lowest possible score. Once the optimal alignment has been found, the *correction rate* (CR) is then:

$$CR = \frac{N - D - S}{N} \times 100\% \tag{1}$$

where N is the total number of labels in the recogniser output sequence.

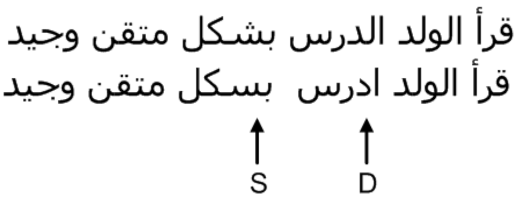


Fig. 5. Optimal string matching. The upper line is the reference line.

Fig 5 illustrates an Arabic reference line and a possible system output. The sentence includes 30 lables. The system output includes one subtitution, one deletion and two insertions. The correction rate equals:

$$CR = \frac{30 - 1 - 1}{30} \times 100 = 93.33\%$$

The example showes how the HTK analysis tool measures the performance of the recognition system.

3 Recognition Results

The performance of the proposed system was assessed using a corpus which includes more than 100 pages of Arabic text in Tahoma font, see Fig 5. Tahoma is a simple font with no overlap or ligature. The data corpus includes 18413 words and 100724 letters, not including spaces.

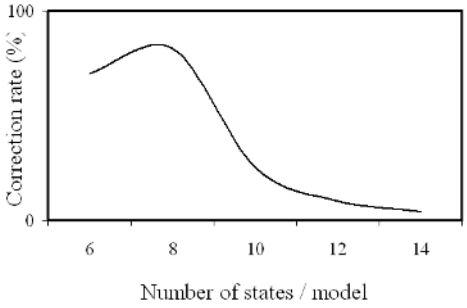


Fig. 6. The impact of the number of states per model on system performance

After noise elimination and deskewing, the 100 pages were decomposed into more than 2500 line images. A set of experiments were performed using 1500 line images for training and 1000 line images for testing. All character models had the same number of states; 8 states. There is no mathematical method to calculate the optimal number of states per model. Alternatively, various values were examined to select the best number of states per model, see Fig 6.

Text line height is proportional to the font size. The line image is measured in pixels. For the corpus under consideration the line image height varies from 35 pixels to 95 pixels depending on the font type and size. To eleminate this dependency, all line images were resized to a single height value; 60 pixels. This

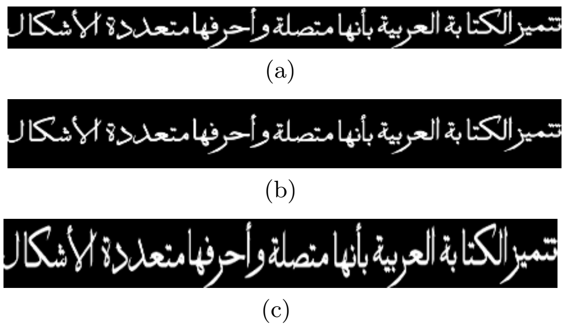

تتميز الكتابة العربية بأنها متصلة وأحرفها متعددة الأشكال
(a)
تتميز الكتابة العربية بأنها متصلة وأحرفها متعددة الأشكال
(b)
تتميز الكتابة العربية بأنها متصلة وأحرفها متعددة الأشكال
(c)

Fig. 7. Text lines with different line heights: (a) 43 pixels, (b) 60 pixels and (c) line image in (a) resized to 60 pixel size

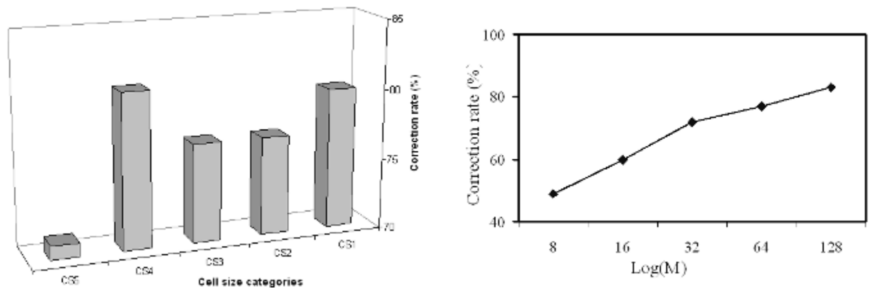
value equals the mean of image heights of all text lines in the data corpus, see Fig 7.

