Modeling and Training Options for Handwritten Arabic Text Recognition

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Irfan Ahmad

Dortmund

2016

Tag der mündlichen Prüfung. Monday, 20th March, 2017

Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr.-Ing. Gernot A. Fink
Prof. Dr. Laurence Likforman-Sulem (Télécom ParisTech, France)

Abstract

Handwritten text recognition is an active and challenging area of research in the field of pattern recognition. This area of research—although, witnessing great progress in recent years—is far from being a solved problem. Arabic text recognition research—starting relatively later as compared to text recognition research in other scripts like Roman and Chinese—benefited from the techniques developed for other scripts. Most of the techniques were directly adapted from existing works with little or no modifications. Although, this approach had advantages like quick transfer of knowledge into Arabic text recognition domain, it led to an area of research less explored—the study of the peculiarities of Arabic script and utilizing those peculiarities to address the challenges in Arabic text recognition research. The present thesis work is an effort in this direction; whereby, we investigate some of the peculiarities of Arabic script in order to develop methods and techniques which can improve the text recognition performance.

We investigated alternative modeling options for HMM-based Arabic text recognition which exploit the fact that many Arabic characters share similar patterns between them. We present the idea of Arabic sub-characters which are smaller character segments defined utilizing the knowledge of the script. The Arabic characters and their position-dependent shapes can, then, be constructed from these fewer number of sub-characters which results in a compact recognizer with a significantly reduced model set. Special 'white space' and 'connector' models were also presented for Arabic text. Connector models are the special ligatures that join the Arabic characters in a word. Moreover, contextual sub-character modeling was investigated which benefited from the proposed connector model.

Arabic characters have a core shape and a number of dots or other diacritical marks either above or below these core shapes. Many Arabic characters have the same core shapes but differ from each other due to the diacritics. Another modeling option that was presented in this work is based on separating the core shapes from the diacritics thereby reducing the model set considerably. This results in a multi-stage text recognition framework where the core shapes are recognized in the first stage and then the diacritics information is utilized in the second stage to generate the final text hypothesis.

A third modeling approach was investigated which basically combined the above two presented ideas in what we term as sub-core-shape models. Here, the core shapes are separated from the diacritics and, then, the core shapes are further divided into sub core-shapes as many characters share similar patterns within the core shapes. This leads to further reduction in the number of modeling units.

We experimented the presented modeling techniques for handwritten Arabic text recognition on a benchmark handwritten Arabic text database. We achieve state-of-the-art results as compared to the other HMM-based Arabic text recognition systems. All the presented modeling approaches performed better than the commonly used character-shape models under normal training conditions. More importantly, the presented techniques performed significantly better under constrained training environments where limited handwritten training data was used. Some experiments were conducted on a second database which further validated our approaches.

For systems using the familiar character-shape models, we presented class-based contextual modeling. The main idea is to limit the number of unique contextual forms by not modeling every contextual form as a separate model, but, instead, by grouping the characters in the left and the right context into classes where characters in each class have similar effects on its neighboring characters. This leads to a significantly compact text recognizer as compared to the HMM systems that use the standard contextual models.

Our next major investigation was related to recognition of handwritten Arabic text when no handwritten training set is available. Our approach was based on the observation that Arabic is an inherently cursive script; therefore, it has a degree of visual similarity both in the handwritten as well as in the machine printed forms. Accordingly, the challenging step of initializing the system in such tasks was performed by using computer generated machine printed text as the training data. Unsupervised adaptation during recognition was performed to further improve the results. Finally, the recognition hypothesis was iteratively fed back to the recognizer to further improve the text recognizer's performance. This area of research can have favorable implication in the future by alleviating, if not completely removing, the need of preparing manually annotated training data for the recognizers which is a costly and time consuming activity. A number of text recognition experiments were conducted to validate the presented techniques and the results were very impressive keeping in mind that no handwritten training set was used.

Acknowledgements

I would like to thank all those whose help and support has contributed in the completion of the present work. I would like to begin by thanking my supervisor, Professor Dr.-Ing. Gernot A. Fink, for his guidance throughout this work. I feel myself to be fortunate to have an opportunity to work in his research group and under his supervision. His insights and discussions greatly helped my understanding. Moreover, I feel that his professional and detail-oriented attitude had a very positive influence on me professionally. I would also like to thank Professor Lawrence Likforman-Sulem for agreeing to review this work and for providing her feedback.

Many thanks to my colleagues: Akmal, Axel, Leonard, René, and Sebastian. It was really nice to be a part of such a cohesive group. They have always helped me and assisted me in my work and also beyond including the assistance in German language skills. I would also like to thank Claudia for the administrative support offered by her.

A special thanks to Professor Dr. Sabri A. Mahmoud for all the help and support he offered me from the very beginning. He introduced me to this area of research and was instrumental in me pursuing this work. I would like to thank all my friends and colleagues in KFUPM, Saudi Arabia, for their support and good wishes. This work benefited from the funded project sponsored by KFUPM under the project grant RG–1313. Thanks to my friend Yousef Elarian for his support and useful advises.

I am indebted to my parents and my wife for their prayers, love, and support which gave me strength and perseverance during my work and beyond. Thank you for always being on my side. I understand that I was unable to name all the people who are to be thanked but I can assure that I do thank you all from my heart.

Table of Contents

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short-Term Memory
BOF	Bag of Features
CER	Character Error Rate
CTC	Connectionist Temporal Classification
EM	Expectation Maximization
GSC	Gradient-Structural-Concavity
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MDRNN	Multi-Dimensional Recurrent Neural Networks
MLA	Model Length Adaptation
MLE	Maximum Likelihood Estimation
MLLR	Maximum Likelihood Linear Regression
MLP	Multilayer Perceptron
NN	Nearest Neighbor
OCR	Optical Character Recognition
OOV	Out of Vocabulary
PAW	Part of Arabic Word
PHMM	Planar Hidden Markov Model
P-KHATT	Printed-KHATT
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
SVM	Support Vector Machine
WER	Word Error Rate

1 Introduction

Text recognition is an active research area in the field of pattern recognition with the goal of developing automated systems that are able read text as effectively—if not better—as humans. The core idea is to recognize—i.e., provide transcription for—text from images captured by various means like scanner, camera, or video recordings. The term text recognition can include related processes—in addition to the actual text recognition—like text localization, image quality improvement, and post-recognition steps like spelling correction. Text recognition has many useful and interesting applications ranging from digitization and indexing of historical manuscripts, automatic reading of car license-plates, automatic processing of bank checks, automatic sorting of envelopes based on postal address, automatic processing of forms, automatic white-board reading, and related tasks involving word spotting. Furthermore, due to the widespread use of papers in our everyday life and the need for them to be stored electronically for efficient storage and retrieval, there is a need for highly reliable and robust document processing systems. One of the core components of any document processing framework is a text recognition system.

If text recognition is done on machine printed text images, it is commonly referred to as printed text recognition or optical character recognition (OCR). On the other hand, handwritten text recognition deals with recognizing text from images containing text handwritten by humans. Although printed text recognition is much easier as compared to handwritten text recognition, it still has its own challenges like recognition of degraded documents, recognition of text printed using multiple or uncommon font typefaces, and documents having non-uniform text alignments and orientations. Handwritten text recognition, on the other hand, does face most of the above challenges in addition to its own peculiar problems like huge variation in human handwritings both within the same writer and also between different writers, variations due to text slants, skews, and uneven writing lines. Machine printed text recognition can be considered a solved problem for many practical applications like postal address sorting system; whereas, the challenges are far from over when it comes to handwritten text recognition (cf. (Plötz and Fink 2009)). Handwritten text recognition is further classified as offline and online text recognition. In offline text recognition, text is recognized in an 'offline' way after the text writing process has completed. Whereas, online text recognition is performed on data captured in real-time. This is typically done when text is written on touch sensitive screens of tablet computers or smart phones. For online text recognition, information like pen-tip pressure and temporal information on writing is available which is not available for offline text recognition. Consequently, online text recognition is regarded as an easier problem as compared to offline text recognition.

Systems based on Hidden Markov Models (HMMs) are among the most successful and widely used text recognizers. HMMs have sound theoretical and mathematical background. They do not need explicit segmentation of text into recognition units like characters or strokes. Segmentation and recognition is performed at the same time. Efficient algorithms exist for decoding and for estimating the model parameters during training. Moreover, it can easily integrate language models during recognition. The main motivation of using HMMs for text recognition originally came from the field of speech recognition which had the same inherent difficulty of segmentation and recognition.

One of the core steps in setting up a text recognition system is to train the system. To adequately train a text recognition system, it is important that we have enough training samples for each class to be recognized. These classes can represent characters, strokes, or other suitable representations of the text in the underlying script. Benchmark database are developed to assist research in this area so that the systems can be robustly trained and then the systems can be evaluated on the evaluation data. Creating these databases is a time consuming and costly activity. The amount of training data that is needed to adequately train a text recognizer is directly related to the number of classes—the basic recognition units—in a recognition system, i.e., there should be sufficient training samples for each recognition class. Thus, for a given script, if we can have alternative representations for the recognition units, the representation that leads to the minimum number of classes seems to have a clear advantage. For a given amount of training data, the text recognizer will have more samples per class which can lead to more robust training as compared to a recognizer that uses representation which has more classes. On the other hand, in situations where little training data is available, the recognizer which has the most compact model set is expected to perform more robustly. Thus, the modeling choice—defining the basic recognition units—is an important decision. For some scripts like Roman, the characters are the most straightforward and obvious modeling choice. But, for some other scripts like Arabic, the choice is not that obvious.

One of the main objectives of the current thesis is to investigate alternative modeling options for the Arabic script such that it can lead to recognition systems that have compact model sets and can perform more effectively for a given text recognition task. More importantly, the text recognizer shall be able to perform text recognition more robustly when little training data is available. Another important objective of the current thesis, which is an extension of the previous objective, is to investigate handwritten Arabic text recognition when no handwritten training set is available.

In the remaining part of the chapter, we will present an introduction to our thesis work. In Section 1.1, we will present an overview of the text recognition process with focus on HMM-based text recognition systems. Our motivations for the current work is presented in Section 1.2 and the contributions of the work is presented in Section 1.3. Finally, we will present the outline of the thesis in Section 1.4.

1.1. The text recognition process

In this section, we will present a brief overview of the important stages involved in a text recognition system with focus on HMM-based text recognition systems. A typical text recognition process consists of a number of stages including image acquisition—capturing the text into a digital form, preprocessing—localizing the text and improving the quality of the text image for later stages, feature extraction—extracting relevant information from the text images, training—training the classifier such that it learns parameters of the models representing the recognition classes, decoding—generating the recognition hypothesis for the text images, and post-processing—improving the recognition results. Some of the steps might be optional, or even not needed, for a given text recognition task whereas others are mandatory. In Figure 1.1, we illustrate the main stages of a typical text recognition process along with highlighting the focus of the current thesis. Below, we describe the text recognition process in more details:

Preprocessing: Text to be recognized is first captured as images using devices such as scanners and cameras. The text regions are then localized from these images before further processing. Text images may, then, undergo a number of steps before they are ready for the next stage, i.e., feature extraction. If an image has multiple text lines, it needs to be segmented into line images, each containing a single text line. Splitting an image of a paragraph of text into individual text lines might not be a trivial task for handwritten paragraphs. Techniques based on connected component analysis, Hough transformation, ink pixel projections, and smearing are commonly used for this task. For an overview of some common text-line extraction algorithms, readers can refer to (Likforman-Sulem, Zahour, and Taconet 2007). Further segmentation of text line images may be needed depending on the classifier used. HMMs are used with images at the text-line level (where an image consists of a single word or multiple words) and, hence, further segmentation is not needed.

Images may also undergo binarization—separating the ink pixels from the background—in the preprocessing stage depending on what type of features are extracted from them. Another step that is commonly involved in preprocessing of handwritten text line images is the baseline correction. The writing line in human handwriting is normally not straight and thus the text line gets skewed. The skew of the individual text lines needs to be estimated and then corrected by rotating the text line in the direction opposite to the skew angle. Techniques based on projection profiles, principal component analysis, text skeletonization, Hough transformation, and text contours are commonly employed for this task (cf., e.g., (Al-Shatnawi and Omar 2009; Abed and Märgner 2007)). Moreover, the handwritten text may not be completely upright on the vertical axis and this variation needs to be minimized correcting the stroke angle with respect to the vertical axis by applying a shear transformation. This is commonly referred to as slant correction. Common techniques for slant correction are based on chain codes and gradient information from the text images (cf., e.g., (Plötz and Fink 2009; Parvez and Mahmoud 2013b))

Last but not the least, the text size may be normalized to minimize the variability across the line images. For handwritten text images, text size normalization is not a straightforward task. Normalizing the height of the core text across the text line images is a common approach (e.g., (Wienecke, Fink, and Sagerer 2005)).

It is important to note that the order of presentation of the preprocessing steps is not necessarily the same order how they are applied in a particular text recognition system. Additionally, not all of the mentioned steps are required in every text recognition task.

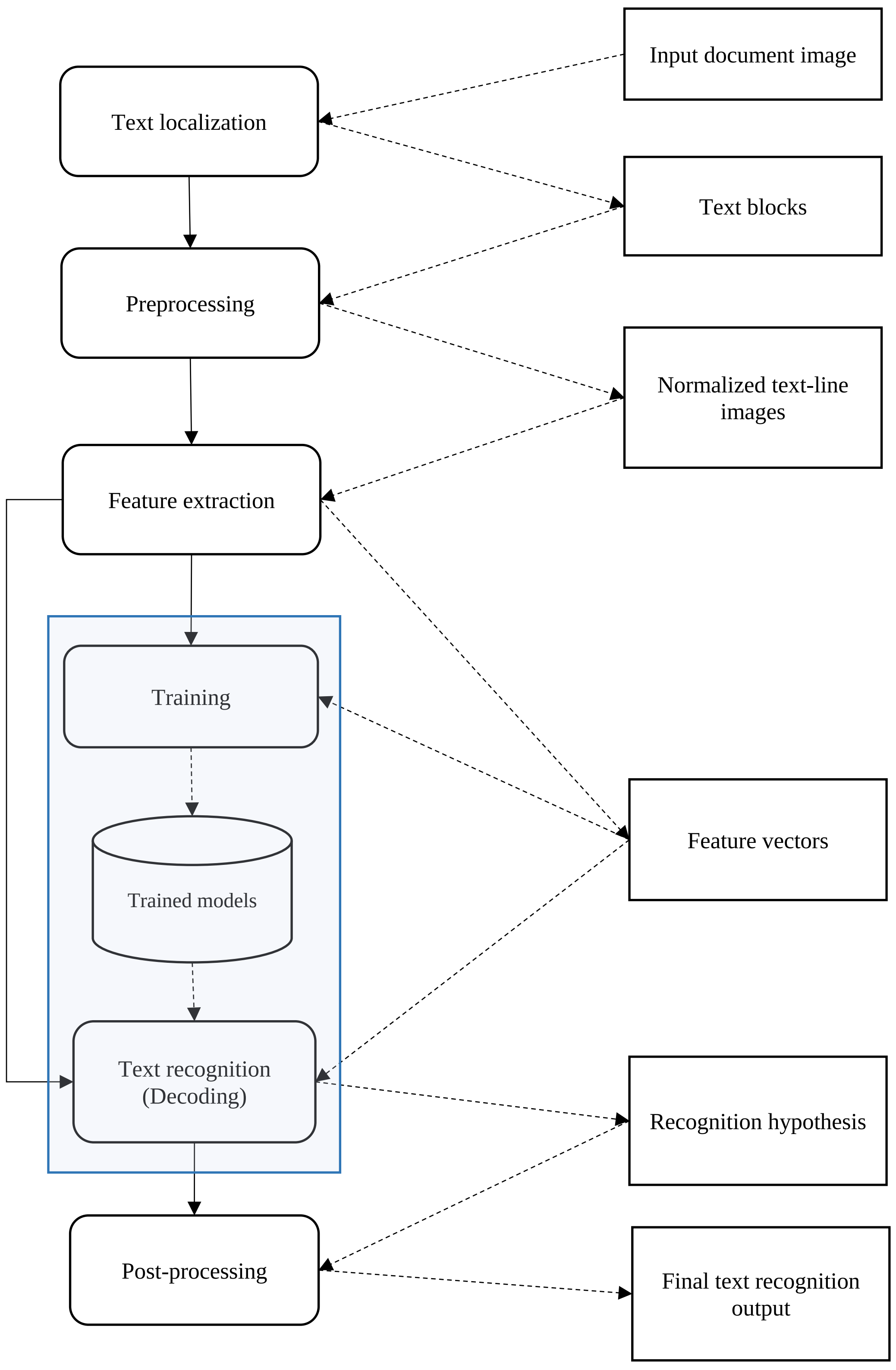


Figure 1.1: Process framework for a typical text recognition system. The shaded area shows the main focus and the scope of the present thesis.

Feature extraction: Feature extraction is an important stage after preprocessing. Adequate features are extracted from the text images which are then used to train a classifier for text recognition. Selecting effective features is important for text recognition. The aim of a feature is to capture the important information about the text strokes such that the different recognition-classes are well separated and the difference in samples within a class is minimized. Another important goal is to keep the dimension of features as low as possible so as to be efficient in terms of space and computation-time requirements.

When using HMMs as classifier for text recognition, the two-dimensional image data need to be sequenced and ordered temporally. The sliding-window technique is the most popular approach to sequence the text image (cf. (Plötz and Fink 2009)). Here a window, having a fixed width and the height as the height of the image, is slid across the image from one end of the image to the other end (in the writing direction). Features are extracted from the image strip under a sliding window.

Training: Training is a core stage of setting up any text recognition system. Sufficient number of text images along with their transcription are needed to train the underlying classifier for each 'class' to be recognized. A class is a representative unit of text to be recognized such as characters or strokes. For the Arabic script, the choice of modelling units is an important decision to be made during the design and setup of a text recognition system. As Arabic characters can have a number of different visual appearances due to their position in the words, a common approach is to model each position-dependent character shape as a class. It is important that each class has sufficient number of samples for adequate training. Benchmark databases are developed to provide sufficient amounts of data for training, calibrating, and evaluating a recognizer's performance under a given environment (e.g., (Marti and Bunke 2003; Mahmoud et al. 2014; Pechwitz et al. 2002; Mahmoud et al. 2011)). The training labels along with the features extracted from the text images are provided to the classifiers during training.

Decoding: Once the training has been done and the recognizer's parameters have been adequately calibrated, the recognizer is ready to be used for decoding. Decoding basically involves generating the recognition hypothesis for the text images. All the preprocessing steps which were performed on the text images used for training the recognizer are also applied for the text images during decoding. Once the input image is preprocessed, features are extracted from them and are fed to the classifier which, in turn, generates the recognition hypothesis.

Viterbi decoding is the most commonly used algorithm for hypothesis generation in the case of the HMM classifier. An important benefit of using HMMs as classifier is that, it is quite straightforward to integrate the statistical n-grams—mainly the bi-grams—as language models during decoding (cf., (Plötz and Fink 2009)). The writing model (using HMMs) along with the n-grams as language models form a robust and effective overall model of handwriting. The n-grams are generally estimated from an external large-text corpora in addition to the training-set transcriptions.

Post-processing: Post-processing steps are optionally used after decoding to improve the recognition results. Rescoring the multiple hypothesis generated during decoding is a commonly applied post-processing step. In the case of HMMs, an initial word network is generated during decoding using shorter n-grams as language models, and the long span n-grams are then used to re-score the initial network. This is done because integrating long span n-grams with decoding can be computationally too expensive to be used during recognition. Another possible post processing step is the classifier combination step. If multiple classifiers were trained and later used to provide separate hypothesis for a given test image, the hypothesis from the different classifiers are finally merged to produce a final recognition hypothesis. Spelling correction is another possible post-processing step that has a potential to improve the final hypothesis especially in natural and unconstrained handwritten text recognition tasks.

1.2. Motivation for the current work

Here we will highlight some of the major issues with text recognition in general and with Arabic text recognition in particular (in the context of the current work) to lay the ground for the motivation of our work.

As mentioned before, availability of sufficient number of training samples for each class is important for adequate training. Moreover, the character frequencies in texts are not uniform, i.e., there are some characters that occur more frequently as compared to other characters. For example, characters '*e*', '*t*', and '*a*' are the most frequent characters whereas characters '*z*', '*q*', and '*x*' are the least frequent characters in the English language (cf., e.g., ("Letter Frequency" 2016)). It is important that the training corpus is large enough to contain enough samples for all the classes. Researchers in this area generally believe that the quantity and quality of training data is as important as developing effective features and classifiers (cf., e.g., (Baird 2007; Varga and Bunke 2008)). The general trend so far has been to train a classifier using large amounts of annotated training data so as to have 'enough' samples for even the low-frequency characters. The task of collecting and manually transcribing the text images to be used as training sets is very costly and time consuming. Moreover, the data collected under a particular environment and setup is, normally, not very useful in text recognition tasks under different environments and setups.

Arabic is one of the Semitic languages. It is spoken by over 400 million people worldwide and is the official language of 22 countries of the world (UNESCO 2015). There are some other languages like Persian and Urdu which use Arabic script for writing; although, they have few more characters than Arabic to cover phones not present in the Arabic language. Research in Arabic text recognition started later than research on scripts like Roman and Chinese. Nevertheless, a lot of effort was devoted in the last two decades on Arabic text recognition research as can be understood based on the number of publications in high quality journals and conferences in addition to a number of text recognition competitions (e.g., (Märgner and Abed 2010; Märgner and Abed 2011; Slimane et al. 2013; Mozaffari and Soltanizadeh 2009; Slimane et al. 2011)). Researchers adapted, with minimal changes, the text recognizers developed for other scripts like Roman to work for the Arabic script (e.g., (Schambach, Rottland, and Alary 2008)). Although this approach has its advantages like script independence, it leaves an important area less explored, i.e., investigating the peculiarities of the Arabic script and using them to develop techniques that can improve a recognizer's performance. Selecting the basic modeling units for the Arabic script is among one such possible areas of investigation.

Arabic characters can take different visual shapes based on their position in a word. While most characters can take four position-dependent visual shapes, some of the characters take only two different shapes. Thus, the most widely used and common approach is to model each character shape (instead of character) as a separate class. It has become the standard modeling choice for Arabic text recognition. In general, this approach works well but leads to some issues. Modeling each character shape as a class leads to almost a four-fold increase in the number of modeling units. This huge model set—typically more than 100 HMMs in an HMM-based text recognition system —in turn, requires even larger amounts of training data so that each class has sufficient number of training samples. Additionally, some character shapes are very infrequent and, as such, having sufficient number of samples for each character shape adds to the difficulty of collection and labelling (by labelling we mean transcribing) the text images for training the recognition system. Some character shapes practically end up having very few samples while other character shapes, at the same time, may have hundreds or thousands of samples. Accordingly, we were interested in investigating modelling and training options for Arabic text recognition which can work robustly under standard training scenarios and can perform reasonably well when very little or no handwritten training data is available. The cursive nature of Arabic script, both in machine printed and handwritten form, and the observation that many characters and character shapes in Arabic share common patterns between them were the main motivating factors that lead us to investigate these properties for better modelling and training options for Arabic text recognizers.

1.3. Contributions of the thesis

The present thesis is related to offline handwritten Arabic text recognition using HMMs. Following are the summaries of the main contributions of the thesis:

1. Arabic sub-characters are presented as an alternative modeling option. Using sub-character HMMs leads to a compact recognizer with a significantly reduced model set as compared to when using the standard system—using character-shapes as models. We also present special white-space and connector models. Moreover, contextual sub-character modeling was investigated which further improved the text recognition performance. The sub-character HMM system was evaluated on a benchmark handwritten Arabic text database, in addition to a second database, and the performance was compared with the standard system. The results were significantly better when using the sub-character HMM system and we report state-of-the-art results on the benchmark database. Moreover, recognition results are much better as compared to the standard system when using limited training data. The work on sub-character HMMs resulted in the following publications:

   - Ahmad, Irfan, Leonard Rothacker, Gernot A. Fink, and Sabri A. Mahmoud. 2013. Novel Sub-Character HMM Models for Arabic Text Recognition. In Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013), 658–62. IEEE. doi:10.1109/ICDAR.2013.135.
   - Ahmad, Irfan, Gernot A. Fink, and Sabri A. Mahmoud. 2014. Improvements in Sub-Character HMM Model Based Arabic Text Recognition. In Proceedings of the 14th International Conference on

Frontiers in Handwriting Recognition (ICFHR 2014), 537–42. Crete: IEEE. doi:10.1109/ICFHR.2014.96.

2. A second modeling option was presented which was based on separating the core shapes from the diacritics. This resulted in a multi-stage text recognition framework where the core shapes are recognized in the first stage and, then, the diacritics information is utilized in the second stage to generate the final text hypothesis. This approach also led to significant reduction in the number of models as compared to the standard system. The multi-stage HMM system was evaluated on a benchmark database under normal training conditions—the complete training set was used to train the system—and with limited training data. Significant improvement in text recognition results is reported under all the training conditions. This work resulted in the following publication:

   - Ahmad, Irfan, and Gernot A. Fink. 2015. Multi-Stage HMM Based Arabic Text Recognition with Rescoring. In Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015), 751–55. IEEE. doi:10.1109/ICDAR.2015.7333862.

3. A third modeling approach was presented which combined the idea of sub-character HMMs and the multi-stage recognition by separating the core shapes from the diacritics. This leads to the highest reduction in the number of modeling units as compared to the standard system as well as the above two modeling approaches. The presented system was evaluated on a benchmark database under normal training conditions and with limited training data. Significant improvement in text recognition results is reported under all the training conditions. Moreover, contextual sub-core-shape modeling was investigated which further improved the text recognition performance.

4. Class-based contextual modeling was presented by grouping the characters in the left and the right context into classes such that characters in each class have similar effects on its neighboring characters. This leads to a significantly compact text recognizer as compared to the HMM systems that use the standard contextual models. This work resulted in the following publication:

   - Ahmad, Irfan, and Gernot A. Fink. 2016. Class-Based Contextual Modeling for Handwritten Arabic Text Recognition. In Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR 2016), Shenzhen, China, 2016.

5. An approach to handwritten Arabic text recognition in the absence of handwritten training data was presented. It resulted in a framework which deals with initializing the text recognizer on machine printed material, improving the recognition results by performing automatic adaptation to unseen data, and iteratively fine-tuning the trained system. Experiments on a benchmark database showed the effectiveness of the framework. The results, although lower than when using the handwritten training data, were very impressive. This work resulted in the following publication:

   - Ahmad, Irfan, and Gernot A. Fink. 2015. Training an Arabic Handwriting Recognizer without a Handwritten Training Data Set. In Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015), 476–80. IEEE. doi:10.1109/ICDAR.2015.7333807.

   As a side work to this above work, we investigated machine printed Arabic text recognition with the aim of recognizing text printed in fonts (i.e., typeface) which was not available during training the recognition system. This work resulted in the following publication:

   - Ahmad, Irfan, Sabri A. Mahmoud, and Gernot A. Fink. 2016. Open-Vocabulary Recognition of Machine-Printed Arabic Text Using Hidden Markov Models. Pattern Recognition 51 (March): 97–111. doi:10.1016/j.patcog.2015.09.011.

We would like to mention that many of the techniques developed as part of this thesis work may work equally well for online text recognition; although, we have not evaluated them on online text recognition tasks. Furthermore, the techniques presented in this thesis may be applicable for some other languages, especially the languages which use Arabic script for writing like Urdu and Persian but we have not evaluated these techniques on those languages yet.

1.4. Outline of the thesis

We start with an introduction to HMMs and lay the basic background and foundations for it in Chapter 2. In Chapter 3, we present an overview of Arabic script and focus on some important aspects of the script that is related to our research. This chapter will serve as a background and covers the domain knowledge necessary for Arabic text recognition research. In Chapter 4, we present the related works on handwritten Arabic text recognition research. We will focus on the literature that is related to the scope of our work with emphasis on HMM-based Arabic text recognition. Our investigations related to the modeling options for Arabic text recognition are presented in Chapter 5 and those related to the training options under constrained training environments are presented in Chapter 6. The experimentations carried out along with the results of the experiments are presented in Chapter 7. Finally, in Chapter 8 we present the conclusions of our work and some possible future works.

2 Fundamentals of Hidden Markov Models

Hidden Markov models (or HMMs) are one of the most popular statistical models for modeling sequential and temporal data. They have simple and sound mathematical and theoretical foundations and have proven to be quite effective in solving real-world problems like automatic speech recognition, text recognition, and labelling biological sequences (cf. (Fink 2014)).

HMMs for speech recognition started gaining popularity in 1970s noticeably after the foundational papers from Rabiner on HMMs and their use for speech recognition (Rabiner and Juang 1986; Rabiner 1989). Since then, HMMs have been extensively researched for speech recognition and many of the state-of-the-art speech recognition systems use HMMs as their underlying technology.

In the context of text recognition, their popularity is also due to the fact that HMMs avoid the need for explicit segmentation of text line images into smaller recognition units like characters or strokes. Segmentation is done implicitly during the recognition of the text lines. Moreover, given a set of observations along with the text transcriptions, HMMs' parameters can be efficiently trained using Expectation Maximization (EM) algorithm.

In the rest of the chapter, we will present the fundamentals of HMMs focusing on aspects related to text recognition. It should be noted that the discussions presented here are informal in nature with the goal of laying the foundations on HMMs. For a more detailed understanding on HMMs, its theory and applications in pattern recognition, interested readers can refer to (Fink 2014; Young et al. 2002). For a detailed discussion on using HMMs for text recognition, readers can refer to (Plötz and Fink 2011).

2.1. Use of HMMs for text recognition

Research on text recognition using HMMs started relatively later in 1990s (cf. (Plötz and Fink 2009)). Although the use of HMMs was reported earlier for isolated digits and character recognition, their real benefit in the context of text recognition arises when using HMMs for word and text line recognition tasks, where most of the other classifiers need an explicit segmentation of text images into characters, strokes, or other representation units.

With regards to text recognition using HMMs, the most common approach was to adapt an HMM-based speech recognition system for text recognition. The one major issue that needed to be addressed was to find an equivalent representation of the time based speech signals being captured as observation. Thus, there was a need to map the information from two-dimensional text images into a one-dimensional observation sequence. This led to the development of sliding window technique where a sliding window passes from one end of the text line to the other and features are computed from the text line slice under the sliding window (Caesar, Gloger, and Mandler 1993). Features from each sliding window are sequenced as a vector.

2.2. Model Definition

HMMs describe a two-stage stochastic process. The first stage describes the state transitions within a finite set of states. In the second stage, an observation is generated at every time period. The observation generated depends only on the current state and not on the previous states or observation histories. HMMs are based on the concept of Markov models which are, essentially, stochastic models which assume Markov property (named after the Russian scientist Andrey Markov) i.e., the next state depends only the information related to the present state and not on the state histories. This assumption works well for many of the real world problems like speech recognition and text recognition, and, solutions to such problem are tractable. In the case of HMMs, the observations are visible but the state sequences are hidden and hence the term 'hidden' in hidden Markov models.

Each HMM representing a recognition unit consists of a number of states including non-emitting start state and final state. Some representations do not explicitly define the non-emitting start and final states. Each emitting state, generating an observation, has a number of mixture densities modelling the state emissions for that particular state. A mixture is modelled as a probability density function (pdf) following a particular distribution. Gaussian pdfs are the most commonly used ones. In this case, the state output probability distribution $b(o_t)$ takes the following form:

$$b_j(o_t) = \sum_{m=1}^{M_j} c_{jm} N(o_t | \mu_{jm}, \Sigma_{jm}) \tag{1}$$

where;

$b_j(o_t)$ is the output probability of observation vector o_t at time t for state j,

M_j is the total number of mixtures in state j,

c_{jm} is the weight of the m^{th} mixture component of state j, and

$N(o| \mu, \Sigma)$ is the multivariate Gaussian with mean vector μ and covariance matrix Σ and is given by:

$$N(o|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)'\Sigma^{-1}(o-\mu)} \tag{2}$$

where;

n is the dimension of the feature vector.

Thus a hidden Markov model M, is defined by:

- A number of states,
- State-transition probabilities including the start probabilities, and
- Output probability distribution for each state which is dependent on mixture densities and mixture weights.

An HMM-based text recognition system consists of a number of HMMs. Each HMM represents a recognition unit. In case of text recognition, the recognition units can be characters, position-dependent character shapes (e.g., for Arabic script), strokes, or other suitable representations. It is important to note that, although these elementary recognition units need to be decided as part of setting up the recognition system, the text images need not be segmented explicitly to train the HMMs representing these recognition units. Only the transcriptions of text at the line level along with the features (computed from the corresponding text line images) are needed to train the HMM models.

2.3. Continuous, semi-continuous, and discrete HMMs

The most straightforward and common setup is for each state to have its own set of mixture densities modelling the state emissions. This setup is termed as continuous HMMs. In a simple scenario, a state may only have a single mixture modelling the emissions. But, using a single-mixture state is normally not sufficient in text recognition problems. As such, the state emissions are commonly modelled using multiple mixture components. Figure 2.1 illustrates a single-mixture per state HMM and a multiple-mixtures per state HMM.

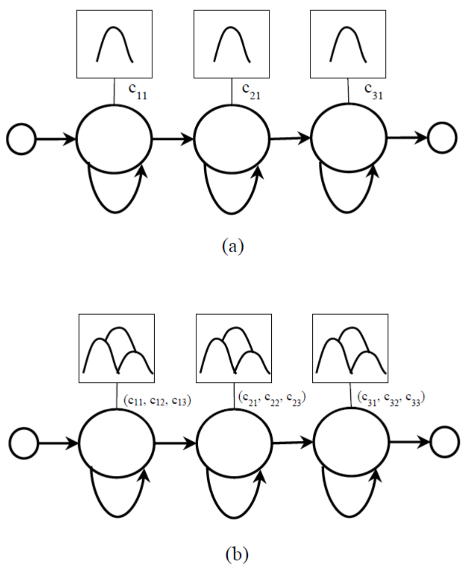

c_{11}
c_{21}
c_{31}
(a)
(c_{11}, c_{12}, c_{13})
(c_{21}, c_{22}, c_{23})
(c_{31}, c_{32}, c_{33})
(b)

Figure 2.1: (a) A single mixture per state HMM and (b) A multiple (three) mixture components per state HMM; c_{jm} denotes the mixture weight for the m^th^ mixture of the j^th^ state.

One practical problem which is sometimes faced when using continuous HMMs is the lack of enough training data. As each state has multiple mixtures and there are many states in an HMM and then there are typically dozens of HMMs in a text recognition system, one ends up having hundreds or even thousands of mixtures that need to be trained using the limited training data. Training each mixture entails estimating its parameters which in the case of Gaussian pdfs are the mean and the variance of the individual mixtures along with the mixture weights. These parameters are in addition to the other parameters of an HMM system such as the state transition probabilities. Thus the need to robustly estimate a huge parameter set for the text recognition system.

One way to mitigate this problem is to involve some level of parameter sharing. Mixture tying is one example of parameter sharing. In mixture tying, mixture components are shared between more than one states. An extreme example is when all the mixtures from all the states of all the HMMs are shared as a global pool. This setup is commonly referred to as semi-continuous HMMs (cf., e.g., (Fink 2014)) or fully tied-mixture system (cf., e.g., (Young et al. 2002)).

There is a third kind of HMMs system referred to as discrete HMMs system. In discrete HMM systems, the pdfs representing the continuous valued observations are replaced by discrete symbols. In addition, the mixture weights in a state are now replaced by probabilities for observing the discrete symbols in that particular state. Discrete HMM systems are natural choice when modelling data which are essentially symbolic like DNA sequences (cf. (Fink 2014; Young et al. 2002)).

Although discrete HMM systems have been used in text recognition tasks (e.g., (Awaida and Khorsheed 2012; Khorsheed 2007; Dehghan et al. 2001)), their use is fairly limited and the benefits of using a discrete system over a continuous or semi-continuous systems are not well established. When using discrete system for text recognition, the feature vectors computed from the image should be symbolic or they need to be converted into a symbolic representation by quantizing them using vector quantization techniques which essentially utilize clustering algorithms. In this thesis, our focus will be on continuous HMM systems. Other types of systems will be discussed explicitly only when needed.

2.4. Model decoding and text recognition

Decoding in the context of HMM-based text recognition refers to transcribing a text image. Given a set of trained HMMs representing the recognition units like characters (it should be noted that word models can be created by simply concatenating the character models) and set of observation vectors representing a test image, the recognition problem is to find the symbol sequence (characters or words) that maximizes the generation probability for the observation sequence computed from the text image. Thus, the recognition problem can be written as:

$$\underset{w}{\operatorname{argmax}} \{P(w|O)\} \tag{3}$$

where;

w is the symbol sequence (characters or words for example), and

O is the observation sequence from the text image.

Using Bayes' rule:

$$P(w|O) = \frac{P(O|w)P(w)}{P(O)} \tag{4}$$

where *P(w)* represents the prior probabilities for the symbol sequence *w*. It is also commonly termed as language model probabilities (represented typically by statistical n-grams). Thus, the most probable symbol sequence given the priors depends on $P(O|w)$.

The likelihood of a model M (a composite model *M* representing a symbol sequence w) generating a given observation sequence is calculated by finding the most likely state sequence (instead of the total probability obtained by summing the probabilities through all the possible state sequences) for the observation sequence. The likelihood of a model M generating the observation O is, thus, given by:

$$P^{*}(\mathrm{O}|\mathrm{M}) = \mathrm{P}(\mathrm{O}, s^{*}|M) = \max_{s} P(O, s|M) \tag{5}$$

$$P^{*}(\mathrm{O}|\mathrm{M}) = \max_{x}\left\{a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)}\right\} \tag{6}$$

where;

$P^{*}(\mathrm{O}|\mathrm{M})$ is the optimal probability of observing observation sequence O = $\{o_1$, o_2, …,$o_T\}$ given the model M. o_t is the observation vector at time interval t,

$a_{x(0)x(1)}$ is the start probability from the entry state $x(0)$ to the first state $x(1)$,

$b_{x(t)}(o_t)$ is the probability of generating observation vector o_t by state x at time t, and

$a_{x(t)x(t+1)}$ is the state transition probability from state $x(t)\ to\ x(t+1)$

An efficient recursion based algorithm—Viterbi algorithm—exists that utilizes the Markov property to compute Equation 6. For more details on this, readers can refer to (Fink 2014).

2.5. Model training

Training an HMM—estimating its parameters optimally—is commonly performed using *Baum-Welch* algorithm (cf. (Young et al. 2002; Fink 2014)). Baum-Welch is an *expectation maximization* (EM) algorithm. It uses the total output probability as the optimization criteria instead of the probability over the optimal path as used in Viterbi algorithm.

During training, we essentially have the transcription of the text images and the corresponding features extracted from the text image. By employing the Baum-Welch training algorithm, means, covariances, mixture weights, and the state transition probabilities are updated after each iteration of the algorithm. A number of iterations of the algorithm is performed to robustly train the parameters. After each iteration, the generation probability of the training data from the model improves over the previous iteration i.e.:

$$P(O|\widehat{M}) \geq P(O|M) \tag{7}$$

where;

$\widehat{M}$ is the updated HMM after an iteration of Baum-Welch training.

The total number of iterations of the Baum-Welch training algorithm are either fixed beforehand or the training is terminated when the improvement converges i.e.:

$$P(O|\widehat{M}) - P(O|M) \leq \varepsilon \tag{8}$$

The threshold ε is set manually. Too few iterations may lead to inadequate training and on the other hand, too many iterations risks the possibility of over-fitting the training data. In such cases, the performance of the recognizer may not be good enough on the test data. For the actual algorithms and the accompanying equations for the Viterbi algorithm and the Baum-Welch training, interested readers can refer to (Fink 2014).

2.6. Deciding the model architecture and initializing the HMMs

When initializing the system, one needs to first decide the architecture of the system including the HMM topology, the number of states, and the number of mixtures per state. Two states are connected if the transition probability between them is non-zero. There are many connection topologies possible but the two most commonly used ones for text recognition are the *linear* and the *Bakis* topologies. Figure 2.2 illustrates the two common topologies used for text recognition. In the linear topology, the transitions moves from start state to the end state where each state on the path can transit only to the next state or to itself. In Bakis topology, a state on the path can transit to itself, or to the next state, or to the state after the next state thereby skipping the next state.

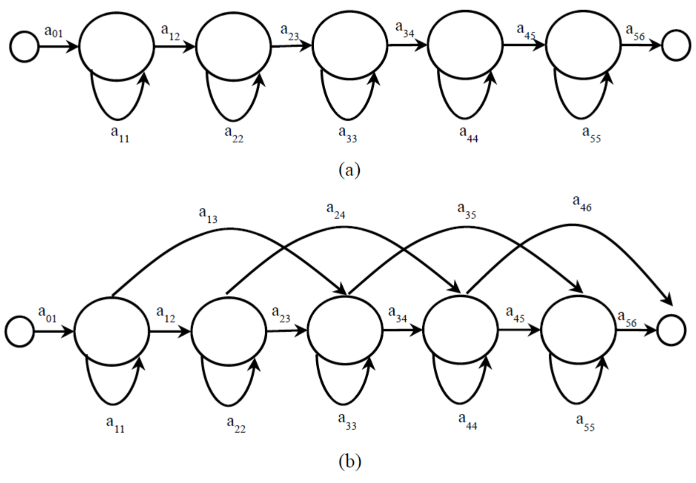


Figure 2.2: Two common HMM topologies used in text recognition; (a) A left-to-right linear topology and (b) A left-to-Right Bakis topology. a_{ij} denotes the state-transition probability from state i to state j.

It is important to note here that many aspects and parameters of an HMM like the model topology, the number of states in an HMM, and the number of mixtures in a state need to be set manually and, generally speaking, there are no algorithms to optimally configure those aspects of the recognition system. One practical way of doing this is by trying various setups (utilizing the domain and expert knowledge) and selecting the setup that gives the best performance on a validation set which is kept separate from the training set. Some heuristic-based techniques have been proposed to optimize these parameters (e.g., (Zimmermann and Bunke 2002; Z. Jiang et al. 2015)).

After deciding the system architecture and before the training step, one has to decide the initial values for the parameters of the system. As mentioned before, word and character segmentation information is not needed when training the HMMs. Nevertheless, more robust initialization can be performed if the boundary information is available as compared to initializing the models when no boundary information is available. The latter case is normally termed as *uniform initialization* or *flat start*. More sophisticated techniques employ a two-stage approach where, training based on uniform initialization is used to annotate the boundary information in the first stage by aligning the image with the transcription and annotating the segmentation information for the elementary units (like character). This annotation information is, in-turn, used in the next stage to perform alignment based initialization of individual HMMs.

When using the uniform initialization method, the values for the mean vectors and covariances for all the mixtures of all the states are normally set to the global (i.e., from training feature sets) means and covariances. When doing initialization using the alignment information, Viterbi based model initialization strategies may be used (cf. (Young et al. 2002)). For each model, given the model specific training data (based on boundary information), the most likely state sequence for corresponding training sequence is found using the Viterbi algorithm. Training vectors are associated to the states and then to the mixtures having the highest likelihood of generating that observation vector. Once the training vectors have been assigned to the mixtures within the states, calculating the mean and variances for the state mixtures is relatively straightforward. The information on the number of training vectors associated with a mixture is also used to calculate the mixture weight. In the first iteration, the training utterances are uniformly distributed between the states and a clustering algorithm (like k-means) is used to cluster the training vectors within a state to a desired number of mixtures. State transition probabilities are updated based on the state occupancy information. A number of iterations of the algorithm is performed to robustly initialize the individual models. Either a predefined number of iterations of the algorithm is executed or if the model converges. For formal details on the algorithm, the readers can refer to (Young et al. 2002).

2.7. Other aspects of HMMs related to text recognition

In this section, we will present some other aspects of HMMs like contextual HMMs, HMM adaptation, and multi-stream HMMs focusing mainly on its application on text recognition.

2.7.1. Modeling the contextual variations

In speech recognition, each phone is typically represented by an HMM and the HMM is termed as monophone (or monomodel in general). A phone may have different pronunciations due to its neighboring phones, i.e., its context. Thus, it is important to somehow capture these contextual variations in order to have a robust speech recognition system. HMMs provide a mechanism to model these contextual variations using the concept of *contextual* HMMs. The various contextual forms of a phone which is defined by a phone and its neighboring phones can be modelled as separate HMMs. Normally, only the next and the previous neighboring phones of a given phone are considered to define its contexts. These contextual forms are thus termed as *triphones* (or *trimodels* in general) and the term is mostly used interchangeably with contextual HMMs. Although contextual HMMs have been used successfully in speech recognition and significant improvements in recognition results have been reported (e.g., (Young and Woodland 1994; Kosmala, Rottland, and Rigoll 1997)), its use in text recognition has not been extensively reported and the benefits of using them over the monomodel HMMs have not been clearly established (Fink and Plötz 2007; Prasad et al. 2008).

To setup contextual HMMs, first the monomodels are initialized and trained and all the different trimodel forms are generated using the training transcriptions. The trimodels for a given monomodel are then created by replicating the monomodel. This is followed by a few iterations of training using the contextual HMMs instead of the monomodels. Using contextual modeling exponentially increases the number of HMMs in the recognition system and this can lead to inadequate training for each of the contextual form. This concern is addressed by performing some form of parameter sharing between the contextual models. The most common approach is to perform state tying of the different contextual forms of the corresponding non-contextual HMM.

There are two main approaches for state tying, i.e., the bottom-up data driven approach and the top-down decision tree based approach. In the data driven approach, the corresponding states of the contextual forms are tied if the inter-state distance is within a threshold. Appropriate distance measure is selected and the threshold value for state clustering is normally set empirically. For the decision tree based clustering approach, the corresponding states of all the contextual forms are initially pooled together and are then successively split based on questions (defined by the experts), each splitting the group into two next level nodes until all the questions have been used or the increase in likelihood is below a threshold. All the states in a resulting leaf node is clustered together.

During recognition, contextual HMMs are used instead of the monomodels while building the recognition network.

2.7.2. Adapting the system for robust recognition

No matter how well a recognition system was trained on the available training data, its performance on the test data is always going to be challenged. The challenges could be due to many reasons like unseen writers, different writing styles, and different environments under which the data was collected. Thus, a recognition system needs to adapt to these unseen settings in order to perform robustly. HMMs provide some adaptation techniques to deal with such situations.

HMM adaptation has been successfully employed in speech recognition tasks where a general purpose recognizer is adapted for speaker specific recognition tasks (e.g., (Gales and Woodland 1996; Leggetter and Woodland 1995)). Training a speaker specific recognizer from the scratch may not be feasible as large amounts of speaker specific training data may be needed. Thus, a small amount of speaker specific data is used to adapt the model parameters of a general purpose speech recognizer. If labelled data for the specific speaker is available then supervised adaptation can be performed. However, if no such data is available then unsupervised adaptation can be performed during the recognition step which essentially uses the recognition hypothesis as labelled data to be used for adaptation in later recognition (Gales and Woodland 1996). In the domain of text recognition, HMM adaptation techniques have been employed at various tasks such as for adapting handwritten text recognizer for a new writers (Saleem et al. 2009) and to adapt a printed text recognizer to a specific font (Ait-Mohand, Paquet, and Ragot 2014).

The task of adaptation to fine-tune the trained model parameters θ such that the adapted parameters θ^* maximizes the likelihood of adaptation data O.

$$\theta^* = \arg\max_{\theta} p(\theta|O)$$

Mixture means and variances are the parameters which are generally adapted. One of the most common techniques employed for parameter adaptation is the Maximum Likelihood Linear Regression (MLLR). It estimates the linear transformations for means and variances and adjusts them accordingly to better fit the new data, i.e., the adaptation data. The transformations are linked across multiple Gaussians so as to robustly estimate them in the presence of limited adaptation data. A group of Gaussians that share the same transform is termed as regression class. Readers can refer to (Leggetter and Woodland 1995; Gales and Woodland 1996) for more details on MLLR based HMM adaptation.

2.7.3. Multi-stream HMMs

It is possible in HMMs to treat the input observation vector as comprising of multiple independent data streams. So, instead of modelling the complete vector as one stream, it can be split into two or more streams. Training the feature vectors as multiple streams, will create separate mixture components for each stream. Figure 2.3 illustrates an HMM with the input observation sequence split into two independent streams. Multi-stream HMMs are commonly used to separate features from two different sources (audio and visual) into two separate streams in audio-visual automatic speech recognition (e.g., (Luettin, Potamianos, and Neti 2001; Manabe and Zhang 2004)). In text recognition, multi-stream HMMs have been used to model different features, computed from the text images, as independent streams (e.g., (Kessentini, Paquet, and Ben Hamadou 2010)).

The assumption made is that the data streams are independent and hence the likelihood from mixture components of the individual streams are multiplied to get the overall likelihood for generating an observation from a state. Thus, the state output probability distribution is given by (Young et al. 2002):

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} N\left(o_{st} | \mu_{jsm}, \Sigma_{jsm}\right) \right]^{\gamma_s} \tag{9}$$

where;

S is the total number of streams, and

γ_s is the stream weight.

The other elements in the equation are similar to the corresponding elements in Equation (1) with the addition of the stream aspect.

Different streams can have different weights so as to give different emphasis to each of the streams. A common approach is to try different weights and select the weights which gives the best recognition results on the validation set.

2.8. Summary

To summarize, HMMs are generative classifiers for statistical modeling of, generally, sequential data. Efficient algorithms like Baum-Welch algorithm and Viterbi algorithm are available for robustly training the classifier and for recognition respectively. HMMs are very successful and widely used classifiers for applications like speech recognition and text recognition. Techniques like contextual HMMs, multi-stream HMMs, and HMM adaptations, when used under suitable scenarios, can further enhance the performance of HMM-based text recognizers. As explicit segmentation of text line images into smaller units like characters or strokes is not needed when using HMM-based text recognition systems, its use for Arabic text recognition is quite popular. Details on the Arabic script in the context of text recognition is presented in the next chapter.