3.1 Cell Size

At any horizontal position, the sliding window is divided into a number of cells, as shown in Fig 2. These cells may or may not overlap. The overlap can be vertical or horizontal and it increases the amount of features generated from a single line and hence increases the processing time.

Table 1. Cell size categories

Category	Cell Size	Horizontal Overlap	Vertical Overlap
CS_1	3×3	-	-
CS_2	3×3	1 pixel	-
CS_3	3×3	2 pixels	-
CS_4	5×5	2 pixels	-
CS_5	5×5	2 pixels	2 pixels



(a) All categories (b) $CS2$

Fig. 8. System performance for the Tahoma font

The first set of experiments studied the cell size parameter. Variuos combinations of cell sizes and overlaps were tested, see Table 1.

Fig 8$-a$ shows the correction rate, CR, of the five categories for the Tahoma font. The codebook here includes 64 clusters. The results illustrates that HTK was more efficiently tuned for Tahoma font rather than for Thuluth font. This is sensible due to decorative curves found at the end of some letters in Thuluth font which overlap with succeeding letters.

3.2 Codebook Size

After extracting features from a line image, the feature vector dimensionality is reduced from 2D to 1D using K-means clustering algorithm [2]. Mapping the 2D feature vector to the nearest cluster is based on the minimum Euclidean distance measure.

This set of expriments studied the codebook size parameter. Five different codebook size values were examined for each font for each category: $8, 16, 32, 64$ and 128. A total number of 50 codebooks were created for this purpose. Fig $8-b$ shows the correction rates of CS_2. It illustrates the improvement in the system performance as the codebook size increases. This conclusion is also applicable to other cell size categories.

3.3 Tri–HMMs

Character models implemented so far are independent; isolated from preceding and succeeding character models. This type is referred to as a mono–model. It has two main advantages: (1) it is easy to train since the total number of models is small; 60 models, (2) the labelling procedure is simple and straightforward. A more efficient type, though more complex, is referred to as a tri–model. Here, each HMM represents a character, its predecessor and its successor. The total number of HMMs jumps to 9393 models. Training and recognition procedures are the same for both types. However, the labelling procedure is more complex with tri-models. The context-dependant tri–HMMs increases the system performance from 84% to 92%.

4 Conclusion

A new system to recognise cursive Arabic text has been presented. The proposed system is based on HMM Toolkit basically designed for speech recognition purpose. Various model parameters have been studied using a corpus that includes data typewritten in a computer–generated font; Tahoma. The system was capable to learn complex ligatures and ovelaps. The system performance has been improved when implementing the tri–model scheme. Future work will concentrate on enlarging the data corpus to include more fonts.

References

1. M. S. Khorsheed, “Off-line Arabic character recognition – A review,” *Pattern Analysis and Applications*, vol. 5, no. 1, pp. 31–45, 2002.
2. L. Rabiner and B. Juang, *Fundamentals Of Speech Recognition.* Prentice Hall, 1993.
3. H. Kim, K. Kim, S. Kim, and J. Lee, “On-line recognition of handwritten chinese characters based on hmms,” *Pattern Recognition*, vol. 30, no. 9, pp. 1489–1500, 1997.
4. W. Kim and R. Park, “Off-line recognition of handwritten korean and alphanumeric characters using hmms,” *Pattern Recognition*, vol. 29, no. 5, pp. 845–858, 1996.
5. M. Pechwitz and V. Maergner, “Hmm based approach for handwritten arabic word recognition using ifn/enit database,” in *The 7th International Conference on Document Analysis and Pattern Recognition*, pp. 890–894, 2003.

6. T. Sari, L. Souici, and M. Sellami, "Offline handwritten arabic character segmentation algorithm: Acsa," in *The 8th International Workshop on Frontiers in Handwriting Recognition*, pp. 452–457, 2002.
7. I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for English and Arabic," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 495–504, 1999.
8. S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book.* Cambridge University Engineering Dept., 2001.
9. M. S. Khorsheed, "Recognising handwritten Arabic manuscripts using a single hidden markov model," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2235–2242, 2003.
10. M. S. Khorsheed, "A lexicon based system with multiple hmms to recognise typewritten and handwritten Arabic words," in *The 17th National Computer Conference*, (Madinah, Saudi Arabia), pp. 613–621, 5-8 April 2004.