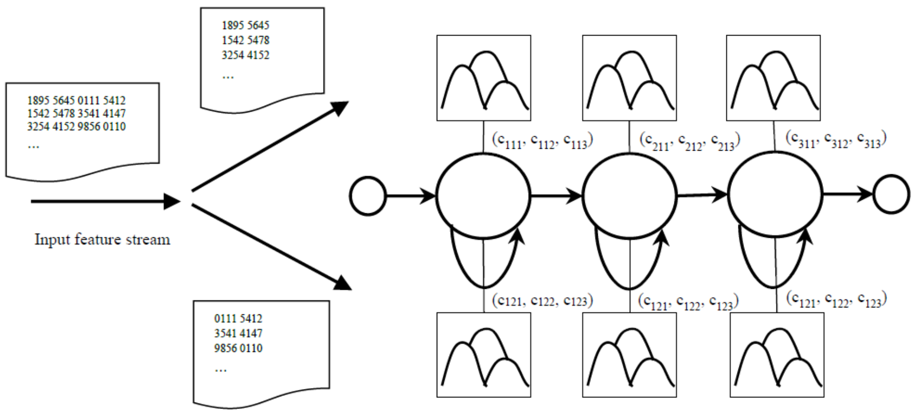


Figure 2.3: An illustration of multi-stream HMMs. The original feature vector split into two streams. c_{jsm} denotes weight for the mth mixture component of sth stream of jth state.

3 Overview of Arabic Script

Arabic is one of the Semitic languages and is the fourth most widely spoken language in the world (cf. (Lewis, Simons, and Fennig (eds.), n.d.; UNESCO 2015)) and the third most widely used writing system in the world. It is spoken by more than 400 million people worldwide including more than 200 million people who speak Arabic as their first language (Lewis, Simons, and Fennig (eds.), n.d.; UNESCO 2015). It is the official language of 22 countries worldwide (UNESCO 2015). Arabic script is also used by many other languages like Urdu, Persian, and Uyghur.

In this chapter we will present an overview of the Arabic script. We will mainly discuss the Arabic writing system without discussing the linguistic aspects of the Arabic language like its grammar and pronunciation as they are not directly related to the topic of the present thesis. For more details on the Arabic language and script, interested readers can refer to (Lewis, Simons, and Fennig (eds.), n.d.).

3.1. Characters and diacritics

Arabic script is cursive both in machine printed and handwritten forms. Arabic alphabets are Abjads—the letters represent the consonants. Arabic is written from right to left and has 28 basic characters. The characters do not have different upper-case and lower-case forms. Figure 3.1 shows the characters in the Arabic script.

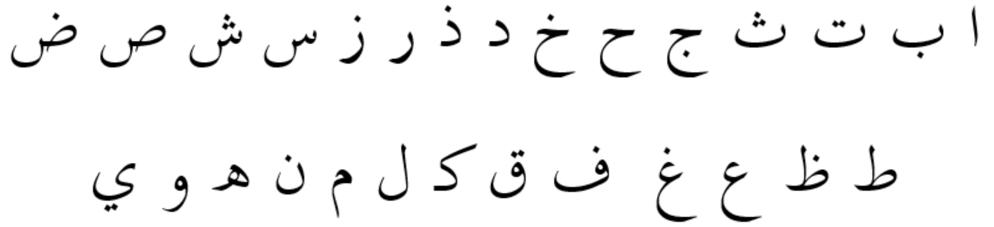


Figure 3.1: Characters in the Arabic script[1].

[1] Roman transliteration for the names of Arabic characters are presented in Figure 3.5.

A character has a core shape (known as *Rasm*) and may have dots (known as *I'jam*) either above (like ن ت) or below (like ي ب) the core shapes. Many of the characters share the same core shape and differ only in the number and position of dots. There are eight characters (خ ذ ز ض ظ غ ف ن) having one dot above, two characters (ت ق) having two dots above, two characters (ث ش) having three dots above, two characters (ب ج) having one dot below, one character (ي) having two dots below, and the remaining 13 characters (ا ح د ر س ص ط ع ک ل م ه و) have no dots either above or below the core shapes. There are no characters that have three dots below them.

The characters are normally connected using a horizontal stroke called *Kashida*. Figure 3.2 shows sample machine printed and handwritten texts in Arabic. It can be observed from the figure that both the handwritten as well as the machine printed texts are cursive and connected.

Apart from the dots, characters can have other diacritics like *Shadda* (◌ّ), *Hamza* (ء), and *Sukun* (◌ْ). Short vowels are also written as diacritics. There are three short vowels (◌ُ◌ِ◌َ) in Arabic. The diacritics, apart from the dots are for the phonetic guidance. In handwritten texts, and to a larger extent even in the machine printed texts, most of the diacritics (apart from the mandatory dots and *Hamza*) are not written but can be deduced by the readers from the context. An exception to this is when writing sacred texts, legal documents, and texts written for the purpose of teaching Arabic. Figure 3.3 shows example Arabic text, with and without the optional diacritics. As an extreme example, some historical manuscripts contain Arabic texts even with the absence of dots. Figure 3.4 shows an example of such a historical manuscript page having Arabic texts (from the Holy Quran) without the dots.

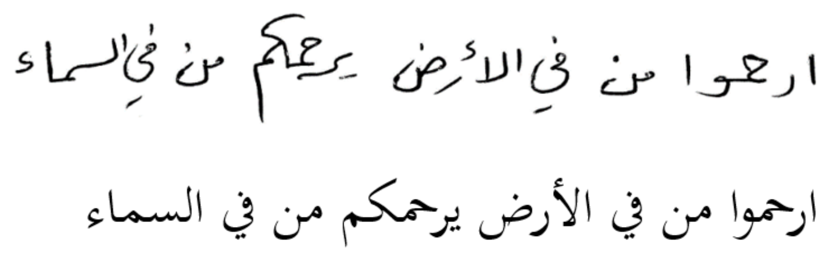

ارحموا من في الأرض يرحمكم من في السماء
ارحموا من في الأرض يرحمكم من في السماء

Figure 3.2: Sample handwritten (above) and machine printed (below) Arabic texts.

أَحِبَّ لِأَخِيكَ مَا تُحِبُّ لِنَفْسِكَ

أحب لأخيك ما تحب لنفسك

Figure 3.3: Sample texts in Arabic, with (above) and without (below) the optional diacritics[2].

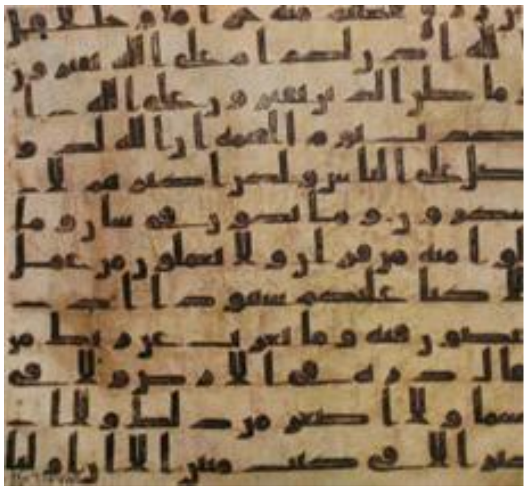


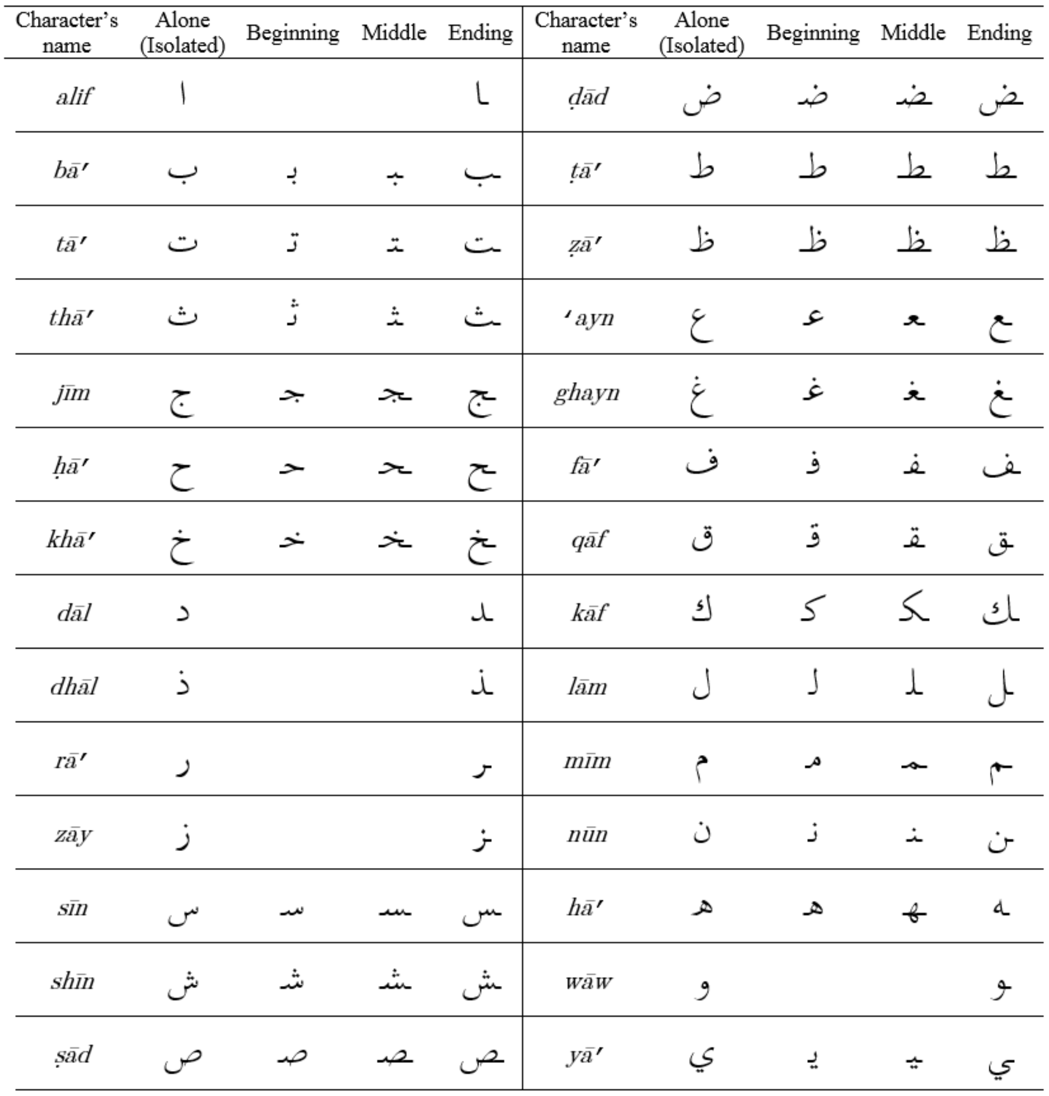
Figure 3.4: Example page from a historical manuscript showing Arabic texts (from the Holy Quran) without the presence of dots and diacritics (Image source: (Abulhab 2009)).

3.2. Position dependent character shapes, words, and PAWs

Characters in the Arabic script can take different visual appearances based on their position in a word. As the Arabic script is cursive, a character in a word is connected to its adjacent characters. 22 of the 28 characters can take up to four different position dependent shapes, i.e., beginning—when the character is connected to a character after it but is not connected to any character before it (like the first character in a word), middle—when the character is connected both before and after it to its adjacent characters, ending—when the character is connected to a character before it but is not connected to a character after it (like the last character in a word), and alone (also termed as isolated)—when there are no characters connected to it. The remaining six characters can only take two of the four position dependent shapes, i.e., ending and alone. Figure 3.5 shows the Arabic characters along with their position dependent shapes.

[2] Transliteration and translation of the Arabic words appearing in the figures in this chapter are presented in Appendix—A.

Those six characters which take only two position dependent shapes basically do not allow the characters after them to connect to them and hence they do not have the beginning and the middle shapes. If they come in the beginning they take the alone shapes and if they come in the middle they take the ending shapes. Figure 3.6 shows some illustrative examples using the Arabic characters *sīn* (س) and *dāl* (د) as examples of characters taking four and two shapes respectively.



Character's name	Alone (Isolated)	Beginning	Middle	Ending	Character's name	Alone (Isolated)	Beginning	Middle	Ending
alif	ا			ـا	*ḍād*	ض	ضـ	ـضـ	ـض
bā'	ب	بـ	ـبـ	ـب	*ṭā'*	ط	طـ	ـطـ	ـط
tā'	ت	تـ	ـتـ	ـت	*ẓā'*	ظ	ظـ	ـظـ	ـظ
thā'	ث	ثـ	ـثـ	ـث	*‘ayn*	ع	عـ	ـعـ	ـع
jīm	ج	جـ	ـجـ	ـج	*ghayn*	غ	غـ	ـغـ	ـغ
ḥā'	ح	حـ	ـحـ	ـح	*fā'*	ف	فـ	ـفـ	ـف
khā'	خ	خـ	ـخـ	ـخ	*qāf*	ق	قـ	ـقـ	ـق
dāl	د			ـد	*kāf*	ك	كـ	ـكـ	ـك
dhāl	ذ			ـذ	*lām*	ل	لـ	ـلـ	ـل
rā'	ر			ـر	*mīm*	م	مـ	ـمـ	ـم
zāy	ز			ـز	*nūn*	ن	نـ	ـنـ	ـن
sīn	س	سـ	ـسـ	ـس	*hā'*	ه	هـ	ـهـ	ـه
shīn	ش	شـ	ـشـ	ـش	*wāw*	و			ـو
ṣād	ص	صـ	ـصـ	ـص	*yā'*	ي	يـ	ـيـ	ـي

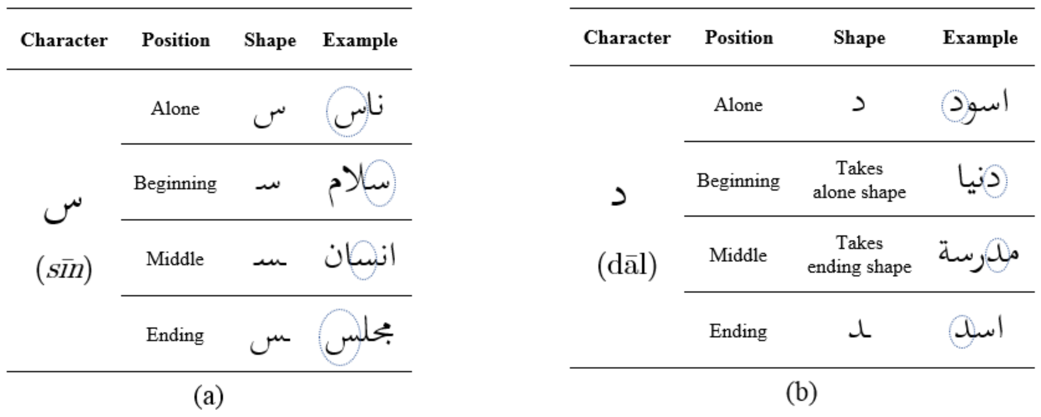
Figure 3.5: Names of Arabic characters[3] and their position dependent shapes. An empty cell indicates that the character does not take that particular position.

[3] There is no single standard for the transliteration of Arabic character names. We have selected a commonly used representation which seems similar to the one published/endorsed by the American Library Association (ALA) and the Library of Congress (LC).

Due to the fact that some of the characters do not allow characters to connect after them, a word in Arabic may be split into multiple components. Each of the split component of a word is called Part of Arabic Word (PAW). Figure 3.7 shows some example words in Arabic having different numbers of PAWs.

3.3. Numerals

Writers in Arabic use two different types of numerals, i.e., the commonly used Arabic numerals also know an Indo-Arabic numerals as well as Eastern-Arabic numerals. It is important to note that the numbers in Arabic are read from left to right. In machine printed text, numbers are commonly written using the Indo-Arabic numerals. People in Middle-East and Gulf region normally prefer using Eastern-Arabic numerals for handwriting whereas people in North-West Africa commonly use the Indo-Arabic numerals. Figure 3.8 shows the ten digits in both the numeral system.



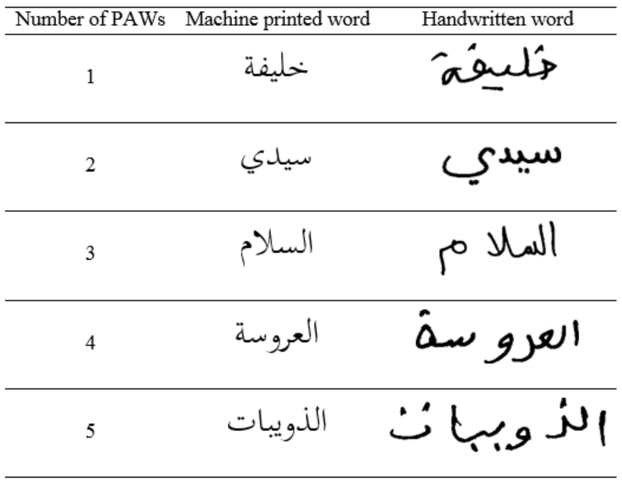
Character	Position	Shape	Example
س (*sīn*)	Alone	س	ناس
	Beginning	سـ	سلام
	Middle	ـسـ	انسان
	Ending	ـس	مجلس

(a)

Character	Position	Shape	Example
د (dāl)	Alone	د	اسود
	Beginning	Takes alone shape	دنيا
	Middle	Takes ending shape	مدرسة
	Ending	ـد	اسد

(b)

Figure 3.6: An illustration, with example words, of characters taking different position dependent shapes; (a) Character *sīn* (س) can take four position dependent shapes, (b) Character *dāl* (د) takes only two position dependent shapes (figure adapted from (Ahmad et al. 2013)).



Number of PAWs	Machine printed word	Handwritten word
1	خليفة	خليفة
2	سيدي	سيدي
3	السلام	السلا م
4	العروسة	العرو سة
5	الذويبات	الذ ويبا ت

Figure 3.7: Sample words in Arabic with different numbers of PAWs. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

1 2 3 4 5 6 7 8 9 0

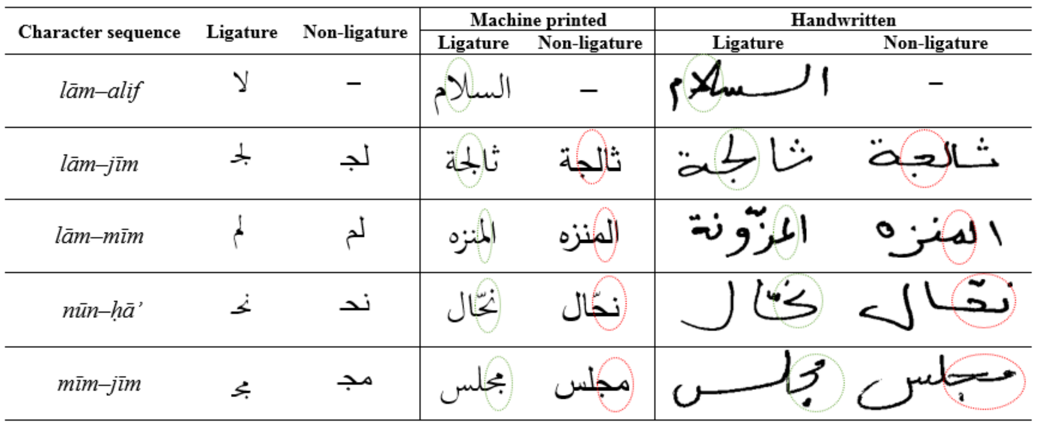
١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ ٠

Figure 3.8: The two numeral systems used in Arabic; the Indo-Arabic system (top) and the Eastern-Arabic system (bottom).

3.4. Ligatures

An important aspect of the Arabic writing system is the presence of special ligatures. Some character sequences can be written in special compact forms instead of simply connecting them using the horizontal *Kashidas*. Ligatures, when written, take visual appearances which are substantially different from mere concatenation of the constituent characters. The character sequence *lām-alif* is a mandatory ligature, i.e., when writing *lām* followed by *alif* it is always written in the ligature form (لا) instead of the non-ligature form (say لـا). Not all character sequences form ligatures but some character sequences like *lām-ḥā'* (لحـ) and *bā'-jīm* (بجـ) are usually written as ligatures.

It is important to note that, as mentioned, only *lām-alif* is a compulsory ligature. Other ligatures are not compulsory and so some writers may write them in ligature forms whereas other writers may write the same character sequences in the non-ligature forms even in similar contexts. In fact, it is also possible that a writer writes a character sequence as a ligature in one instance whereas he/she writes it in a non-ligature form in other instances. Figure 3.9 illustrates some character sequences, their ligature forms, and their non-ligature forms (except *lām-alif*) both in the machine printed and the handwritten texts. Elarian et al. (Elarian et al. 2015a) presented a good analysis on Arabic ligatures and their importance in text recognition.



Character sequence	Ligature	Non-ligature	Machine printed		Handwritten	
			Ligature	Non-ligature	Ligature	Non-ligature
lām–alif	لا	–	السلام	–	السلام	–
lām–jīm	لجـ	لجـ	ثالجة	ثالجة	ثالجة	ثالجة
lām–mīm	لمـ	لمـ	المنزه	المنزه	المزونة	المنزه
nūn–ḥā'	نحـ	نحـ	نحّال	نحّال	نحال	نحال
mīm–jīm	مجـ	مجـ	مجلس	مجلس	مجلس	مجلس

Figure 3.9: Example character sequences, their ligature and non-ligature forms with examples from machine printed and handwritten texts. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

3.5. Challenges in Arabic handwritten text recognition

Handwritten Arabic text recognition faces a number of challenges and it is an open research problem. Some of the problems faced by handwritten Arabic text recognition are similar to that faced by other scripts, like handwriting variability due to different writers and even for a single writer, problems related to text skews and slants, and problems related to touching and overlapping texts. However, as the Arabic script has its own characteristics, handwritten Arabic text recognition faces some unique issues that need to be addressed appropriately. In the remaining part of this section, we will present the major challenges in handwritten Arabic text recognition related to the characteristics of the Arabic script.

Position dependent shapes: One of the major issues related to Arabic text recognition is the fact that Arabic characters can take different shapes based on their position in a word. Although there are only 28 different characters in the Arabic script, position based variations lead to almost 100 different character shapes. For some characters, the variations between their various position dependent shapes are not very large, whereas for other characters, the intra-character variations in appearances are quite large as shown in Figure 3.10. In fact some of the character shapes of a character may look much different than other character shapes of the same character while, at the same time, may look much similar to the character shapes of some other characters as illustrated in Figure 3.11.

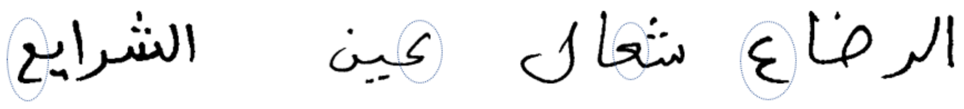


Figure 3.10: Different position dependent character shapes of the character *'ayn* (ع). We can observe that the character shapes are visually quite different from each other. (Handwritten text image source: IFN/ENIT (Pechwitz et al. 2002)).

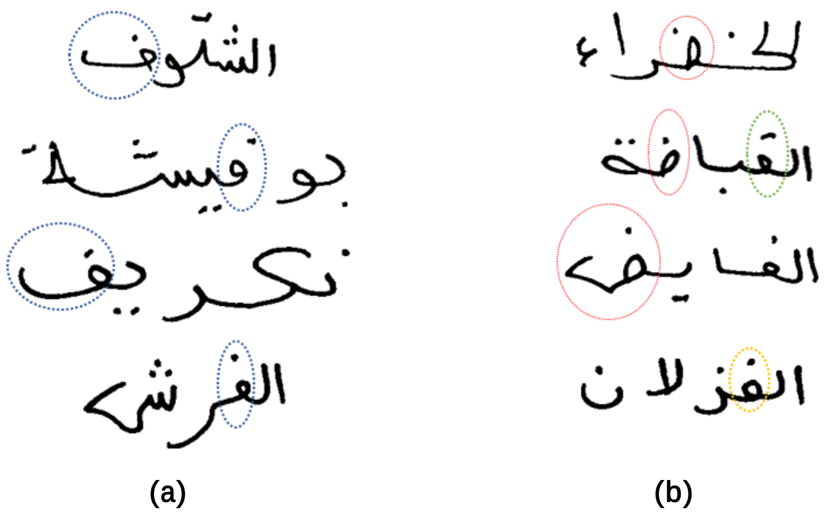


Figure 3.11: Examples illustrating the visual similarities between different characters shapes. (a): The four different character shapes for the character fā' (ف). (b): Encircled character shapes that visually look similar to fā' but are not fā'. Note: The character shapes encircled in a specific color belong to one Arabic character. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

Dots and other diacritics: Another major issue related to handwritten Arabic text recognition is related to the dots and other diacritics present in Arabic texts. As mentioned before, some characters in the Arabic script have dots either above or below them. Moreover, there are up to a maximum of three dots that some of the characters have. Writers do write these dots in a number of different ways. Some writers write these dots clearly, as is the case with the machine printed texts. But sometimes, writers misplace the dots such that the dots do not come directly above or below the character they were meant for. Sometimes writers join two dots together as a single stroke. Some writers also write three dots either as a single stroke or as a dot over a stroke. Another less frequent variation of writing dots is to write them as small circles. Additionally, sometimes the dots can be missing or touching the character's main body. Writers do not write other diacritics but sometimes they may do so, especially the diacritic *Shadda*. In handwritten texts, these diacritics can be easily confused with dots. Moreover, diacritics like *Shadda,* if present over a character, can be viewed as another writing variation for that particular character shape. Figure 3.12 illustrates the problems related to dots and diacritics in handwritten Arabic texts.

Presence of special ligatures: Another prominent issue related to handwritten Arabic text recognition is the presence of ligatures. *lām-alif* ligatures need to be given special attention and in most cases they need to be treated as special characters instead of treating them as two separate characters (i.e., *lām* and *alif*). Additionally, other optional ligatures when present in handwritten texts ideally need special attention, too. But due to the fact that these character sequences are not always written in ligature form (please refer to Figure 3.9), it is difficult to treat these character sequences in a consistent manner. Even if the optional ligatures are treated as special characters, the problem arises that we may potentially end up with a huge number of special characters that need to be handled appropriately for text recognition tasks.

Irregular white-spaces and the usage of two different numerals: Handling white-spaces in handwritten Arabic texts is also not a trivial task. Because of the concept of PAWs in Arabic, white-spaces not only appear between words in Arabic texts but also within words (please refer to Figure 3.7). Last but not the least, due to the fact that both the Indo-Arabic as well as Eastern-Arabic numerals are used in the Arabic writing system, attention needs to be paid on the types of numerals used by the writers when recognizing handwritten Arabic text. If it is not possible to predict the numerals that will be used by writers, or if there are possibilities that both the numerals may be used, then this situation needs to be appropriately addressed in order to recognize the digits robustly.

3.6. Summary

Arabic is a Semitic language and is a widely used language in the world. Some other languages also use the Arabic script as their writing system. Arabic script is cursive both in machine printed and handwritten forms. Arabic characters can have different position dependent shapes. Many characters have dots either above or below the core shapes. There are other diacritics that may be present above or below the characters. Some characters do not connect to other characters in a word leading to some words being segmented into parts which are commonly known as PAWs. The Arabic writing system has special ligatures to represent certain character sequences. Some of the unique aspects of the Arabic script pose unique challenges when recognizing handwritten Arabic texts. How those challenges are addressed by the researchers are presented in the next chapter.

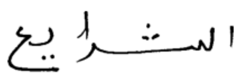
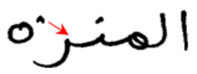
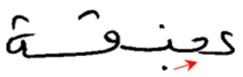
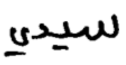
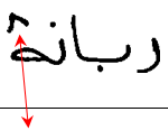
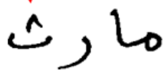
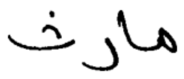
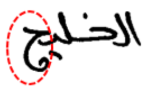
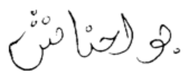
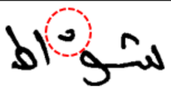
Handwritten image samples	Machine printed texts	Observations
	الشرايع	Dots clearly written and distinguishable.
 	المنزه عجنقة	Dots misplaced as indicated by the red arrow.
 	سيدي ربانة	Two dots written as a stroke. In the second example the stroke for two dots resembles strokes used commonly for three dots as indicated by the red arrow.
 	مارث مارث	Three dots either written as a single stroke (first example) or written as a dot above a stroke (second example).
	الخليج	A dot missing in the red encircled character (ج).
	بولحناش	Dots written as small circles.
	شوّاط	Diacritic shadda (encircled red) looks similar to dots.

Figure 3.12: Example illustrations of problems related to dots and diacritics in the context of handwritten Arabic text recognition. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

4 Related Work

Although research in optical character recognition (OCR) began in 1940s with commercial OCRs appearing in 1950s (cf. (Mantas 1986)), one of the earliest studies on Arabic text recognition was conducted in 1970s (cf. (Al-Badr and Mahmoud 1995)). Earliest work on handwritten Arabic cursive text recognition, where an image contains a word or a sentence in Arabic with connected characters, can be dated back to late 1980's (cf., e.g., (Almuallim and Yamaguchi 1987)). For an overview of the early developments in the field of machine printed and handwritten Arabic text recognition, readers can refer to the work of Al-Badr and Mahmoud (1995).

In this chapter, we will present the literature review on offline handwritten Arabic text recognition research with special focus on HMM-based techniques. We, first, start with a brief overview of the historical developments in handwritten Arabic text recognition. This is followed by a detailed discussion on HMM-based techniques. Other statistical-sequence classifiers particularly those which are based on deep neural networks will be presented towards the end of this chapter. Finally, we will summarize the representative works done in the field of handwritten Arabic text recognition grouped by the nature of the text recognition task. We would like to mention that many of the techniques and approaches presented in the literature may not, necessarily, be novel in the sense that similar techniques might have already been investigated for text recognition of other scripts or in other related research areas like speech recognition.

We will not be discussing the document analysis and preprocessing techniques which are concerned with extracting the Arabic text lines from document images as they are not the focus of our present thesis. Preprocessing techniques related to feature extraction will only be briefly discussed for the sake of completion. Interested readers can refer to (Lorigo and Govindaraju 2006; Parvez and Mahmoud 2013b; Khorsheed 2002) for published surveys on offline handwritten Arabic text recognition.

4.1. A brief overview of the historical developments

Some of the earliest works on handwritten Arabic character and word recognition were in the online domain (cf. (Al-Badr and Mahmoud 1995; Amin, Al-Sadoun, and Fischer 1996)). Online text recognition is, relatively, an easier problem as compared to offline text recognition because the temporal information of the pen strokes is available. Offline handwritten Arabic text recognition started in the late 1980s and early 1990s (cf., e.g., (El-Desouky et al. 1991; Almuallim and Yamaguchi 1987; Goraine, Usher, and Al-Emami 1992)). Most of the earlier systems were based on the syntactic/structural pattern recognition techniques pioneered by Murray Eden (cf. (Mantas 1986)). In the case of word images, they were explicitly segmented and then recognized (e.g., (Almuallim and Yamaguchi 1987; Goraine, Usher, and Al-Emami 1992)). This normally involved thinning of the word images. Al-Badr and Haralick (1998) presented an interesting work on printed Arabic text recognition in 1998 that does not perform explicit segmentation of words in text images but tries to perform recognition and segmentation in an overlapping manner. In this sense, it is somewhat similar to the HMM-based approaches. The system was based on detecting a set of pre-defined shape primitives in a word image (using morphological erosion using the shape primitives as structuring element). A word is defined by the spatial arrangements of the shape primitives. To recognize a word, the system involves a search that tries to maximize the posterior probability of the arrangement of the shape primitives.

One of the earliest works on offline handwritten Arabic text recognition was from Almuallim and Yamaguchi in 1987 (1987). They performed handwritten word recognition on word images written by two different writers. The images were captured using a video camera. The word images were initially preprocessed which involved thinning and then segmented into strokes. An algorithm was presented to segment the text into strokes. The strokes were grouped and features (stroke-group specific) were extracted from these strokes to classify these strokes using rule-based pattern matching. Further, the strokes were combined to form characters, again, using a set of rules. The system parameters were manually adjusted using training samples of 200 words written by two persons. Evaluation was performed on 400 words. Problems related to segmentation were mentioned as the major issues leading to miss-classification.

In 1992, Goraine et al. (Goraine, Usher, and Al-Emami 1992) presented a word recognition task that, too, involved thinning and segmentation of word images into strokes. The strokes were represented by direction codes and were classified into 11 primitives. The strokes were, later, combined into characters using rules. These rules were based on information related to the strokes like their type and position and also using the information from dots and other diacritics. A dictionary was used as a lookup for words to correct the miss-classified characters. Evaluation results were reported on a small dataset of 180 words written by three writers.

Authors in (Farah, Souici-Meslati, and Sellami 2006) presented holistic Arabic word recognition using artificial neural networks (ANNs), k-nearest neighbors (k-NNs), and fuzzy k-NN classifiers. A lexicon size of 48 words from Arabic literal amounts were used. Structural features like number and position of dots, ascenders, descenders, loops, and the number of sub-words were extracted from pre-processed images. Classifier combination was performed by summing the normalized scores for the top three classes from each of the three classifiers. Finally, post-processing based on syntax analysis, using Arabic lexicon for bank literal amounts, was performed to further improve the results.

Parvez and Mahmoud (2013a) presented a structural classifier for handwritten Arabic text recognition. The main idea was to represent the characters with fuzzy polygonal approximation. The fuzzy logic approach (also used in other syntactic approaches to text recognition like (Abuhaiba, Holt, and Datta 1998)) was taken to account for writer variability. The text line images were first segmented into PAWs. Later the PAWs were segmented into characters. During recognition, the input segment was matched (fuzzy polygon matching) with the representative character sample from each character class using nearest neighbor classifiers. For lexicon based word recognition, a lexicon reduction step was employed prior to word matching. The lexicon reducer utilizes the information on the number and positions of dots.

Database and competitions for text recognition: The availability of a ‘good’ database of annotated text images is paramount to the development and improvement of methods and techniques for text recognition which ultimately leads to the improvement of the state-of-the-art. Handwritten Arabic text recognition is no different. Perhaps, the first most influential and effective database for handwritten Arabic text recognition which was made freely available to the research community is the IFN/ENIT database (Pechwitz et al. 2002). The IFN/ENIT database has played an important role in enabling the research community to come up with new ideas and better methods for handwritten text recognition and test them on this database. It has long served as a benchmark database for Arabic text recognition research. Moreover, it has gained special interest from the research community partly due to the fact that regular competitions were held using the database and the results were presented in the top conferences related to the field like ICDAR and ICFHR (e.g., (Märgner, Pechwitz, and Abed 2005; Märgner and Abed 2007; El Abed and Märgner 2010a; Märgner and Abed 2010; Märgner and Abed 2011)). KHATT database of handwritten Arabic text is another freely available handwritten Arabic text database (Mahmoud et al. 2012; Mahmoud et al. 2014). It consists of text paragraphs written by 1000 different writers. The database is divided into disjoint training, development, and test sets and contains text lines with unrestricted writing style. The database is more challenging than the IFN/ENIT database but, as of now, no text recognition competition using the database has been reported. Some other databases like the CENPARMI database of Arabic bank checks (Al-Ohali, Cheriet, and Suen 2003), the AHDB database containing words from Arabic check lexicon (Al-Maadeed, Elliman, and Higgins 2002), and the database presented in (Kharma, Ahmed, and Ward 1999) are limited in terms of size and writers. Moreover, they are either not free (e.g., (Al-Ohali, Cheriet, and Suen 2003)) or not easily accessible. In recent years, competitions held under NIST-OpenHaRT included large vocabulary text line recognition tasks using larger databases developed under DARPA MADCAT program (Tong et al. 2014). These databases were previously not available openly to the research community and the use and distribution of the database and even reporting the comparative results are still restrictive (cf. (Bluche et al. 2014; Tong et al. 2014; NIST 2016)). Maurdor evaluation campaign is another competition held recently (Oparin, Kahn, and Galibert 2014). It is interesting as well as challenging in the sense that the images containing entire text blocks need to be transcribed and not the pre-segmented line images.

4.2. HMM-based Arabic text recognition

In this section, we will present the literature review on HMM-based handwritten Arabic text recognition. An HMM-based text recognition system works on text images at line level, i.e., an image containing a line of text. This line of text may be anything from a single character or digit to multiple running words. Although HMMs can be used for isolated digit and isolated character recognition tasks (such as those presented in (Mahmoud 2008; Awaida and Mahmoud 2009)), other classifiers like support vector machines (SVMs), random forests, and ANNs are more commonly used for such tasks as recognition can be performed holistically without the need of segmentation (cf., e.g., (Mahmoud and Al-Khatib 2010)). Some researchers have manually segmented the text image into recognition units (e.g., (Safabakhsh and Adibi 2005)). The motivation behind this approach was not very clear as explicit segmentation is not needed when using HMMs.

Preprocessing: When performing text recognition using HMMs, the text images are first pre-processed. A number of steps can be carried out during preprocessing which aims to enhance the image before feature extraction and recognition stages. Some of these steps are related to improving the image quality in general like noise removal (e.g., (Benouareth, Ennaji, and Sellami 2008; Pechwitz, Märgner, and Abed 2006; Premkumar Natarajan et al. 2012)). Other steps are related to feature extraction techniques or to reduce variability due to multiple writers and writings like thinning (e.g., (Benouareth, Ennaji, and Sellami 2008; Khorsheed 2003)), baseline correction (e.g., (Pechwitz, Abed, and Märgner 2012; Pechwitz, Märgner, and Abed 2006; Pradeep Natarajan et al. 2011)), skew and slant correction (e.g., (Dehghan et al. 2001; Pechwitz, Märgner, and Abed 2006; Premkumar Natarajan et al. 2012; Kessentini, Paquet, and Ben Hamadou 2010; Chammas, Mokbel, and Likforman-Sulem 2015)), contour smoothing (e.g., (Kessentini, Paquet, and Ben Hamadou 2010)), stroke width normalization (e.g., (Pechwitz, Abed, and Märgner 2012; Pechwitz, Märgner, and Abed 2006; Alma'adeed, Higgins, and Elliman 2002; Dehghan et al. 2001; Azeem and Ahmed 2013)), line height normalization (e.g., (Pechwitz, Abed, and Märgner 2012; Hamdani et al. 2014; Hamdani, Doetsch, and Ney 2014)), character width normalization (e.g., (Pechwitz, Märgner, and Abed 2006; Alma'adeed, Higgins, and Elliman 2002)), inter word and inter PAW space normalization (e.g., (Azeem and Ahmed 2013)), and ruled line removal (e.g., (Saleem et al. 2009; Chammas, Mokbel, and Likforman-Sulem 2015)). Although most of the preprocessing steps are optional (unless a particular feature extraction technique needs that step mandatorily), performing effective preprocessing has shown to improve the recognition results (e.g., (Pechwitz, Abed, and Märgner 2012; Azeem and Ahmed 2013; Stahlberg and Vogel 2015; Chammas, Mokbel, and Likforman-Sulem 2015)). Some features require that the sliding window be of constant height for all the text images. In these situations, the image height is normalized while keeping the aspect ratio, i.e., an image's height to width ratio, constant (e.g., (Dreuw, Jonas, and Ney 2008)).

Sliding Window: Most of the features extraction techniques are based on the sliding window principle where a window, having the same height as the images' height and a few pixels wide, is run from one end of the image to the other along the writing direction of the text. A number of features are extracted from the image slice under the window and are concatenated to form a feature vector. Sliding windows may (e.g., (Azeem and Ahmed 2013; Schambach, Rottland, and Alary 2008; Chammas, Mokbel, and Likforman-Sulem 2015)) or may not (e.g., (Abed and Märgner 2009)) overlap with each other. Additionally, a sliding window maybe be further subdivided into a number of vertical cells and features extracted from each cell of a window are then concatenated to form a feature vector (e.g., (Pechwitz, Abed, and Märgner 2012; Abed and Märgner 2009; Pechwitz, Märgner, and Abed 2006; El-Hajj, Likforman-Sulem, and Mokbel 2005; Kessentini, Paquet, and Ben Hamadou 2010)).

Normally the sliding windows are of fixed width. However, in (Benouareth, Ennaji, and Sellami 2008) the authors presented non-uniform width sliding window approach where the sliding windows' width is computed based on pixel projections of the Arabic text image on the horizontal axis and selecting the midpoint between adjacent maximum/minima pairs as the windows' boundary. They reported improvement in recognition results over the uniform width sliding window approach. Al-Hajj et al. (Al-Hajj Mohamad, Mokbel, and Likforman-Sulem 2007; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009) presented slanted sliding windows for Arabic text recognition. Features were extracted from three different sliding windows. One of the sliding windows is the normal vertical window while the other two windows slant to the left and to the right of the vertical window respectively. Features from each orientation of the sliding window are used to train separate HMM systems thus leading to a total of three HMM systems. The justification for using the slanted sliding windows was to capture the writing inclinations which was cited as the major source of recognition errors. The slant angles for the left slanted and the right slanted sliding windows were decided empirically.

In (Khorsheed 2003), the author used structural features without the use of sliding window. Also a hybrid HMMs/ANNs system with explicit grapheme segmentation was presented in (Menasri et al. 2007) which does not use the sliding window approach.

Features: When it comes to feature extraction, a variety of features have been investigated by the researchers. Appropriate features need to be selected and computed from the text images. The features should aim at minimizing the intra-class variability and at the same time maximizing the inter-class variability. Moreover, the features also should ideally be scale and rotation invariant (as well as to other distortions) as much as possible. Many features reported in the literature for handwritten Arabic text recognition are the same features (normally with little adaptation) that have been used for other scripts as well; like image pixels ((Dreuw, Jonas, and Ney 2008; Hamdani et al. 2014; Abed and Märgner 2009)), pixel densities (e.g., (Abed and Märgner 2009; El-Hajj, Likforman-Sulem, and Mokbel 2005; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009; Azeem and Ahmed 2013; Kessentini, Paquet, and Ben Hamadou 2010)), number of black-white transitions (e.g., (Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009)), gradient features (e.g., (Premkumar Natarajan et al. 2012; Azeem and Ahmed 2013)), concavity features (e.g., (Abed and Märgner 2009; El-Hajj, Likforman-Sulem, and Mokbel 2005; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009; Premkumar Natarajan et al. 2012)), chain-code directions (e.g., (Dehghan et al. 2001; Kessentini, Paquet, and Ben Hamadou 2010)), Fourier descriptors (e.g., (Safabakhsh and Adibi 2005)), Gabor filters (e.g., (Cao et al. 2014)), and percentile features (e.g., (Premkumar Natarajan et al. 2012)). However, some features were specifically designed for the Arabic script (e.g., (Azizi et al. 2010; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009)). In (Azizi et al. 2010), authors presented the use of structural features like the number and position of dots in addition to ascenders and descenders. In (Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009), authors presented features related to the baseline of the Arabic text images. Features like the distance of the center-of-gravity with respect to the baseline, pixel densities both above and below the baseline, and number of black-white transitions above the baseline, and baseline dependent concavity features were extracted. In (Pradeep Natarajan et al. 2011), the authors presented baseline dependent percentile features. Authors in (Hamdani et al. 2009; Abed and Märgner 2009), presented the use of on-line features for offline text recognition. It is based on recovering the temporal information from offline handwritten text images. In (Khorsheed 2003), the author presented the use of line segment lengths and their orientation as features. The text image is first skeletonized and then, using a line approximation algorithm, is broken down into small line segments.

It is also common, and has shown to be effective, to append derivative features to the original feature vector (e.g., (Hamdani et al. 2014; El-Hajj, Likforman-Sulem, and Mokbel 2005; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009;

Premkumar Natarajan et al. 2012)). If the feature vector is too large, feature reduction techniques like PCA (e.g., (Hamdani, Mousa, and Ney 2013; Hamdani et al. 2014)), LDA (e.g., (Premkumar Natarajan et al. 2012; Saleem et al. 2009)), or some other transformations (e.g., (Pechwitz, Abed, and Märgner 2012; Cao et al. 2014)) are employed.

Type of HMM systems: Most of the HMM systems for Arabic text recognition are either continuous HMM systems (e.g., (Hamdani et al. 2014; El-Hajj, Likforman-Sulem, and Mokbel 2005; Safabakhsh and Adibi 2005; Premkumar Natarajan et al. 2012; Azeem and Ahmed 2013)) or semi-continuous HMM systems (e.g., (Benouareth, Ennaji, and Sellami 2008; Pechwitz, Abed, and Märgner 2012)). Discrete HMM systems have also been reported in literature (e.g., (Benouareth, Ennaji, and Sellami 2006; Hamdani et al. 2009; Alma'adeed, Higgins, and Elliman 2002; Dehghan et al. 2001; Khorsheed 2003)) but no clear reasons were presented for preferring it over the commonly used continuous or semi-continuous HMMs. When using discrete HMMs, the features need to be converted into symbolic representation which is done by employing vector quantization techniques, the core of which is a clustering algorithm.

Rothacker et al. (Rothacker, Vajda, and Fink 2012) presented Bag-of-Features (BoF) HMMs for Arabic text recognition. The authors integrated the Bag-of-Features representation, which is popular in computer vision domain, with HMMs. The features from the training set are clustered in an unsupervised way to construct the visual vocabulary (the codebook). The HMMs are trained to estimate the probabilities of observing the visual words from a sliding window in a given state. Multiple visual words can be observed in a given state which makes it different from discrete HMMs in that respect. Authors in (Khoury et al. 2013; Giménez et al. 2014), presented Bernoulli HMMs for handwritten Arabic text recognition. The binary pixels from text images are input directly to the HMMs whose state emissions are modelled using Bernoulli distributions instead of the commonly used Gaussian distributions. To cope with image distortions in the vertical and horizontal axis, window repositioning is performed such that the image slice's center-of-mass is at the center of the sliding window. In (Miled and Amara 2001), authors presented Planar HMM (PHMM) for handwritten Arabic text recognition where each state's emission is, again, an HMM. They argued that planar HMM systems can cope with variability and distortions more robustly as compared to the normal 1D-HMM systems. A word was modeled as a planar HMM having five states where the states (each again being an HMM) represent the upper diacritics, ascenders, core area, descenders, and the lower diacritics respectively.

Modeling: As Arabic characters have different position dependent shapes, the most common approach is to model each character shape as a separate HMM (e.g., (Benouareth, Ennaji, and Sellami 2008; Pechwitz, Abed, and Märgner 2012; Hamdani et al. 2009; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009; Azeem and Ahmed 2013)). This approach is better than modeling each character as an HMM (e.g., (Khorsheed 2003)) as the character shape variability due to its position forms is difficult to model with a single HMM (cf., e.g., (Schambach, Rottland, and Alary 2008)). Additionally, it is also common to model the two-character special ligatures as separate HMMs instead of modeling their constituent characters separately. In (Schambach, Rottland, and Alary 2008), the authors presented parallel path HMMs where each character is modelled as an HMM having multiple paths and each path models a character shape of the character represented by the HMM. In (Menasri et al. 2007), the authors presented what they termed as 'letter-body alphabet'. The motivation was to reduce the number of classes resulting from using character shape as models. The technique involves removal of dots and other diacritics from the character images and performing explicit grapheme segmentation. The resultant set of unique graphemes forms the modeling units. This design seems to be developed keeping in mind a word recognition task involving words that can uniquely be described even after removal of dots. This setup may not work in situations where words may only differ in dots and diacritics, or in case of lexicon free character recognition[4]. Segmentation errors (under segmentation) were cited as the major source of recognition problems.

Word models are built by concatenating the character HMMs (please note that here character is used in a loose sense and can mean either character, character shape, or other representations as described above). Some researchers have used complete word HMMs (e.g., (Benouareth, Ennaji, and Sellami 2006; Alma'adeed, Higgins, and Elliman 2004)), i.e., using HMMs to model words holistically. This approach has many issues. First of all, this will typically lead to huge model set as each word will be represented by a separate HMM. A lexicon may have hundreds, or thousands, or even more words. This will also mean need for more training data

[4] By character recognition we mean character-based recognition where the text recognizer hypothesizes characters instead of words. The system is still trained and tested on text line images and not on isolated characters. We will use the term 'character recognition' to mean 'character-based recognition' in this thesis.

as each word should have sufficient samples in training set to adequately train the model parameters. It may be suitable only for text recognition tasks involving very small lexicon size (cf., (Dehghan et al. 2001)). Moreover, words are of different length and, thus, selecting the number of states for a word HMM may not be that straightforward. In (Benouareth, Ennaji, and Sellami 2006), the number of states for a word HMM was decided based on the number of characters the word has.

Background space modeling is also useful in Arabic as white spaces do not only occur between words but also within words, i.e., between PAWs. Dreuw et al. (Dreuw, Jonas, and Ney 2008) presented explicit white space modeling for Arabic text recognition. Both between words and within word (i.e., between PAWs) white spaces were modelled using a single state HMM model. In (Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009), authors used explicit space models for between word spaces and after the Arabic character *alif*.

Topology: The most common HMM topology used in Arabic text recognition is the Bakis topology (please refer to Section 2.6) (e.g., (Hamdani et al. 2014; Benouareth, Ennaji, and Sellami 2006; Pechwitz, Abed, and Märgner 2012; El-Hajj, Likforman-Sulem, and Mokbel 2005)) where a state can skip the next state and transit to the state after the next state. Characters in the Arabic script are, generally, much wider and have more possibilities for stroke variability as compared to the characters in the Roman script. Thus, using Bakis topology seems to be a better option as it can cope with variability more robustly. Nevertheless linear topologies have also been used successfully (e.g., (Azeem and Ahmed 2013; Stahlberg and Vogel 2015; Schambach, Rottland, and Alary 2008)). In (Dehghan et al. 2001), the authors presented HMMs where the number of forward jumps from a state was empirically set between 2 and 4. In (Khorsheed 2003), no constraints were imposed on forward jumps from a state. The author used a single HMM model to represent all the character shapes for an Arabic character and thus having no restriction on forward jumps might have, to some extent, helped to cope with the appearance variability due to a single model representing all the character shapes of a character.

Model length adaptation: The number of HMM states can be fixed for all the models (e.g., (Pechwitz and Märgner 2003; Benouareth, Ennaji, and Sellami 2008; El-Hajj, Likforman-Sulem, and Mokbel 2005; Premkumar Natarajan et al. 2012)). Alternatively, each HMM model can have different number of states. Various techniques for model length adaptation (MLA), i.e., deciding the optimal number of states for a given HMM model have been proposed (e.g., (Zimmermann and

Bunke 2002; Dreuw, Jonas, and Ney 2008; Z. Jiang et al. 2012)). As Arabic characters have high variations in width and glyph complexity, adapting the length of HMM models seems to be a good approach as compared to using the same number of states for every HMM. Additionally, if explicit models are used to model the background space, it is normally a single state model. Dreuw et al. (Dreuw, Jonas, and Ney 2008) presented model length adaptation based on average character width information. The average length of character was found using the state occupancy statistics from the training data. The information about the average number of frames per class was used in (Dehghan et al. 2001) to decide the number of states for a particular model. In (Chammas, Mokbel, and Likforman-Sulem 2015), the authors used 5-state models for narrow characters and punctuation marks and 8-state models were used for characters having wider glyphs. In (Hamdani et al. 2011), the authors presented their work on MLA where they manually clustered the Arabic character shapes into four groups depending on their expected average widths. They assigned HMM models for each group with a specific number of states with the models in the group containing the narrowest characters having 5 states and the models belonging to the widest characters having 17 states each. It was not clear how they decided the particular numbers of states. In (Khorsheed 2003), the number of line segments (obtained after line approximation on skeletonized text images) a character has determines the number of states its model contains. Jiang et al. (2012) presented the concept of information entropy of states and its use in MLA by removing those states from a model which have very low entropies. The low entropy states are the ones which either have extremely low self-transition probabilities (termed as *slipping* states) or have very low total incoming probabilities (termed as *blocking* states) or have very low total outgoing probabilities (termed as *absorbing* states). Further, the authors extended the work in (Z. Jiang et al. 2015) to provide more elaborate approach to optimize the number of states and the number of mixtures per state using an iterative algorithm. The problem with the proposed algorithms is that they are based on defining specific rules involving many thresholds which may entail large overheads to optimize them for a given recognition task.

The time duration a state of a given model is active, is mainly decided by the transition probabilities for that particular state. Some researchers argue that the state duration needs to be explicitly modelled with some appropriate distributions instead of using a simple transition probability (Benouareth, Ennaji, and Sellami 2008). In (Benouareth, Ennaji, and Sellami 2008), authors presented explicit state duration modeling. Three different distributions (two continuous distributions and one discrete distribution) were investigated to model the state duration. Gamma distribution gave the best improvement.

Mixtures: In the case of continuous HMM systems, the most common approach is to have multiple mixtures per state. The number of mixtures per state is normally a fixed constant and is decided based on the systems' performance on the development sets (e.g., (Benouareth, Ennaji, and Sellami 2008; Azeem and Ahmed 2013)). Jiang et al. (Z. Jiang et al. 2015) presented an algorithm for the optimization of mixture components per state based on some heuristics and a rule based algorithm involving many thresholds. It works together with HMM model length optimization. The authors reported improvement in recognition result by using the optimization technique for the model length and the number of mixtures per state. In (Hamdani et al. 2014; Hamdani, Doetsch, and Ney 2014), mixtures were shared between every two consecutive states. The justification for sharing the mixtures between two states was to make sure that each Gaussian is visited at least once even if a state is skipped because of the Bakis topology. In (Premkumar Natarajan et al. 2012), the authors presented two mixture tying approaches for Arabic text recognition. In one approach all the mixtures for a given model are tied, i.e., all the states of a model share the same mixture pool. In the second approach, all the character shape models for Arabic representing the same character have their mixtures tied together for the respective states and is referred in the paper as *state tied mixtures* (STM). STM setup gave slightly better results in comparison to other setups.

Training: Baum Welch algorithm is the most common algorithm used to train the HMM models (e.g., (Dehghan et al. 2001; Alma'adeed, Higgins, and Elliman 2002; Khorsheed 2003; Premkumar Natarajan et al. 2012; Azeem and Ahmed 2013; Chammas, Mokbel, and Likforman-Sulem 2015)). Other researchers preferred Viterbi training instead of the Baum Welch training (e.g., (Dreuw et al. 2009; Benouareth, Ennaji, and Sellami 2006; Benouareth, Ennaji, and Sellami 2008; Pechwitz, Abed, and Märgner 2012; Schambach, Rottland, and Alary 2008)). Discriminative training based on modified form of Maximum Mutual Information (MMI) was employed in (Dreuw, Heigold, and Ney 2009). The 'lack of enough training data' is a common problem faced in statistical pattern recognition problems and handwritten text recognition is no exception. Adding synthetically generated training data in addition to the original training data is one of the most common approaches in situations where the training data is deemed not big enough.

Dreuw et al. (Dreuw, Jonas, and Ney 2008) added additional training data by simply shifting the original training images by few pixels on the y-axis. This approach, although simple, showed to improve the recognition results. Parameter smoothing (see (Young et al. 2002, page–160) for details on parameter smoothing) after training was performed in (Dehghan et al. 2001) to overcome the problem of non-robust training due to insufficient training data.

Contextual HMMs: As mentioned in Section 2.7.1, contextual HMMs are used to model the variations in a character's appearance due to its neighboring characters. Contextual HMMs with decision tree clustering were presented in (Bianne-Bernard et al. 2011; Hamdani, Mousa, and Ney 2013; Hamdani et al. 2014; Stahlberg and Vogel 2015; Morillot et al. 2013). In (Hamdani, Doetsch, and Ney 2014), the authors presented decision tree based clustering for contextual HMMs where the clustering questions were based only on a character's core shape. In (El-Hajj, Mokbel, and Likforman-Sulem 2008), the authors presented contextual modeling using HMMs for Arabic text recognition where a few contextual forms were manually selected to be modelled. These were mainly characters with descenders which potentially lead to overlaps with the neighboring characters. A total of only 44 contextual forms were added to the original model set. It seems that state clustering was not performed which is understandable given that only a few contextual models were added. In (Premkumar Natarajan et al. 2012), the authors used contextual models for Arabic text recognition. A slight improvement was reported over the use of context independent modeling. One possible reason for not so large improvement in recognition performance, as stated in the paper, was that the use of Arabic character shapes as HMMs already captures most of the context and hence additional contextual modeling, with the implication of addition of many more models, may not be very helpful. The use of contextual HMMs with state clustering was reported in (Cao et al. 2014).

Multi-stream HMMs: Use of multi-stream HMMs for Arabic text recognition was presented by Kessentini et al. in (Kessentini, Paquet, and Ben Hamadou 2010). Four different features were extracted from the text line images and multiple 2-stream HMMs were trained using different combinations of these features. Multi-stream HMMs performed better than feature fusion or classifier fusion. However, it was stated that using stream weights did not improve the results over the equal weight streams.

Model Adaptation: To cope with writing variability during recognition, HMM adaptation techniques have been used successfully (see Section 2.7.2). In

(Premkumar Natarajan et al. 2012), the authors presented the use of MLLR based unsupervised writer adaptation where only the means (and not the standard deviations) were updated. In (Dreuw et al. 2009), the authors used the writer information in the training set to estimate writer dependent feature transformations which are then used to transform the features for every set of writers. These transformed features are then used to train the writer dependent models. After doing a first pass decoding of the test set, text dependent writer clustering is performed. An unsupervised association of writer clusters from the test set to the writers in the training set is performed and then the writer dependent models are used for decoding in the second pass. The authors in (Dreuw, Heigold, and Ney 2009) presented confidence based discriminative training where, in the first pass, a recognition hypothesis is generated and confidence scores (at the word level and at the state level) are used to discriminatively train a second system using the hypothesized text in the first pass. The idea of state level confidence scores was not to reject a word entirely if its confidence score is low and, instead, utilize data at state level (for training) for those states whose confidence level is high. In (Hamdani et al. 2014), the authors presented multi-pass decoding where, after the fast pass the writing styles are clustered in an unsupervised way. In the next pass, feature adaptation is done using these clusters, i.e., the features are transformed using MLLR based transformations estimated for the clusters. In (Cao et al. 2014), the authors presented two approaches to writer and writing style adaptation. In the first approach, a writer independent system is adapted separately for each writer in the training set. During recognition, the input document is associated to a writer from the training set with a score from an SVM based writer identification system. If the score is high (above some threshold), the document is decoded using the writer specific system otherwise it is decoded using the writer independent system. In the second approach, the input document is decoded as a first step. The text hypothesis is used to group documents in the training set having similar writing styles. The codebook is then adapted to the set of documents in training set which matches the writing style of the input document. The HMM model state transition probabilities are then adapted using the average character width information from the input document. Finally, the input document is decoded once again using the adapted system.

Multiple Recognition Systems: Another strategy employed to improve the recognition results is to use multiple recognition systems and later combine their results after recognition. If the recognizers have different characteristics like different features used (e.g., (Abed and Märgner 2009)), different training criteria (e.g., (Farah, Souici-Meslati, and Sellami 2006)), or different classifiers (e.g., (Farah, Souici-Meslati, and Sellami 2006)), it is expected that they make errors in different regions, i.e., their errors are independent of each other. Thus combining their results effectively can lead to overall results which are better than the results of the best individual system. The challenge is to design the systems such that they have different properties and the challenge is also to design effective combination schemes. In (Al-Hajj Mohamad, Mokbel, and Likforman-Sulem 2007; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009), the authors trained three different HMM systems, each using features from sliding windows oriented at specific angles. The final recognition output was obtained by combining the recognition results of the individual recognizers. Three different combination schemes were investigated, i.e., the summation of normalized likelihood scores from the individual recognizers, the majority voting scheme, and training (using scores from individual HMM classifiers) an ANN (Multilayer Perceptron) system to output the top choice. In (Azeem and Ahmed 2013), the authors also used three different HMM systems each trained on features from sliding windows tilted in certain angles (similar to (Al-Hajj Mohamad, Mokbel, and Likforman-Sulem 2007; Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009)). Finally the results from the three system were combined using sum, majority vote, and maximum rules. The exact decisions on how and which scores to combine were set as rules and involved some score thresholds. In (Azizi et al. 2010), the authors presented the use of six different measures (which were computed from individual classifier's output) of classifiers dissimilarity. These measures were used to select a sub-set of recognition systems which are the most diverse with the hope that the overall combination of the results would be better than the individual system's performance. Based on experimental results, the authors showed that combining the recognition systems based on diversity is better than combining systems which give the highest individual recognition results. In (El Abed and Märgner 2010b), the authors presented their investigation on combining different recognition systems. Different strategies like majority voting and voting schemes based on ranks of different classes from each system were presented including training an ANN classifier to output the combination result. In (Schambach, Rottland, and Alary 2008), the authors presented classifier combination using weighted sum voting where the scores are weighted based on the recognizers' individual performances. The individual systems differ from each other based on different preprocessing techniques employed before extracting the features from the text images. In (Alma'adeed, Higgins, and Elliman 2004), the authors presented multiple HMM recognition systems, each to recognize a sub-set of words from the Arabic literal amount lexicon. As a first step, an input word is assigned to one of the eight groups using features like number and position of dots, and the number of PAWs. Next, an HMM system specifically trained on word samples from that group is used to recognize the input word. In (Menasri et al. 2007), the authors presented a hybrid HMMs and ANNs system which involves explicit segmentation of text images into graphemes after the removal of dots and diacritics. Features are extracted for individual graphemes. An iterative training is performed where the HMM system annotates the data used to train the neural network system which, in turn, computes the observation probability distribution for the HMM system. In (Pradeep Natarajan et al. 2009), the authors use the segmentation information generated by an HMM system to train an SVM classifier for the Arabic characters. In the next step, the scores from both the classifiers are combined to output the final recognition hypothesis.

Language Modeling: When it comes to recognizing multiple words in a text line image, the use of language model is imperative. The good thing is that the HMMs can seamlessly integrate statistical n-grams (as language model) for decoding. One of the most important concerns when using the language models is to deal with Out of Vocabulary (OOV) words, i.e., words in the test set which were not known before. As one increases the lexicon size the OOV rate is, in general, expected to decrease. But having a very large lexicon comes with its own issues. It can be difficult to estimate the language model robustly as many of the words will occur infrequently in the corpus. Moreover, having a large lexicon will also lead to more recognition ambiguities (cf. (BenZeghiba, Louradour, and Kermorvant 2015)). Thus a balance between the lexicon size and the possibility of missing words due to them being OOVs needs to be maintained. In (Premkumar Natarajan et al. 2012; Pradeep Natarajan et al. 2011), the authors presented the use of word language models for handwritten Arabic text recognition. A lexicon size of 120K words (92K in (Pradeep Natarajan et al. 2011)) is trained using a large corpus containing 217 million words (90 million words with OOV rate of 4.2% in (Pradeep Natarajan et al. 2011)).

As Arabic is a highly inflectional language, the problem of OOV is even higher. A simple word based lexicon is generally not the best choice. Authors in (Hamdani, Mousa, and Ney 2013) presented open vocabulary Arabic text recognition where the Arabic words were morphologically analyzed and decomposed into prefixes, root, and suffixes. For building the lexicon and estimating the language models, words with high frequencies in the training corpus were kept in the lexicon and the infrequent words were decomposed and the prefixes, roots, and suffixes were added to the lexicon. Special markers were used for prefixes and suffixes in order to reconstruct the words from the recognized segments. This approach resulted in limiting the lexicon size and at the same time, many OOV words can be recognized by constructing them by merging different prefixes, roots, and suffixes. BenZeghiba et al. in (BenZeghiba, Louradour, and Kermorvant 2015) presented hybrid word/PAW language model for Arabic text recognition. The frequently occurring Arabic words in the training corpus were included in the lexicon and the remaining words were broken into PAWs. This leads to reduction in the lexicon size and also reduction in the OOV words. Words were later reconstructed by combining these PAWs after recognition.

When performing lexicon free character recognition, character language models can be used. When using character language models, the issue of OOV is not present. High order n-grams (typically trigrams or 4–grams) are typically preferred but integrating a high order n-gram during decoding is computationally very expensive. Bigrams can integrate seamlessly due to the reason that the HMMs are first order Markov chains. Some researchers prefer using bigrams in the forward pass of decoding and trigrams in the backward pass when the search space is limited (e.g., (Premkumar Natarajan et al. 2012; Pradeep Natarajan et al. 2011)).

Post-processing: In (Khorsheed 2003), the author presented a word recognition task using character HMMs without the use of lexicon. Spell checking was employed to correct some of the recognized words which were not available in the dictionary. In (Premkumar Natarajan et al. 2012) and (Cao et al. 2014), the authors reported the use of glyph models without dots to rescore the n-best list generated by the recognizer.

4.3. Other statistical sequential models

After a deep learning trend in computer vision and speech recognition and the work on offline text recognition by Graves and Schmidhuber (e.g., (Graves and Schmidhuber 2009; Graves 2012)), deep learning based recognition systems have received high interest. One of the main drawbacks of using non-HMM classifiers including those based on ANNs was that the text needed to be explicitly segmented for training and recognition. The use of connectionist temporal classification (CTC) in the context of RNNs allows for recognition without the need for prior segmentation (cf. (Graves and Schmidhuber 2009)). This has enabled the use of neural networks based classifier for the task of offline handwritten text recognition and, thus, has gained popularity in recent years (e.g., (Graves and Schmidhuber 2009; Graves 2012; Abandah, Jamour, and Qaralleh 2014; Hamdani et al. 2014; Hamdani, Doetsch, and Ney 2014; Moysset et al. 2014; Bluche et al. 2014)). Moreover, the RNNs architecture involving long short-term memory (LSTM) enables it to capture longer contexts which may be important for offline text recognition tasks. Interested readers can refer to (Graves and Schmidhuber 2009; Graves 2012) for a detailed discussion on RNNs, CTC, and LSTM in the context of offline text recognition.

There are currently three most popular approaches when using neural network-based classifiers that avoid the explicit segmentation of the text images. The first approach is the one presented by Graves and Schmidhuber (2009) and later extended by Graves in (Graves 2012). Raw image pixels are given as input to a multi-dimensional RNN (MDRNN). A hierarchical architecture is designed to convert the two-dimensional image into a one-dimensional sequence which is finally labelled by the output layer. Abandah et al. presented handwritten Arabic text recognition using recurrent neural networks (Abandah, Jamour, and Qaralleh 2014). Their approach is based on explicit segmentation of text line images into graphemes. A rule-based algorithm is presented to segment the Arabic text into PAWs and then segment them into graphemes. Dots and diacritics were separated from text images and later associated with PAWs. A number of features were extracted from the segmented graphemes. RNNs with bidirectional LSTMs (BLSTMs) and CTC layer is used for training and recognition. The BLSTM architecture allows for integrating the contextual information from both the directions across the handwritten text. A dictionary is finally used to correct the output of the recognizer based on the decoding lexicon.

The second approach involving neural networks is to use them, as some researchers term, *in-tandem* with HMMs (Hamdani et al. 2014). A hybrid HMMs-ANNs system was presented in (Hamdani et al. 2014) where the HMM system was used in tandem with the BLSTM RNNs. Forced alignment from a pre trained HMM system is used to label each observation with its character annotation. This forced aligned data is used to train the RNNs. The trained network is used to calculate the posterior distribution for each observation over the character labels. This, in turn, is used to retrain a new HMM system which is finally used for text recognition.

Finally, the third approach is to use neural networks with HMMs in a hybrid way where the state output probability is directly estimated using the neural networks. A hybrid HMM-ANN system was presented in (Hamdani, Doetsch, and Ney 2014) where the HMMs were not retrained again using the features trained on the RNN. Instead, the posteriors estimated by the RNN were directly used as emission probabilities for the HMMs. In (BenZeghiba, Louradour, and Kermorvant 2015), the authors presented a hybrid HMM-ANN system where the ANN is a Multi Directional LSTM Recurrent Neural Networks and is used to estimate the HMMs' state emission probabilities. The neural network was trained using the CTC criterion thereby avoiding the need of explicit segmentation of text images. Due to the large training data requirements for such networks, a seed model was trained using some external text database and in another text recognition task, some artificial training data was added by applying several transformations to the original training images. In (Moysset et al. 2014), the authors presented a multi-lingual text recognition system. It is a hybrid RNN-HMM system where the optical modeling is performed using Recurrent Neural Networks (RNNs) with LSTM cells. Raw pixels were taken as features to train the network. CTC was used to avoid the explicit segmentation of text images into character shapes. A stepwise training was performed where simple and clean text images were used initially before incorporating the complete dataset which also involves challenging and noisy images. For decoding, HMMs were used where each character shape was represented by a single state HMM (with self-loop) whose emission probabilities were estimated using the RNN system trained as described above. The training data size was augmented by adding synthesized data obtained by applying some transformations (like shrinking, slanting, and expanding) on the original training images. In (Bluche et al. 2014), the authors presented a handwritten text recognition system developed for Arabic text recognition. Most of the characteristics of the system are similar to the one presented in (Moysset et al. 2014). Eleven different RNN based systems were trained, each with different random seed and different and disjoint training set in the initial training stage (stepwise training was employed). After the initial training, all the training data was used by each of the eleven systems for further training. Later, the results were combined from these systems using an incremental approach based on their individual accuracies and based on the improvements in recognition results after adding them. In (Stahlberg and Vogel 2015), the authors train an initial HMM system and use the forced alignment information to train a neural network system which models the emission probabilities for the final HMM system.

In (Morillot et al. 2013), the authors presented a comparative study of using two recognition systems for handwritten Arabic text recognition. One of the systems was based on HMMs while the other was based on RNNs involving BLSTMs. They employed same features and sliding window attributes for both the system. The RNN-based system outperformed the HMM-based system based on the evaluation results. Based on conducting text recognition experiments on two separate handwritten text databases, the authors in (Bluche, Ney, and Kermorvant 2014) concluded that, both Deep multilayer perceptron (MLP) and LSTM-RNNs (the popular choice in text recognition domain) are equally suited for training. Moreover, they also had similar conclusions for the use of features, whether raw pixel value or handcrafted.

4.4. Summary of the related works

In this section we will present a summary of published works on handwritten Arabic text recognition tasks. We will divide the published works into three groups. The first group includes representative works on holistic recognition of isolated digits, characters, and PAWs. It also includes some early works on word recognition which either recognizes the words holistically or use syntactic classifiers to perform explicit segmentation based recognition. The second group contains representative works on handwritten text recognition using the IFN/ENIT database (Pechwitz et al. 2002). Text recognition tasks using the IFN/ENIT database is of particular importance due to the popularity of the database, its use in many text recognition competitions, and also due to the fact that the database is available free to the research community. The third group includes representative works on recognizing text images containing multiple words. The main aspects that separate the last group from the second group is the use of language models and dealing with the out of vocabulary (OOV) words.

In Table 4.1, we present the summary of the works related to isolated digits, isolated characters, holistic PAW recognition, and some early works on word recognition. The table includes only a small selection of the published works in this area which we believe are representative of the tasks. The main aspects for effective recognition are related to careful preprocessing, use of effective features and classifiers. Handwritten digits recognition is one of the easiest recognition tasks. It is a ten class (representing digits from 0 to 9) problem and is, to a larger extent, considered solved. Rates of over 99% are reported in the literature for already segmented isolated digits recognition (e.g., (Awaida and Mahmoud 2009)). In case of number recognition (containing multiple digits) involving touching digits, the task still faces some challenges (e.g., (Alamri, He, and Suen 2009)). Some practical uses of digit recognition is in recognizing numeral amounts in bank checks and in zip code reading. Isolated character recognition is the next simplest task. The classes represent the characters of the script. The main challenges related to isolated character recognition lie in distinguishing the characters which share the same core shape and different only with respect to the dots. Isolated character recognition is also a largely solved problem and is of limited practical use. Recognizing the Arabic PAWs and words holistically may make practical sense only when the number of classes are very limited such as the tasks related to literal amount recognition form bank checks which typically contains few dozen words or PAWs.

In Table 4.2, we present the summary of the works related to handwritten word recognition using the IFN/ENIT database. Apart from the challenges related to preprocessing, development of effective features, and the effective use of classifiers; word recognition tasks have to deal with defining appropriate modeling units like characters (or character shapes for Arabic), training them with or without the segmentation of word images, constructing the decoding network from these modeling units, and dealing with alternative pronunciations for a word. Use of HMM-based classifiers are by far the most common approach under this category.

In Table 4.3, we present text recognition tasks dealing with text images containing multiple words. As mentioned earlier, the main aspects which separates it from word recognition tasks are the use of language models and dealing with OOV words. Moreover, if a text image includes multiple lines, separating these lines is another important challenge that needs to be addressed.

With this knowledge of the state-of-the-art in handwritten Arabic text recognition as foundations, we present our contributions related to HMM-based handwritten Arabic text recognition in the next two chapters.

Table 4.1: Summary of the representative works on holistic recognition of digits, characters, PAWs, and some early works on words recognition.

System	Key Features	Characteristics of the database	Recognition results	Remarks
Alamri et al. 2009 (Alamri, He, and Suen 2009)	• SVM system with radial basis function as the kernel • Gradient features • A rule based algorithm to separate touching digits	Digit images from CENPARMI database of Arabic checks: • 24,784 digit images used for training • 6199 digit images used for evaluation • 132 images of pairs of touching digits used for evaluation of the touching digit recognition task	• Misclassification rate of 1.52% for non-touching isolated digits recognition • Misclassification rate of 7.78% for touching digits recognition	Isolated and touching digits recognition
Awaidah and Mahmoud 2009 (Awaida and Mahmoud 2009)	• Discrete HMMs system • Gradient, concavity and structural features (GSC) • A digit image segmented into a number of frames such that each segment has approximately the same number of pixels	21,120 digit images written by 44 writers • 15,840 images used for training and the remaining 5280 images were used for evaluation	• Misclassification rate of 0.87%	Isolated digits recognition
Mahmoud and Al-Khateeb 2010 (Mahmoud and Al-Khatib 2010)	• Three different classifiers (k-NNs, HMMs, and SVMs) • Log Gabor filters with several scales and orientations were used as features	Isolated digit images from CENPARMI database of Arabic checks:	• Misclassification rate of 1.05% using SVM classifier • Misclassification rate of 2.79% using HMM classifier	Isolated digits recognition

		• 7390 digit images used for training • 3035 digit images used for evaluation	• Misclassification rate of 1.25% using k-NN classifier	
Al-desouky et al. 1991 (El-Desouky et al. 1991)	• Image thinning and normalization before feature extraction • Direction chain code along with character zone and dots information are used as features • Each character image if first associated to a group and then classified using some distance measure computed from the features	Isolated character images written by two writers with specific constraints imposed when writing • Information not available on the database size	• Character error rate (CER) of 6%	Isolated character recognition
Cheriet et al. 2007 (Cheriet et al. 2007)	• Discrete HMMs system • Freeman chain codes (length and direction) from text image's skeleton is used as features • PAWs modelled as HMMs • Number of states in an HMM is decided based on the number of characters in a PAW	Isolated PAW images from CENPARMI database of Arabic checks • Lexicon size of 67 PAWs	• PAW error rate of 26.47% on the evaluation set	Holistic recognition of PAWs
Almuallim and Yamaguchi 1987 (Almuallim and Yamaguchi 1987)	• Syntactic classifier for word recognition • Thinning and segmentation into strokes • Strokes are classified into groups as a first step and features (group specific) are extracted from strokes in order to classify the strokes in the second step • Strokes are combined to form characters using rule based pattern matching • Use of word dictionary as post-processing	• 200 words written by two writers are used for training • Evaluation was performed on 400 word images	• Word error rate (WER) of 9%	Word recognition task

Goraine et al. 1992 (Goraine, Usher, and Al-Emami 1992)	• Syntactic classifier for word recognition • Thinning and segmentation into strokes • Strokes represented as direction codes and classified into 11 primitives • Strokes are combines into characters using rules • A dictionary lookup for words to correct misclassified characters was used as post-processing	180 word images with a total of 600 characters written by three writers were used for evaluation	• WER of 10%	Word recognition task
Abuhaiba et al. 1998 (Abuhaiba, Holt, and Datta 1998)	• Syntactic classifier for character recognition from text line images • Thinning and segmentation into strokes • Segmentation of strokes into tokens • Token recognition • Token recombination to form core shapes • Adding secondary strokes information to core shapes using rules to form characters	Handwritten text line images: • 13 pages written by 13 different writers used for training • 20 pages written by 20 different writers not in training used for evaluation	• CER of 48.9%	• Character recognition from text line images • CER includes both errors and rejections
Dehghan et al. 2001 (Dehghan et al. 2001)	• Discrete HMMs system • An HMM for each word in the lexicon • Stroke width normalization, baseline correction • Histograms of chain-code directions from the image strips within a sliding window are used as features • The number of states in a word HMM is set based on the average number of frames the word has in the training set	17,000 handwritten word images of 198 city names • Lexicon size of 198 names • 60% of the data used for training and the rest 40% is used for evaluation	• WER of 34.95%	Holistic word recognition task
Alma'deed et al. 2002 (Alma'adeed, Higgins, and Elliman 2004)	• Discrete HMMs based multi-stage recognition • An HMM for each word in the lexicon	4700 handwritten words written by 100 writers • Lexicon size of 47 words	• WER of 40% • WER of 31% with some post-processing	Holistic word recognition task

	• Normalization of image height and stroke width, slope correction • Words grouped in the first stage based on features like number and position of dots, HMM systems specific to groups are then used to classify the words	• Two-third of the word images used for training and the remaining one-third was used for evaluation		
Farah et al. 2006 (Farah, Souici-Meslati, and Sellami 2006)	• Multi-classifier system (ANNs, k-NN, and fuzzy k-NN) • Text image binarization and smoothing • Structural features like loops, ascenders, descenders, and dots information • Classifier combination using score summation	4800 handwritten words written by 100 writers • Lexicon size of 48 words • 1200 word images were used for training and the remaining 3600 word images were used for evaluation	• WER of 6%	Holistic word recognition task

Table 4.2: Summary of the representative works on text recognition using the IFN/ENIT database.

System	Key Features	Recognition results (WER)				Remarks
		Train–Test Configuration				
		abc–d	*abcd–e*	*abcde–f*	*abcde–s*	
Menasri et al. 2007 (Menasri et al. 2007)	• Hybrid HMMs/ANNs recognition system • Explicit segmentation of word images into recognition units • An HMM represents a 'letter-body' which is one of the core shape segments after separation of dots and other diacritics • HMMs observation probability distribution was computed using neural networks	12.6	-	-	-	
Benouareth et al. 2008 (Benouareth, Ennaji, and Sellami 2008)	• Semi-continuous HMMs system • Explicit state duration modelling (using Gamma distribution) • Non-uniform sliding windows with windows' width decided using pixel projection information • Character shape HMMs with fixed number of states for all models • Bakis topology • Viterbi training	9.80	-	-	-	
Schambach et al. 2008 (Schambach, Rottland, and Alary 2008)	• HMMs system • Arabic character modelled with parallel path HMMs where each path represents a	-	-	12.78	26.06	It was the winning system in ICDAR2007 competition (Märgner and Abed 2007)

	character shape of the representing character • Model length adaptation • 3 systems combined using weighted majority voting					
Al-Hajj et al. 2009 (Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009)	• Multiple continuous HMMs system • Pixel density and concavity features • Two slanted sliding windows in addition to vertical sliding windows provide features for three different HMM systems respectively • Character shape HMMs with Bakis topology • Explicit white-space modelling between words, and after *alifs* • Classifier combination using an MLP system	9.04	-	-	-	The authors presented the winning system ((El-Hajj, Likforman-Sulem, and Mokbel 2005)) in ICDAR2005 competition (Märgner, Pechwitz, and Abed 2005)
Dreuw et al. 2008 and 2009 (Dreuw, Jonas, and Ney 2008)(Dreuw et al. 2009)	• Continuous HMMs system • Image slice and its horizontal derivatives as feature • PCA for dimensionality reduction • Explicit white-space modelling • Model length adaptation with each character model having additional variants • Additional virtual training samples by shifting the image pixels along y-axis • Use of writing variant probability • Writer adaptive training	5.82	11.22	-	-	
Kessentini et al. 2010 (Kessentini, Paquet,	• Multi stream HMMs system • Slant and slope correction, contour smoothing	-	20.4	17.91	25.49	

and Ben Hamadou 2010)	• Contour and pixel features • Each feature forms a separate stream					
Pechwitz et al. 2012 (Pechwitz, Abed, and Märgner 2012)	• Semi-continuous HMMs • Baseline correction, line thickness normalization • Pixel values from sliding window used as features after dimensionality reduction • Sliding window divided into vertical cells • Character shape HMMs with Bakis topology • Viterbi training	8.2	-	-	-	
Natarajan et al. 2012 (Premkumar Natarajan et al. 2012)	• Continuous HMMs system • Skew and slant correction • Percentile features, angle and concavity features, and derivatives. Feature reduction using LDA • Contextual HMMs (triphones) • State tied mixtures (STM) for different character shape models of a character • Unsupervised writer adaptation (only means updated)	10.6	-	-	-	
Rothacker and Fink 2012 (Rothacker, Vajda, and Fink 2012)	• Bag-of-features HMM • Slant and slope correction • Harris Corners detectors and SIFT descriptors for features computations • PCA for dimensionality reduction	3.8	7.1	9.8	19.3	
Graves 2012 (Graves 2012)	• RNN (BLSTM) system • Image pixels as features • CTC output layer with word dictionary for decoding	-	-	6.63	18.94	The author presented the winning system in ICDAR2009

	• For words with more than one variant, the sum of probabilities for all the variants was used as the probability for the word					competition (Mozaffari and Soltanizadeh 2009)
Parvez and Mahmoud 2013 (Parvez and Mahmoud 2013a)	• Structural classifier • Characters represented using fuzzy polygonal approximation • Nearest neighbour classifier with fuzzy polygonal matching • Lexicon reduction using dots information	-	20.42	-	-	Training set included pre-segmented characters from another database. IFN/ENIT training sets were not used
Azeem and Ahmed 2013 (Azeem and Ahmed 2013)	• Multiple continuous HMMs system • Stroke width and white-space normalization • Character shape models • Concavity and gradient features • Slanted sliding windows (same as Al-Hajj et al. (Al-Hajj Mohamad, Likforman-Sulem, and Mokbel 2009)) • Classifier combination using sum, majority vote, maximum rules	2.3	6.56	6.9	15.2	
Giménez et al. 2014 (Giménez et al. 2014)	• Bernoulli HMMs based system • Binary image pixels as features • Sliding window repositioning based on ink centre-of-mass	4.7	6.1	7.80	15.38	The authors presented the winning system ((Giménez, Khoury, and Juan 2010)) in ICFHR2010 competition (Märgner and Abed 2010)
Abandah et al. 2014 (Abandah, Jamour, and Qaralleh 2014)	• RNN (BLSTM) system • Explicit segmentation of text into graphemes • Statistical, skeleton, boundary, diagonal features, Fourier descriptors	1.04	6.54	7.54	15.20	

	• Word recognition using CTC output layer and dictionary					
Hamdani et al. 2014 (Märgner and Abed 2011)(Hamdani et al. 2014)	• BLSTM (RNNs) in tandem with continuous HMMs • Pixel grey values as features extracted from repositioned sliding windows and PCA for feature reduction for HMM training in step one • RNNs trained features used in the next step to train the HMMs • Gaussians shared between two successive states of HMMs • Viterbi training with MLE criteria • Contextual HMMs (triphones) with decision tree based clustering	-	-	7.80	15.45	The authors presented the winning system in ICDAR2011 competition (Märgner and Abed 2011)
Stahlberg and Vogel 2015 (Stahlberg and Vogel 2015)	• HMMs system with deep neural network training • Baseline and slant correction, image height and line thickness normalization • Pixel values as features after PCA based reduction and segment based features • Deep neural networks based discriminative training after forced alignment using HMMs • The use of space and connector models as proposed in (Ahmad, Fink, and Mahmoud 2014) • Contextual triphones with decision tree based clustering • Writer adaptive training	2.4 (ranges from 2.9 to 2.4)	6.1 (ranges from 6.1 to 6.9)	6.8 (ranges from 6.8 to 7.3)	11.5 (ranges from 11.5 to 12.5)	The results vary for different configurations of features and training strategies

Table 4.3: Summary of the representative works on recognition of text images containing multiple words and on the use of language models.

System	Key Features	Characteristics of the database	Recognition results	Remarks
Saleem et al. 2009 (Saleem et al. 2009)	• Continuous HMMs system • Percentile, angle, correlation, energy, gradient, and GSC features. Feature reduction using LDA • Character shapes as HMMs with position dependent tied mixtures • MLLR based unsupervised adaptation (only mixture means updated) • Ruled line removal • Trigram language model estimated on a text corpus of 90 million words with vocabulary size of 92K words • Forward pass of decoding using bigrams and backward pass using trigrams	DARPA MADCAT dataset: • 8250 documents for training, 218 documents for development, and 224 documents for evaluation	• WER of 30.0%	
Natarajan et al. 2012 (Premkumar Natarajan et al. 2012)	• Continuous HMMs system • Skew and slant correction • Percentile, angle, correlation, energy, and GSC features, and derivatives. Feature reduction using LDA • State tied mixtures (STM) for different character shape models of a character • Contextual HMMs (triphones) • Unsupervised writer adaptation (only mixture means updated)	DARPA MADCAT dataset: • 37,608 documents for training, 868 documents for development, and 885 documents for evaluation	• WER of 25.2%	

	• Trigram language model estimated on a text corpus of 217 million words with lexicon size of 120K words			
Hamdani et al. 2013 (Hamdani, Mousa, and Ney 2013)	• Continuous HMMs systems • Pixel grey values as features extracted from repositioned sliding windows • PCA for feature reduction • Contextual HMMs (triphones) with decision tree based clustering • Language model (n-grams) built using most frequent Arabic words in training corpus + less frequent words were decomposed morphologically into prefixes, roots, and suffixes	1. DARPA MADCAT dataset available under OpenHaRT: • 42K pages for training 470 pages for development 2. The KHATT database: • 9475 lines for training, 1902 lines for development, and 1997 lines for evaluation	1. OpenHaRT database: • WER of 34.1 on constrained task with 94K vocabulary size (90K full words) • WER 25.9 on unconstrained task with vocabulary size of 200K full words 2. KHATT database • WER of 32.5 on constrained task with 15K vocabulary size (10K full words) • WER 26.8 on unconstrained task with vocabulary size of 200K full words	• Constrained task refers to LM training on only the training set of the used database. Unconstrained task uses a text corpus of 1 billion words • It seems for the KHATT database, the lines which included the fixed repetitive text has also been included for training and evaluation • Details on the evaluation set for the OpenHaRT Database are missing
Hamdani et al. 2014 (Hamdani et al. 2014)	• BLSTM (RNNs) in tandem with continuous HMMs • Pixel grey values as features extracted from repositioned sliding windows and PCA for feature reduction for HMM training in step one	DARPA MADCAT dataset available under OpenHaRT: • 42K pages for training 470 pages for development	• WER of 26.8 (CER of 10.1) on constrained task with 94K vocabulary size	Constrained task refers to LM training on only the training set of the used database. Unconstrained task uses a text corpus of 1 billion words (cf.

	• RNNs trained features used in the next step to train the HMMs • Gaussians shared between two successive states of HMMs • Viterbi training with MLE criteria • Contextual HMMs (triphones) with decision tree based clustering • Unsupervised writer adaptation • Language model (n-grams) built using most frequent Arabic words in training corpus + less frequent words were decomposed morphologically into prefixes, roots, and suffixes		• WER of 17.0 (CER of 4.5) on unconstrained task	(Hamdani, Mousa, and Ney 2013))
Hamdani et al. 2014 (Hamdani, Doetsch, and Ney 2014)	• Hybrid HMMs/RNNs system • Pixel grey values as features extracted from repositioned sliding windows for HMM training in step one • Gaussians shared between two successive states of HMMs • Model length adaptation • Emission probability of HMM states simulated using trained LSTM-RNNs in step two • Contextual HMMs (triphones) with decision tree based clustering with clustering questions based on a character's core shape • 4-gram closed vocabulary language model with 400K vocabulary size estimated using a text corpus containing about 1 billion words	DARPA MADCAT dataset available under OpenHaRT: • 42K pages for training, 470 pages for development, and 633 pages for evaluation	• WER of 19.9 (CER of 8.3)	

Cao et al. 2014 (Cao et al. 2014)	• Continuous HMMs system • Percentile feature, gradient and concavity features, and Gabor filter based features. Feature reduction using region dependent transformation (RDT) • State tied mixtures (STM) for different character shape models of a character • Contextual HMMs (triphones) with state tying • Discriminative training • Writer adaptation • Trigram language models and recurrent neural network language models (RNNLM) estimation using training set and Gigaword text corpus and 300K vocabulary size • N-best rescoring using SVM • Revaluation of N-best scores using dots and diacritics free glyph models • Multiple systems (differs in preprocessing) combination using a weighted voting algorithm	DARPA MADCAT dataset: • NIST OpenHaRT 2013 evaluation • Evaluation on uncontrolled set 'i'	• WER of 7.4% on NIST OpenHaRT 2013 evaluation • WER of 22.1% on the uncontrolled set 'i'	
Bluche et al. 2014 (Bluche et al. 2014)	• Hybrid HMM/MDLSTM (RNNs) system • Image pixels as features • 4 LSTM layers applied in parallel, one for each scanning direction • Initial training on isolated words followed by text line images • 11 different RNNs trained, each with different random seeds and different initialization data	DARPA MADCAT dataset available under OpenHaRT	• WER of 20.1 on constrained task • WER of 18.4 on unconstrained task	Constrained task refers to LM estimation on the training set. Unconstrained task uses Arabic GigaWord text corpus for language model estimation

	• Character shapes as models (recognition units) • Each character shape represented by a single state HMM whose emission model is the trained MDLSTM system • Use of 3-gram language model with 60K vocabulary size • Stepwise classifier combination			
Moysset et al. 2014 (Moysset et al. 2014)	• Hybrid HMM/MDLSTM (RNNs) system • Raw image pixels as features • 4 LSTM layers applied in parallel, one for each scanning direction • Initialization on clearer image samples • Augmented training data used by applying several transformation to the original text images • Character shapes as models (recognition units) • Each character shape represented by a single state HMM whose emission model is the trained MDLSTM system • Hybrid word/PAW language model	The Maurdor database: • 9729 text zones for training, 1835 text zones for development, and 1582 text zones for evaluation	• WER of 29.5 using 3-gram hybrid word/PAW language model	Text images were at zone level with many images having more than one text line. Thus, segmentation of text zone into lines was part of the recognition task
BenZeghiba et al. 2015 (BenZeghiba, Louradour, and Kermorvant 2015)	• Hybrid HMMs/RNNs system • An RNNs system (MDLSTM) estimated the emission probabilities for HMMs • Initial seed model trained on OpenHaRT 2013 dataset • Hybrid word/PAW language model where most frequent words are included in the recognition lexicon and the remaining	1. The Maurdor database: • 13,496 line images for training and 1125 line images for development, and 2093 line images for evaluation	1. The Maurdor database: • WER of 33.5 using 4-gram PAW language model trained on the training and the development set	Synthesized training data, after applying several transformation to the original training images, was added to train the system

	words are broken into PAWs. Words are reconstructed after PAW recognition	2. The KHATT database: • 4428 line images for training, 876 line images for development, and 959 line images for evaluation	• WER of 33.2 using 3-gram hybrid word/PAW language model trained on training and development set 2. KHATT database • WER of 30.9 using 4-gram PAW language model trained on the training and the development set • WER of 31.3 using 3-gram hybrid word/PAW language model trained on training and development set	
Stahlberg and Vogel 2015 (Stahlberg and Vogel 2015)	• HMMs system with Deep neural network training • Baseline and slant correction, image height and line thickness normalization • Pixel values as features after PCA based reduction and segment based features • Deep neural networks based discriminative training after forced alignment using HMMs	1. The KHATT database: • 9462 line images for training, 1899 line images for development, and 1996 line images for evaluation	• WER ranges between 30.5 to 31.6 for different configurations of features and training strategies	It seems for the KHATT database, the lines which included the fixed repetitive text has also been included for training and evaluation

	• Contextual triphones with decision tree based clustering • Writer adaptive training • Trigram language model estimated from KHATT training corpus	

5 Modeling Options for Arabic Text Recognition

We have seen in Chapter 3 a brief overview of the Arabic script and its peculiarities and some of the resulting challenges related to Arabic text recognition. Moreover, we have also seen in Chapter 4 how the researchers addressed some of these issues when dealing with offline handwritten Arabic text recognition. Based on these discussions we concluded that, although a lot of work and effort have been put in the area of handwritten Arabic text recognition, there are aspects which are not well explored. One of these aspects is the modeling choice for Arabic text recognition. Investigating the modeling options for Arabic text recognition is one of the main objectives of the present thesis.

In this chapter, we will present the different modeling options we have investigated during the course of this work. First, in Section 5.1, we present some discussions which motivated us for these investigations. In the next three sections, we will present the three different, but related, approaches to modeling which we have investigated. We will discuss the modeling approaches and the methods and techniques involved to use these modeling approaches for the text recognition tasks. Finally, in the last section, we present some other aspects which we have investigated for training an HMM-based recognizer which are related to multi stream HMMs and modeling contexts using contextual HMMs. The experiments and the results related to these modeling and training options will be presented in Chapter 7.

5.1. Motivation for this work

As presented in Chapter 4, some researchers, mostly in the initial days of Arabic text recognition, selected characters as modeling units. Due to the fact that Arabic characters have many position dependent shapes, using characters as models is not the best choice. Some other researchers used Arabic words or PAWs as the modeling units. This approach has its own set of problems as was discussed in Section 4.2. There are some other approaches to modeling but it seems that either they were not explored thoroughly enough or they were not very successful due to reasons like poor recognition performance, or the need to explicitly segment the text into these recognition units.

The most successful and commonly used approach is to model Arabic character shapes as separate models. Although this approach works well in general, there are issues related to this modeling approach. The number of modeling units increases four-fold from 28 (i.e., the number of Arabic characters) to around 100 excluding the digits and other special characters. This results in a bulky recognizer with a large model set. A large number of modeling units also means the need for large amounts of training data so that each model gets trained adequately. Lack of enough training data is a common issue in text recognition research.

Moreover, as with other languages, the frequency of different characters appearing in any text corpus is not uniformly distributed. In fact, based on text analysis of an Arabic corpus containing more than five million characters (Intellaren 2016), it was found that not only the characters are unevenly distributed but also the distribution is highly skewed. The two most common characters, i.e., *alif* (ا) and *lām* (ل), constituted almost a quarter of all the texts. On the other hand, the five least occurring characters (out of the total of 28 characters) constitute only around 2% of the entire text. The frequency of character *ẓā'* (ظ) was almost 70 times lower than the frequency of character *alif* (ا). Now, if we consider the distribution of character shapes—the preferred choice of modeling unit—instead of characters, the distribution will be even more skewed. Some character shapes from the low frequency characters will have very few occurrences and, thus, the issue of adequately training them is present. These considerations led us to investigate some alternative representations for the modeling units which can possibly alleviate the above mentioned concerns. In the following sections, we will present the alternative modeling options that we have explored.

5.2. Sub-character modeling for Arabic text recognition

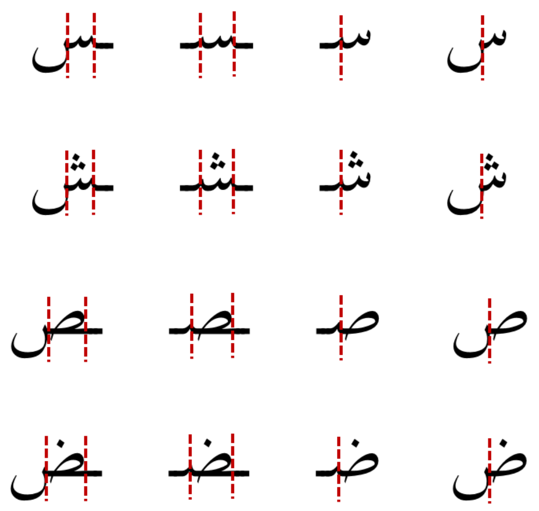
Most of the Arabic characters are wider when compared to the Roman characters. Moreover, some character shapes have high visual similarity with other character shapes of the same character as well as with the character shapes of other characters. Figure 5.1 illustrates these observations by showing four different Arabic characters and their position dependent shapes. From the figure we can clearly see a high degree of similarity between different character shapes. In fact, many of the character shapes differ from each other only due to the presence of horizontal connecting strokes (i.e., *Kashidas*). But, it should be noted that it is important to model even the small differences between the character shapes for optimal recognition performance as was stated/demonstrated in (Schambach, Rottland, and Alary 2008).

Characters	Isolated	Beginning	Middle	Ending
س	س	سـ	ـسـ	ـس
ش	ش	شـ	ـشـ	ـش
ص	ص	صـ	ـصـ	ـص
ض	ض	ضـ	ـضـ	ـض

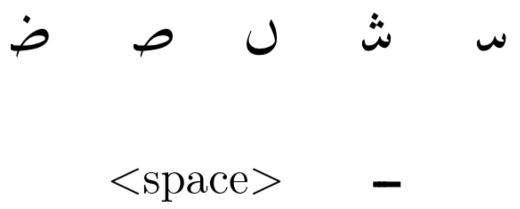
Figure 5.1: Sample character shapes in Arabic illustrating common patterns. (Figure adapted from (Ahmad et al. 2013))

In order to exploit the similarity in patterns between different characters and their position dependent shapes, we propose what we term as *sub-character* modeling. The idea of sub-characters (and sub-strokes) modeling is present in the literature mainly for the task of online text recognition of East-Asian scripts like Kanji. Nakai et al. (Nakai et al. 2001) presented sub-stroke HMM modeling for online Kanji handwritten text recognition. Kanji has more than 6000 characters and using a separate HMM for each character leads to a huge system with a large model set. Thus, the idea was to represent these characters using a set of 25 elementary sub-strokes. A hierarchical dictionary was constructed which included a mapping of sub-strokes to strokes which are, in-turn, mapped to Kanji sub-characters which were, finally, mapped to Kanji characters. The main motivations stated behind sub-stroke modeling were: to have a compact recognizer consuming less memory, faster recognition as a result of efficient network search involving sub-strokes, and the need for less training data. As stroke orders can vary while writing a character, multiple variations can be added to the dictionary. Automatic generation of multiple definitions for dictionary entry in terms of stroke order was presented by the authors in (Nakai, Shimodaira, and Sagayama 2003). Tokuno et al. (Tokuno et al. 2002) presented contextual sub-stroke modeling as an extension to the work presented in (Nakai et al. 2001). It was stated that the presented 25 elementary strokes which can construct the Kanji characters are influenced by their neighboring strokes and, thus, for recognizing real and challenging handwriting, it was necessary that this context is modelled. Bottom up data driven clustering was not feasible to merge the similar states as the amount of training data was not sufficient to robustly train all the contextual forms before tying the states. Thus a top-down approach of successive state splitting was used instead. Hu et al. (Hu, Gek Lim, and Brown 2000) presented sub-character HMMs for online handwriting recognition of isolated digits, characters, and words. The two main reasons stated for using sub-character/stroke models were that it results in a reduced model set and that it will need fewer training samples. Characters are defined in terms of strokes. Each character has multiple definitions due to the possibility of different stroke orders.

The notion of sub-character (or sub-stroke) modeling in online domain is somewhat different than what we propose for offline text recognition. In online domain, the timing information for strokes is available which can be utilized relatively easily to logically define a character (or word) into strokes. Whereas, in offline text recognition such timing information is not available and the division of characters or words into smaller units is purely visual and should satisfy the sequencing constraint from the sliding windows' perspective. The main idea of our sub-character modeling is to have character segments instead of characters shapes as HMMs. Splitting the character shapes will allow sharing of parameters. The Arabic characters and their position dependent shapes can, then, be constructed from these fewer number of sub-characters which results in a compact recognizer with reduced model set. To illustrate this idea, in Figure 5.2, we present the same four characters as was shown in Figure 5.1 along with the sub-character units that can be used to reconstruct all the position dependent character shapes for these four characters. By using character shapes as modeling units, one ends up having 16 different HMMs to model these four characters. Using the presented sub-character patterns, these 16 different character shapes can be reconstructed using only five different sub-character patterns in addition to a *white-space* model and a *connector* model. These two models are shared by sub-character representations of many other characters. It is important to note that the technique proposed here does not require explicit segmentation of characters into these sub-character units as is the case with other segmentation based approaches like the ones discussed in (Bose and Kuo 1994; Lorigo and Govindaraju 2006).



(a)

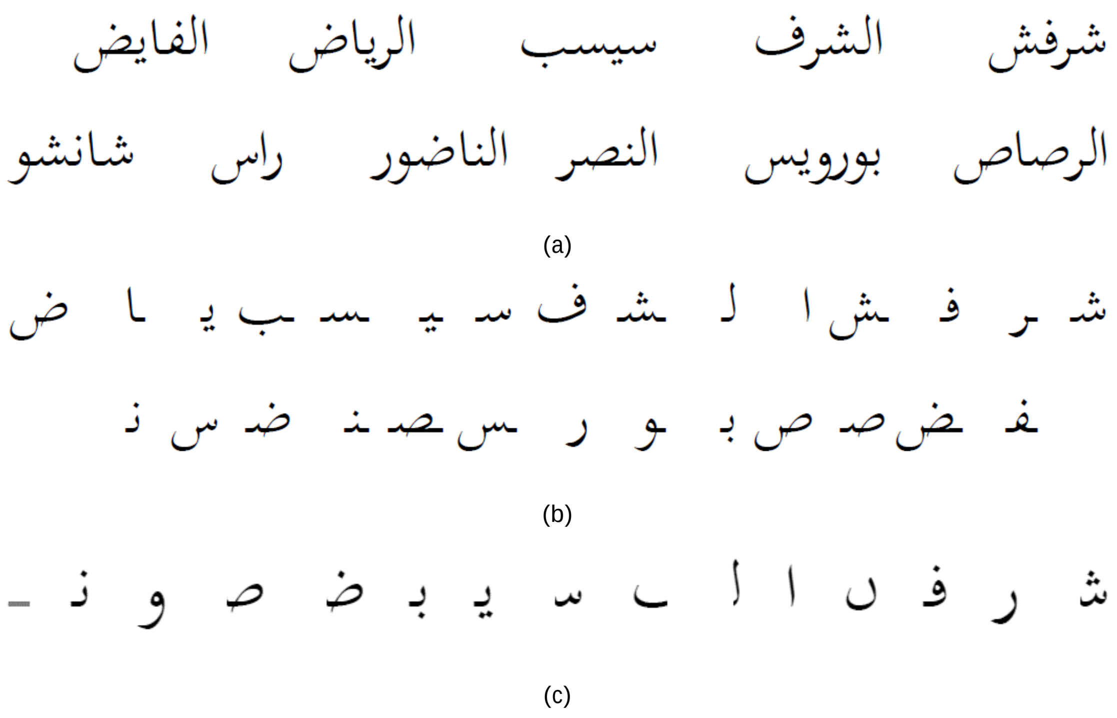


(b)

Figure 5.2: The 16 character shapes in (a) can be represented using only five sub-character shapes as illustrated in (b). (Figure adapted from (Ahmad et al. 2013))

Figure 5.3 illustrates the benefit of sub-character modeling using some word examples. In Figure 5.3 (a), a number of Arabic words are presented. These words can be constructed using 28 character shapes as listed in Figure 5.3 (b). The same words can be constructed using only 15 sub-character shapes, including the connector stroke, as listed in Figure 5.3 (c).

Many of these similar patters across different characters can be captured in the horizontal writing direction. This is important because in offline handwritten text recognition, we do not have the sequence information regarding the writing stroke which is, otherwise, available in online handwriting recognition tasks. Thus, the patterns are normally converted into features using the sliding windows which run across a text image in the writing direction. The HMM models for sub-characters can learn the patterns automatically during the embedded training as long as they have been defined adequately in the dictionary. Once all the different sub-character patterns have been identified for the Arabic characters, creating the complete HMM structure is a straightforward process. Character models can be constructed by concatenating the sub-character models. For lexicon based recognition, the structure can be extended hierarchically. Figure 5.4 illustrates an example HMM structure for building the different position dependent variations of the character *Seen* (س).



شرفش الشرف سيسب الرياض الفايض
الرصاص بورويس النصر الناضور راس شانشو
(a)
(b)
(c)

Figure 5.3: Sample words in (a) can be constructed using 28 character shapes listed in (b) or using the 15 sub-character shapes listed in (c).

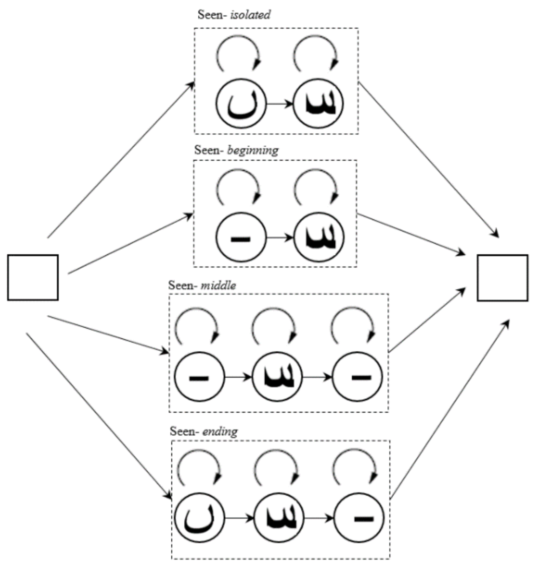


Figure 5.4: An illustration of constructing character models using sub-character HMMs. (Figure adapted from (Ahmad et al. 2013))

5.2.1. Special 'connector' and 'white-space' models

As we have briefly presented above, apart from the different sub-character patterns for the Arabic script, we have also defined two special models. One is for the *Kashida* stroke connecting two characters which we term as the *connector* model and the other is for the background space which we term as the *white-space* model or simply the *space* model. Many character shapes differ from other character shapes of the same character by the absence (or presence) of the *Kashida* stroke. This connecting stroke plays an important role in Arabic handwriting due to the cursive nature of the script. The *Kashida* stroke is a highly variable stroke which, in some cases, may be just a small horizontal glyph; whereas, in other instances it may be a long stretching stroke and it may even be wider than the average character widths. Additionally, in some case, the *Kashida* stroke might almost be missing between two connecting characters when the text is written compactly. Several factors effect this variability including the writers' preferences and the pairs of connecting characters. Figure 5.5 illustrates the variability of *Kashida* stroke with the help of some examples.

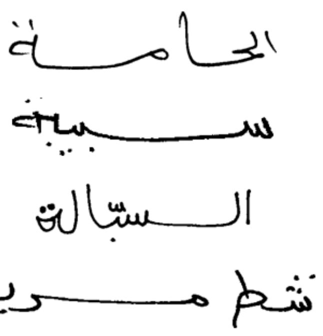
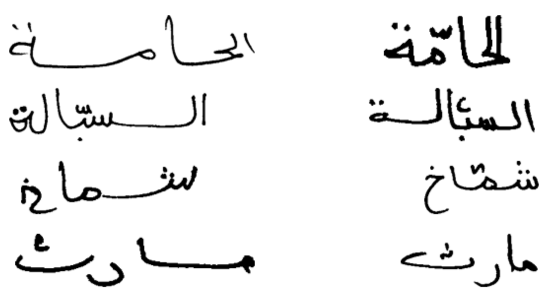
Observations related to *Kashida* stroke	Text image samples
Long *Kashida* strokes	
Same words written in different *Kashida* styles based on writers' choice	

Figure 5.5: Some examples to illustrate the variability in the *Kashida* stroke. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

Dealing with background white-spaces is also not a trivial issue in Arabic handwriting. White spaces in Arabic texts do not appear only between words but also between PAWs. Moreover, white spaces between two words or PAWs are not very uniform in Arabic handwriting. In principle, any character shape having the ending or the alone position should be followed by white-spaces in the text. To address white-spaces in Arabic text recognition, Dreuw et al. (Dreuw, Jonas, and Ney 2008) presented white-space modeling by adding variants to the lexicon dictionary which contain explicit white-space models between words and between PAWs. Using writing variants containing explicit white-space models as presented in (Dreuw, Jonas, and Ney 2008) does improve the recognition performance as was demonstrated in the paper but, it is not the best solution to the problem. A hard decision needs to be made during recognition between selecting a definition that contains white-space models and a definition that does not contain white-space models. Whereas, in handwritten Arabic text, it is easy to find situations where white-spaces are present between some PAWs but are absent between other PAWs in the same text line as is illustrated in Figure 5.6 (a). Moreover, the width of white-space between two PAWs may be similar, or even larger, to the width of white-space between two words as is illustrated in Figure 5.6 (b). Last but not the least, in some situations it may be difficult to find white-spaces between two words but there might still exist white-space between two PAWs (in the same text line) as is illustrated in Figure 5.6 (c). Based on these observations we concluded that a more robust approach to white-space modeling is needed when dealing with handwritten Arabic text recognition.

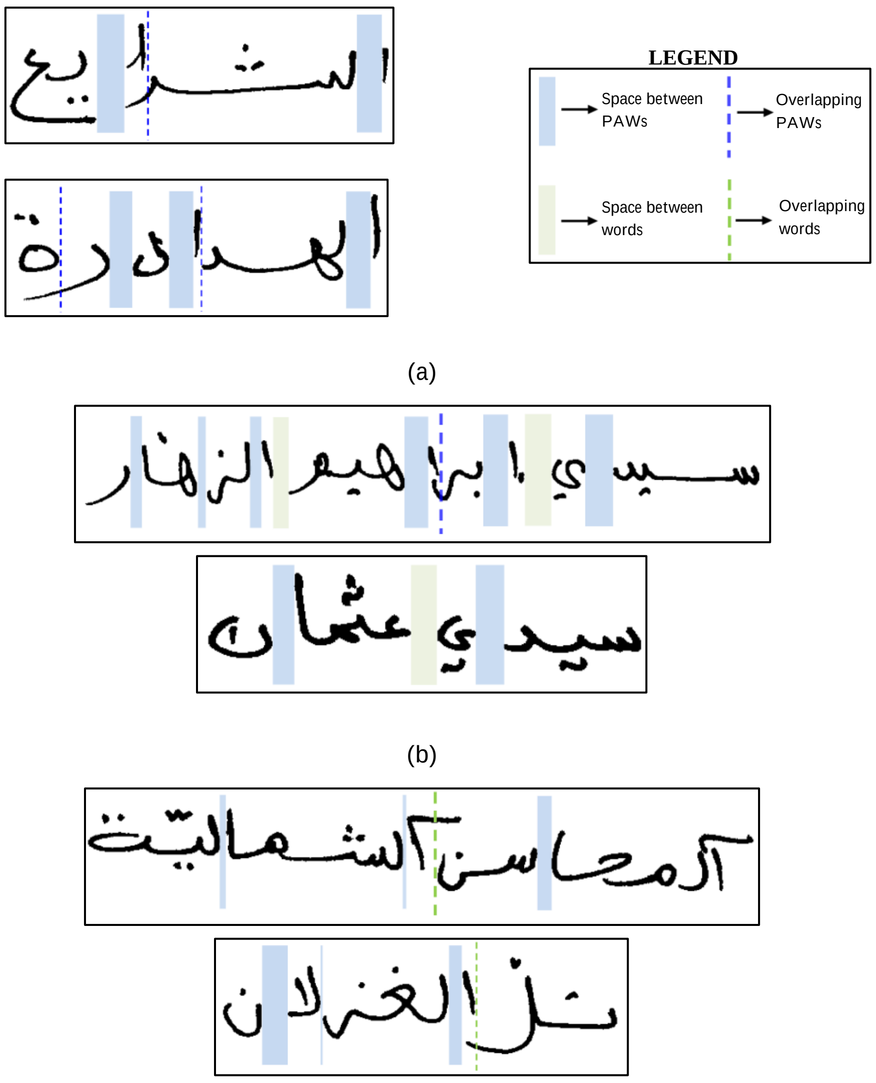


Figure 5.6: Illustrations of problems associated with white-spaces in handwritten Arabic texts. (Figure adapted from (Ahmad, Fink, and Mahmoud 2014)).

We propose a single state white-space and connector models for Arabic text recognition. Additionally, we propose a special structure for these two models such that the models can be used during recognition in some instances and can be skipped in other instances. White-space model can be skipped when white spaces are not present between two words or PAWs whereas the connector model can be skipped when the *Kashida* stroke is either two small or is entirely missing between two characters. The idea of a skipping model was adapted from the concept of *tee* models for short pauses in speech recognition (Young et al. 2002 pages 109–110; Shih, Narayanan, and Kuo 2003). We allow transition from non-emitting entry states to the non-emitting exit states for the two models thereby allowing a possibility to skip the emitting state in some situations. The models are trained using the training data so that it learns the skip probability along with the other model parameters. This way, no hard decisions need to be made to include the space model (and the connector) between two PAWs (between two connected characters in the case of connector model) during the recognition stage. The model will be included, or skipped, based on the model parameters and the observation sequences. Thus, the skip decisions are embedded as a part of the model. In order to guide the training of the space model, we added a special space model both at the beginning and the end of each text line. This model has a rigid structure such that it uses only the first and the last observation sequence from the text line image during the training stage. The beginning and end of the text line image was padded with background patches (i.e., white-space). The state of this special space model was tied with the state of the space model for robust training of the space model. Figure 5.7 illustrates our space modeling idea. As we will see in Section 5.5.2, the use of connector model also seems to help in contextual modeling as the connector captures the connecting context between two characters. Key steps in sub-character modeling based Arabic text recognition are presented in Figure 5.8.

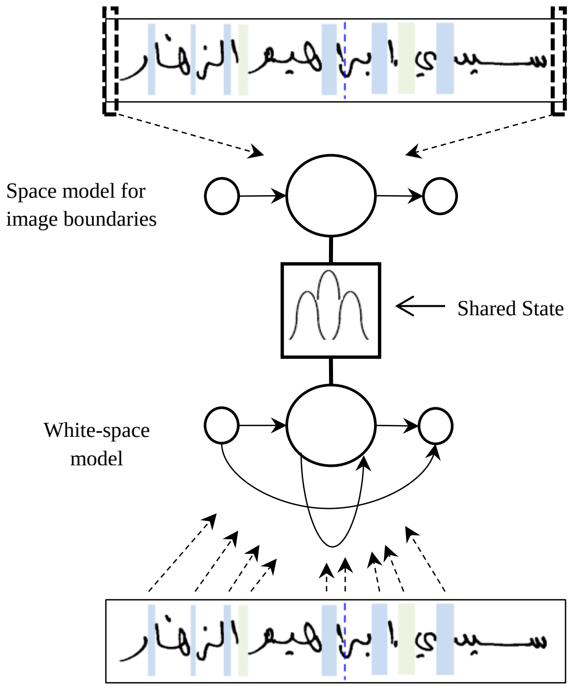


Figure 5.7: Illustration of the proposed white-space model. (Figure adapted from (Ahmad, Fink, and Mahmoud 2014)).

Data Preparation:

1. Training annotation: The character-shape-based annotations for the training set images are converted to sub-character representation by utilizing the character shape to sub-character mapping.
2. Dictionary: Words (or characters for character recognition tasks) are defined as a sequence of sub-characters in the dictionary.

Training:

1. The *connector* and the *white-space* models are defined as single state models.
2. All the sub-character models are initialized using the initialization procedures as presented in Section 2.5.
3. Transitions are added from non-emitting entry states to the non-emitting exit states of the *connector* and the *white-space* models.
4. All the models are iteratively trained using the Baum-Welch training algorithm.

Decoding:

1. Standard decoding using the Viterbi algorithm utilizing the modified dictionary.

Figure 5.8: Key steps in the presented sub-character modeling based Arabic text recognition.

5.3. Modeling core shapes and diacritics separately

In this section, we will present the second modeling approach we investigated and the text recognition framework incorporating this new modeling option. This modeling options comes from the observation that many characters in the Arabic script share the same core shapes (the *Rasm*) but differ only based on the numbers and positions of dots and other diacritics. Thus, if we can separate the core shapes from the diacritics, we can end up having a significantly reduced model set. Using the same word examples as the ones presented in Figure 5.3, we can notice that the words can be constructed using 19 core shapes (and four diacritics: one-dot above, three-dots above, one-dot below, and three-dots below) as illustrated in Figure 5.9.

To have a comprehensive look, in Figure 5.10, we present the different character-shape groups that share the same core shapes in the Arabic script. On counting the number of unique core shapes, we find that the 121 different character shapes in the Arabic script, as shown in Figure 5.10, have a total of 58 unique core shapes. Apart from the core shapes, there are a total of eight different diacritics including the *Shadda* and the *Hamza* diacritics. Thus, if we can separate the diacritics from the core shapes, we can end up with a much smaller number of classes to model and to train.

Based on these observations, we investigated a multi-stage text recognition approach where a core-shape recognition system is trained separately from a diacritics recognition system and the final text recognition hypothesis is made by combining the results from the two systems. The details of the multi-stage recognition framework is presented in the next sub-section.

5.3.1. Multi-stage recognition of handwritten Arabic text

The main idea is to model the core shapes separately from the diacritics, and later combine the information from both sources to make the final text recognition hypothesis. Thus, we will have two separate HMM systems: one for the core shapes and the other for the diacritics. These two systems will be trained separately and independently from each other. Recognition results from the two systems are integrated during decoding based on the framework presented in Section 5.3.1.2.

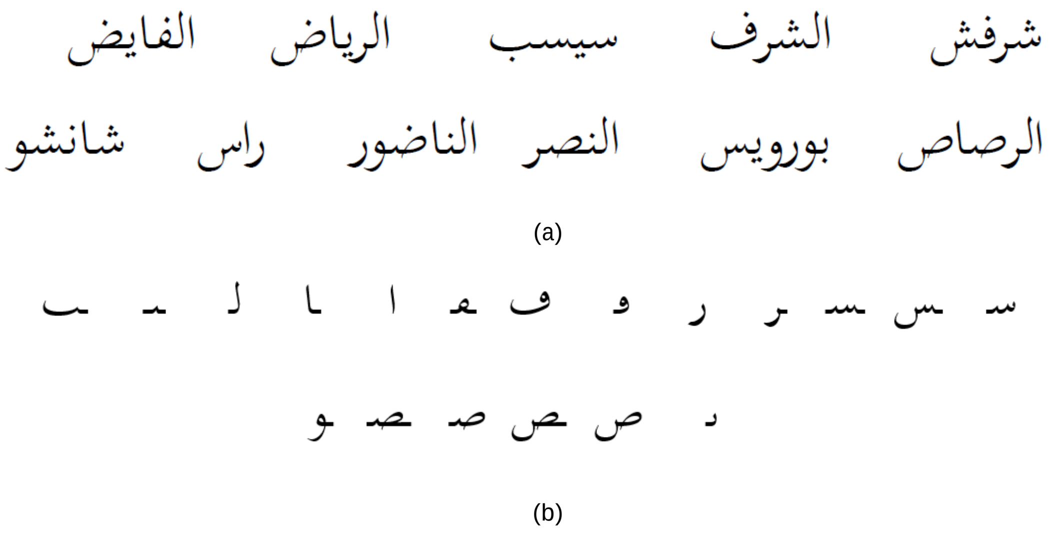

شرفش الشرف سيسب الرياض الفايض
الرصاص بورويس النصر الناضور راس شانشو
(a)
(b)

Figure 5.9: Sample words in (a) can be constructed using the 19 core shapes (and four different diacritics) listed in (b).

Isolated character shapes	Core shapes	Beginning character shapes	Core shapes	Middle character shapes	Core shapes	Ending character shapes	Core shapes
ا أ إ آ	ا					ـا ـأ ـإ ـآ	ـا
ب ت ث	ٮ	بـ تـ ثـ يـ ئـ نـ	ٮـ	ـبـ ـتـ ـثـ ـيـ ـئـ ـنـ	ـٮـ	ـب ـت ـث	ـٮ
ى ي ئ	ى					ـى ـي ـئ	ـى
ن	ں					ـن	ـں
ج ح خ	ح	جـ حـ خـ	حـ	ـجـ ـحـ ـخـ	ـحـ	ـج ـح ـخ	ـح
د ذ	د					ـد ـذ	ـد
ر ز	ر					ـر ـز	ـر
س ش	س	سـ شـ	سـ	ـسـ ـشـ	ـسـ	ـس ـش	ـس
ص ض	ص	صـ ضـ	صـ	ـصـ ـضـ	ـصـ	ـص ـض	ـص
ط ظ	ط	طـ ظـ	طـ	ـطـ ـظـ	ـطـ	ـط ـظ	ـط
ع غ	ع	عـ غـ	عـ	ـعـ ـغـ	ـعـ	ـع ـغ	ـع
ف	ڡ	فـ قـ	ڡـ	ـفـ ـقـ	ـڡـ	ـف	ـڡ
ق	ٯ					ـق	ـٯ
ك	ک	كـ	كـ	ـكـ	ـكـ	ـك	ـک
ل	ل					ـل	ـل
م	م	مـ	مـ	ـمـ	ـمـ	ـم	ـم
ه ة	ه	هـ	هـ	ـهـ	ـهـ	ـه ـة	ـه
و ؤ	و					ـو ـؤ	ـو
لا لأ لإ	لا					ـلا ـلأ ـلإ	ـلا

Figure 5.10: Character shape to core shape mapping for Arabic script. (Figure adapted from (Ahmad and Fink 2015a)).

In order to model and train the core shapes separately from the diacritics, we need to separate them from the text images. Thus, a text image will be split into two images: one containing the core shapes and the other containing the diacritics. We will present our core shape and diacritics separation algorithm shortly. We also need to modify the transcriptions for the text images of the training set. Two separate transcriptions need to be generated, one for the core shapes and the other for the diacritics. Modifying the transcriptions is straightforward based on the script knowledge and the mapping as illustrated in Figure 5.10.

Once we have the text images containing the core shapes and the accompanying transcriptions, the core-shape HMMs can then be trained using the standard training procedures. Similarly, training a diacritics recognition system is straightforward too; although, it should be noted that appropriated features need to be computed from the diacritic images which may not, necessarily, be the same set of features which are computed from the core shape images. We will discuss feature extraction in more details in Chapter 7 when we present our experiments.

5.3.1.1 Core Shapes and diacritics separation algorithm

Before going into the details of the algorithm, we would like to point out that, in principle, separating the core shapes from the diacritics is not a very difficult task for the Arabic script. The reason behind this is the fact that the average component sizes of the different diacritics are too small as compared to the component sizes of the core shapes. Thus, doing some basic component analysis should work quite robustly for the Arabic script. That being said, it is also true that the task is not that trivial once it comes to separation of core shapes and diacritics from handwritten text images. The most common problem faced in handwritten text scenario is the fact that the diacritics are sometimes sticking to the core-shape glyph as a single component (this issue was discussed in Section 3.5 and an example of this situation was illustrated in Figure 3.12). Also the issues of broken or incomplete strokes and the presence of *alifs* (ا) in the alone form add to the problem. Thus, a simple component analysis is not sufficient and the use of some additional heuristics becomes inevitable.

Our algorithm for separating core shapes and diacritics, along with two illustrative examples, is presented in Figure 5.11. First, we compute the average size of all the components in the input text image. Next, all the components whose sizes are less than the average component size are removed from the original image and are placed in a new empty image. This new image now contains all the possible diacritics in addition to a few core shapes. This image, now, undergoes a second analysis stage to filter out the core shapes. Vertical strokes, representing the character *alif* (ا) or broken stroke segments from characters like *ṭā'* (ط), are among the most common core shapes that get added to the list of probable diacritics and, accordingly, need to be removed. These strokes can be identified by their height-to-width ratio (if the height of the component is twice, or more, its width) and by its size (the component is not too small like the dots). Another set of strokes that belong to the core shapes, but can get added to the list of possible-diacritics, are the characters like *dāl* (ذ), or the broken strokes around the baseline. These strokes can be identified by their sizes, which are significantly bigger than the diacritics, and based on the fact that these strokes are not too far from the core text region. The remaining components in the new image are expected to be the diacritics and are saved as such. The core-shape image is obtained from the original image by removing the components identified as diacritics.

5.3.1.2 Multi-stage text recognition framework using the core-shape system and the diacritics system

In this section, we will present our multi-stage text recognition system based on the idea of modeling and training the core shapes and the diacritics separately. In the first stage, an input image is processed to separate the core shapes from the diacritics which results in two separate images: one for the core shape and the other for the diacritics. Next, features from the core-shape image is fed to the core-shape HMMs and the recognition hypothesis is generated. Now, depending on the text recognition task, i.e., character recognition vs. word recognition, a slightly different approach is taken. First we will detail the character recognition task (i.e., hypothesizing characters for text line images using the system trained on text line images) and, later, we will discuss the word recognition task.

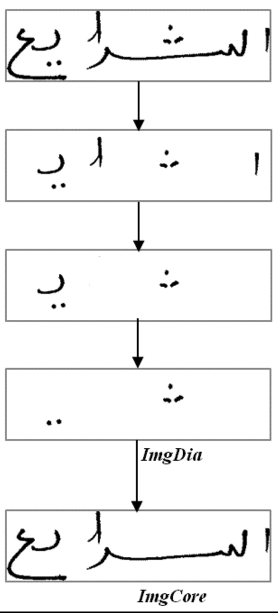
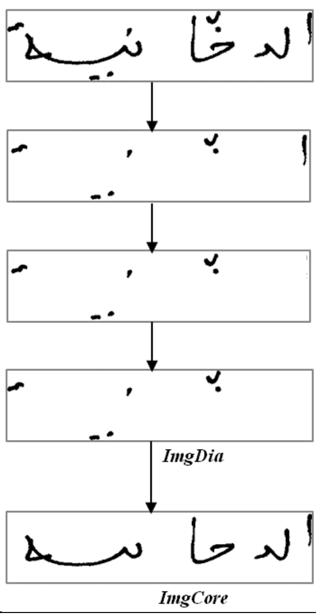
Algorithm	Illustration Example 1	Illustration Example 2
Input: Original image (*img*) **Output**: An image having core shapes (*imgCore*), and an image having diacritics (*imgDia*) 1. ***img*** : The original image 2. ***imgDia*** : Includes all the components from the original image whose size is less than the average size of components in the original image 3. Remove from *imgDia* a component 'c', if: • The length of 'c' is > *thresh1*, AND • The length is at least twice the width (For *alifs* and characters having vertical long stroke) 4. Remove from *imgDia* a component 'c', if: • The width of 'c' is > *thresh2*, AND • It is within the core text region, i.e., it is not too far away from the baseline (For fragmented core shapes or small isolated characters like *dāl*) 5. ***imgCore*** : *img* − *imgDia*		

Figure 5.11: Core shapes and diacritics separation algorithm. (Figure adapted from (Ahmad and Fink 2015a))

Character recognition: In case of character recognition, the core-shape system generates hypothesis for the core shapes. Some core shapes, representing characters that do not have diacritics over them (like *lām* and *mīm*), will have exactly one character represented by them. Other core shapes will be representing more than one character. Thus, for each core shape in the recognition hypothesis, a lookup is used to generate all the possible character alternatives represented by that core shape. This information is used to build the character recognition network for the diacritics system. The recognition network along with the features from the diacritics image is used by the diacritics system to generate the final character hypothesis. Thus, the diacritics system assists the primary core-shape system. Its search space is limited by the output of the core-shape system.

As an extension to the above described method, it is also possible to generate N-best list when performing core-shape recognition. The justification for doing this is to explore more alternatives instead of only a single hypothesis for a core shape. Many of these alternatives can be, relatively, easily ruled out by the diacritics system. Thus, there might be some benefit in expanding the search space for the diacritics system. The multi-stage character recognition framework is illustrated in Figure 5.12.

Word recognition: In the case of word recognition, we construct a dictionary that defines every word as a sequence of only the core shapes. This dictionary can easily be built by utilizing the character to core-shape mapping. Feature from the core-shape image is fed to the core-shape system to generate word hypothesis by utilizing the dictionary. Again, an N-best list of words can be generated instead of hypothesizing only a single word. The list of hypothesized words is then fed to the diacritics system to generate recognition scores for all the words. Scores from the core-shape system and the diacritics system are added for each word respectively. Finally, the word having the best score is output as the final recognition hypothesis. The need to add scores from both the system in the case of word recognition is important because there are words in Arabic which consists of no diacritics (like سلام) and thus the score from the diacritics system alone will not be helpful in making a final decision. Moreover, there can be two different words (like الحب and اللب) having exactly the same diacritics distribution. Thus, the scores from the core-shape system as well as the diacritics system are combined for outputting the final word hypothesis. Although, it should be noted that, here too, the diacritics system's search space is limited by the output of the core-shape system. Figure 5.13

illustrates our multi-stage word recognition framework with the help of an example. Key steps in the multi-stage Arabic text recognition are presented in Figure 5.14.

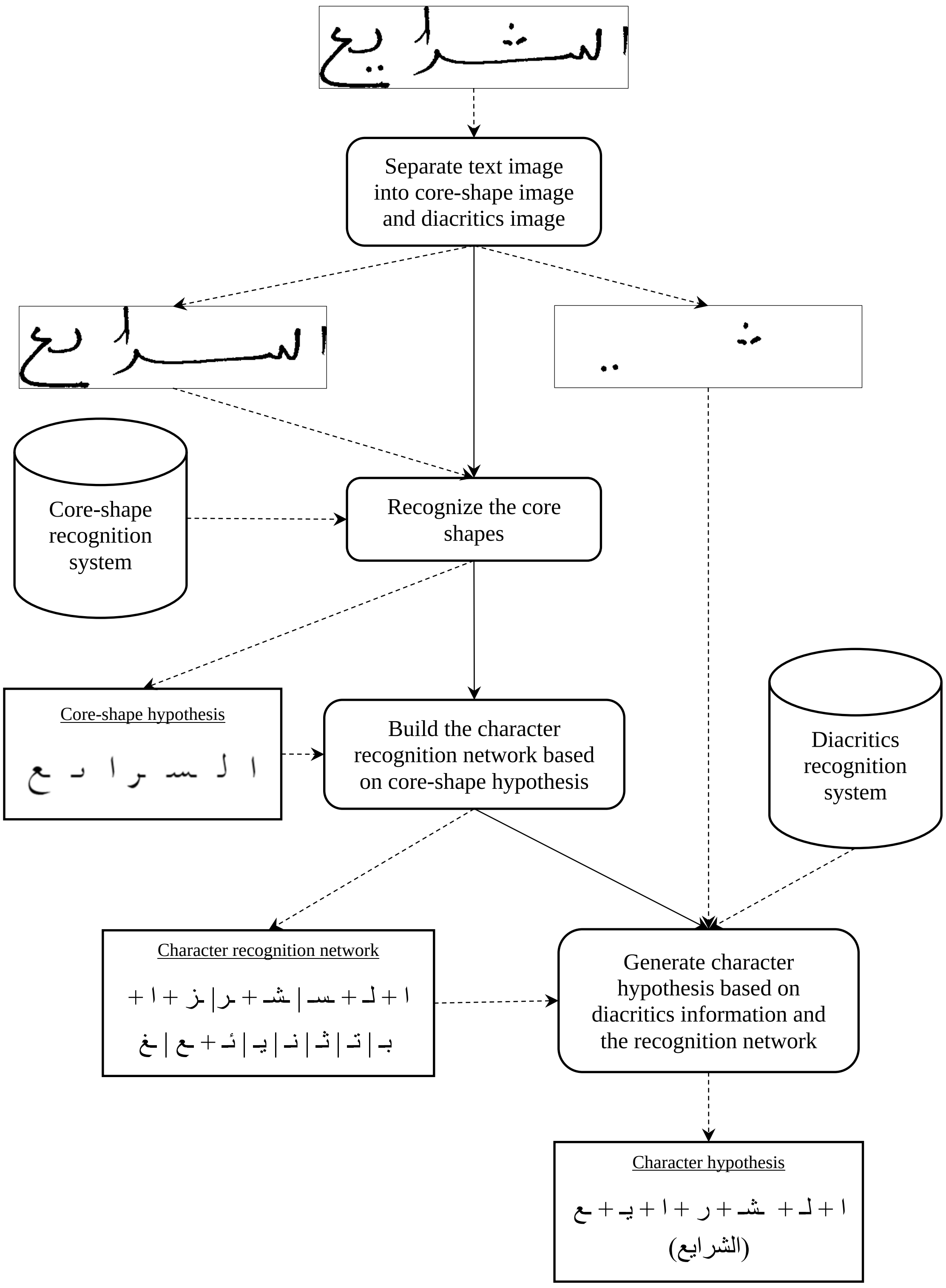
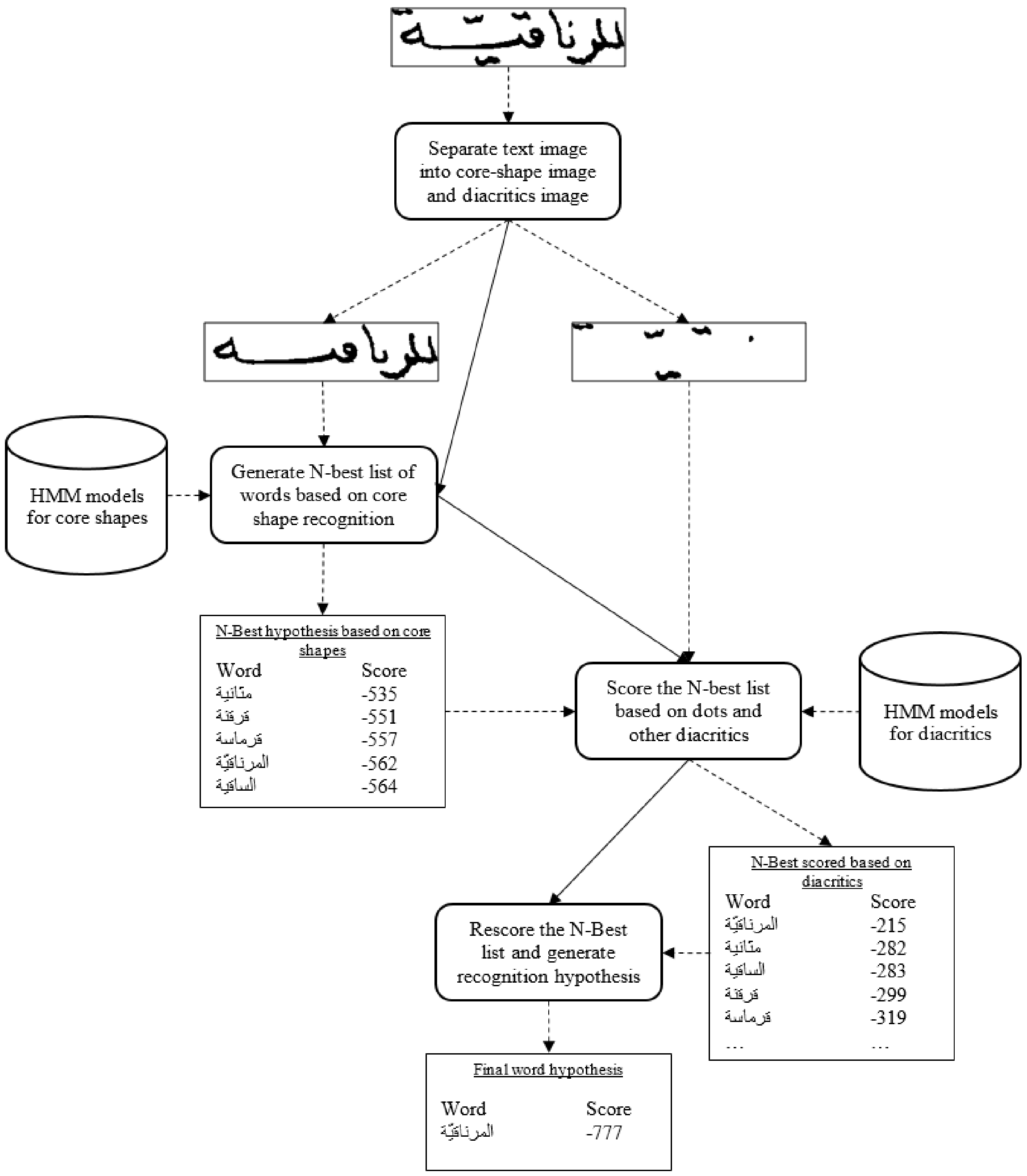

Separate text image into core-shape image and diacritics image
Core-shape recognition system
Recognize the core shapes
Core-shape hypothesis
ا ل سـ ـر ا د ع
Build the character recognition network based on core-shape hypothesis
Diacritics recognition system
Character recognition network
ا + لـ + سـ | شـ + ـر | ـز + ا +
بـ | تـ | ثـ | نـ | يـ | ئـ + ـع | ـغ
Generate character hypothesis based on diacritics information and the recognition network
Character hypothesis
ا + لـ + شـ + ر + ا + يـ + ع
(الشرايع)

Figure 5.12: The multi-stage character recognition framework. (Handwritten text image source: IFN/ENIT (Pechwitz et al. 2002)).


Separate text image into core-shape image and diacritics image
HMM models for core shapes
Generate N-best list of words based on core shape recognition
Score the N-best list based on dots and other diacritics
HMM models for diacritics
Rescore the N-Best list and generate recognition hypothesis

N-Best hypothesis based on core shapes

Word	Score
متانية	-535
قرقنة	-551
قرماسة	-557
المرناقيّة	-562
الساقية	-564

N-Best scored based on diacritics

Word	Score
المرناقيّة	-215
متانية	-282
الساقية	-283
قرقنة	-299
قرماسة	-319
…	…

Final word hypothesis

Word	Score
المرناقيّة	-777

Figure 5.13: An illustration of the multi-stage word recognition system. (Figure adapted from (Ahmad and Fink 2015a))

Data Preparation:

1. Training annotation:
   - The character or character shape based annotations for the training set are converted to core shape representation, utilizing the character shape to core shape mapping.
   - Another set of annotations are created for diacritics by utilizing the character shape to diacritics mapping. Characters having no diacritics are replaced by spaces.
2. Dictionary: Two separate dictionaries are created.
   - One of the dictionary contains word (or character for character recognition tasks) definitions in terms of core shapes.
   - The other dictionary contains word (or character for character recognition tasks) definitions in terms of diacritics.

Training:

(a) Core-Shape system

1. All the core-shape models are initialized using the initialization procedures as presented in Section 2.5.
2. The models are iteratively trained using the Baum-Welch training algorithm and utilizing the core-shape annotations.

(b) Diacritics system

1. The white-space model is defined as a single-state model.
2. All the diacritics models are initialized using the initialization procedures as presented in Section 2.5.
3. The models are iteratively trained using the embedded Baum-Welch training algorithm and utilizing the diacritics annotations.

Decoding:

(a) Character recognition

1. An input image is separated into a core-shape image and a diacritics image.
2. The core-shape system utilizes the features from the core-shape image to hypothesize a sequence of core shapes. It uses the character to core shape dictionary for decoding. (Optional) An N-best list of core-shape sequences is generated.
3. A character recognition network is constructed based on the hypothesized core-shape sequence.
4. The diacritics system utilizes the features from the diacritics image along with the constructed character network from step–3 to generate the final recognition hypothesis. It uses the character to diacritics dictionary for decoding.

(b) Word recognition

1. An input image is separated into a core-shape image and a diacritics image.
2. The core-shape system utilizes the features from the core-shape image to generate an N-best list of words. It uses the word-to-core-shape dictionary for decoding.
3. A word recognition network is constructed to include only the N-best hypothesized words.
4. The diacritics system utilizes the features from the diacritics image along with the constructed word network from step–3 to score the N-best words. It uses the words-to-diacritics dictionary for scoring.
5. Scores from the systems are added for respective words and the word having the best score is output as the final hypothesized word.

Figure 5.14: Key steps in the multi-stage Arabic text recognition consisting of the core-shapes system and the diacritics system.

5.4. Integrating sub-characters with core-and-diacritics system

In this section, we will present our third approach to modeling Arabic characters. It is a natural extension to the first two approaches, namely the sub-character approach and the core-shape and diacritics approach. The idea here was to combine both the approaches which can lead to even further reduction in the number of basic HMMs needed to model all the Arabic characters and their position-dependent shapes. We can remove the dots and other diacritics from the characters in the first step and, then, share the similar patterns in the core-shapes using the sub-character approach. If we take a look at the four Arabic characters and their 16 position-dependent shapes that were presented in Figure 5.1, we can see that, by removing the diacritics and using the sub-character approach, we can reconstruct the 16 position-dependent shapes using only five patterns (including the *Kashida* connector and the white-space model) in addition to two different diacritics (one-dot above and three-dots above). Using the word examples from Figure 5.3, we can notice that the words can be constructed using only 10 unique patterns (and four diacritics: one-dot above, three-dots above, one-dot below, and three-dots below) as illustrated in Figure 5.15.

5.4.1. Multi-stage recognition framework

A multi-stage text recognition framework using this modeling approach is quite similar to the one presented in Section 5.3.1. For setting up the training procedure, the training transcriptions are modified to represent the core shapes in terms of the sub-core shapes including the connector stroke. The sub-core-shape models are trained using the modified transcriptions along with the features from the core-shape images. There is no change in the training procedure for the diacritics system.

The recognition setup is also similar to the multi-stage approach, discussed before, except some small modifications which will be discussed here. For character recognition task, the stage which involves recognition of the core shapes needs a core shapes to sub-core shapes dictionary. Similarly, for word recognition tasks, the words' definitions in terms of sub-core shapes are needed. The rest of the decoding procedure is exactly the same. Key steps in the multi-stage Arabic text recognition involving the sub-core shapes and diacritics are presented in Figure 5.16.

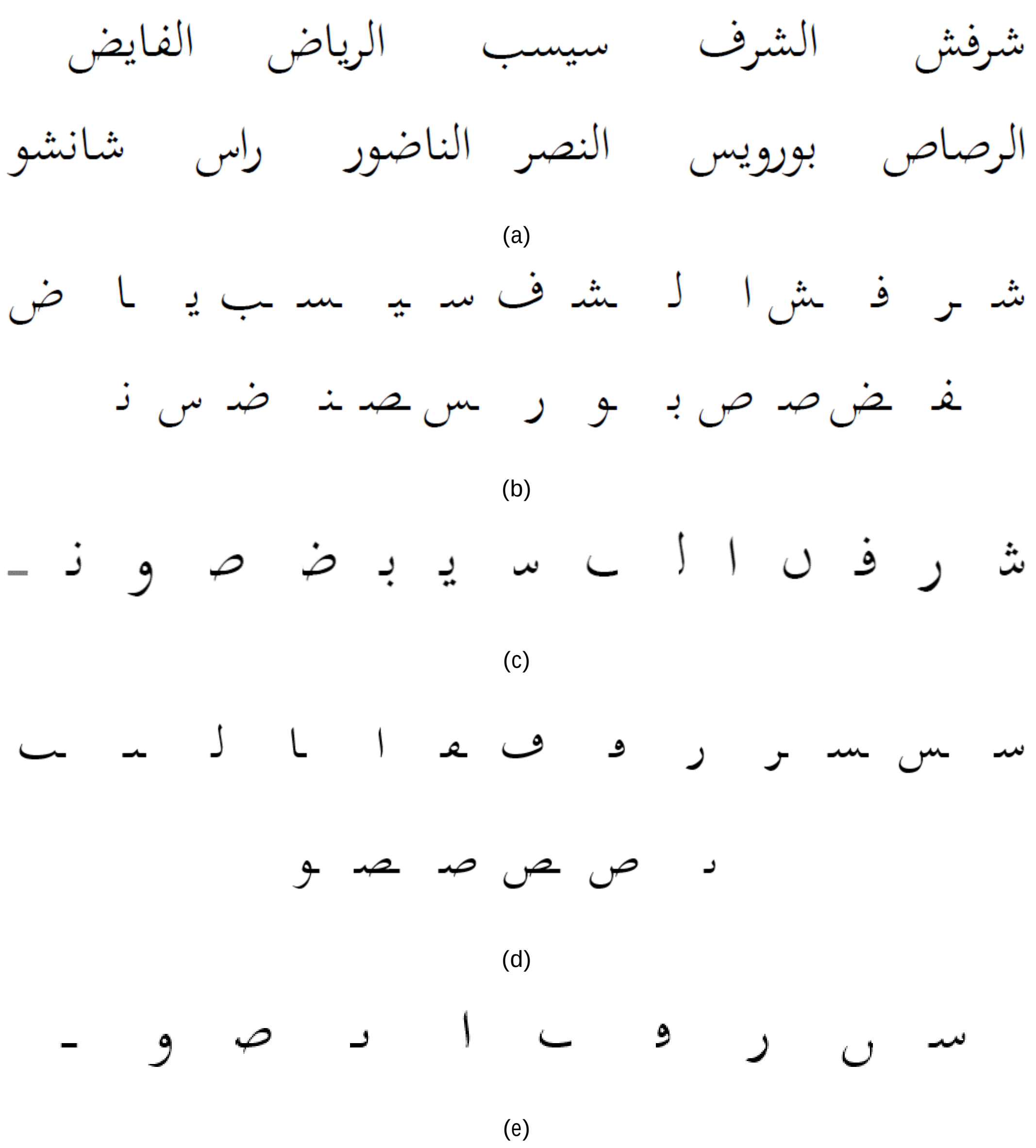

شرفش الشرف سيسب الرياض الفايض
الرصاص بورويس النصر الناضور راس شانشو
(a)
(b)
(c)
(d)
(e)

Figure 5.15: Sample words in (a) can be constructed using 28 character shapes listed in (b), or using 15 sub-character shapes as listed in (c), or using 19 core shapes (and four different diacritics) as listed in (d), or using 10 sub-core shapes (and four different diacritics) as listed in (e).

Data Preparation:

1. Training annotation:
   - The character shape based annotations for the training set are converted to sub-core shape representation, utilizing the character shape to sub-core shape mapping.
   - Another set of annotations are created for diacritics by utilizing the character shape to diacritics mapping. Characters having no diacritics are replaced by spaces.
2. Dictionary: Two separate dictionaries are created.
   - One of the dictionary contains word (or character for character recognition tasks) definitions in terms of sub-core shapes.

- The other dictionary contains word (or character for character recognition tasks) definitions in terms of diacritics.

Training:

(a) Core-Shape system

1. The connector and white-space models are defined as single state models.
2. All the sub-core shape models are initialized using the initialization procedures as presented in Section 2.5.
3. Transitions are added from non-emitting entry state to the non-emitting exit state for the connector and white-space models.
4. The models are iteratively trained using the embedded Baum-Welch training algorithm and utilizing the core-shape annotations.

(b) Diacritics system

1. The white-space model is defined as a single-state model.
2. All the diacritics models are initialized using the initialization procedures as presented in Section 2.5.
3. The models are iteratively trained using the embedded Baum-Welch training algorithm and utilizing the diacritics annotations.

Decoding:

(a) Character recognition

1. An input image is separated into a core-shape image and a diacritics image.
2. The sub-core shape system utilizes the features from the core-shape image to hypothesize a sequence of core shapes. It uses the character to sub-core shape dictionary for decoding. (Optional) An N-best list of core-shape sequences is generated.
3. A character recognition network is constructed based on the hypothesized core-shape sequence.
4. The diacritics system utilizes the features from the diacritics image along with the constructed character network from step–3 to generate the final recognition hypothesis. It uses the character-to-diacritics dictionary for decoding.

(b) Word recognition

1. An input image is separated into a core-shape image and a diacritics image.
2. The sub-core shape system utilizes the features from the core-shape image to generate an N-best list of words. It uses the word to sub-core shape dictionary for decoding.
3. A word recognition network is constructed to include only the N-best hypothesized words.
4. The diacritics system utilizes the features from the diacritics image along with the constructed word network from step–3 to score the N-best words. It uses the words to diacritics dictionary for scoring.
5. Scores from the systems are added for respective words and the word having the best score is output as the final hypothesized word.

Figure 5.16: Key steps in the multi-stage Arabic text recognition consisting of the sub-core shapes system and the diacritics system.

5.5. Other improvements related to model training

In this section, we will present some other strategies we investigated for improvement in model training. We investigated two major areas: one is related to multi-stream HMMs, while the other is related to contextual HMMs. We will present these two approaches in this section.

5.5.1. Multi-stream HMMs

Multi-stream HMMs (please refer to Section 2.7.3 for an introduction on it) has been used successfully in the domain of speech recognition. They are mainly popular in audio-visual speech recognition domain, where features from each source form a separate stream (cf., e.g., (Luettin, Potamianos, and Neti 2001; Manabe and Zhang 2004)). Kessentini et al. (Kessentini, Paquet, and Ben Hamadou 2010) investigated multi-stream HMMs for handwritten Arabic text recognition. Different combinations of four different features, computed from the text image, were trained as two-stream HMMs. The recognition results of two-stream HMM systems were better than the recognition results obtained when the respective features were fused or when results were combined from separate HMMs trained on the individual features.

In our case, we tried a slightly different approach to multi-stream HMMs. Instead of using completely different features as individual streams, we computed the horizontal derivative features for every feature frame and use the derivative features as a separate stream. Based on the experimental results (presented in Chapter 7), we observe that using multi-stream HMMs, as presented here, does improve the recognition performance. One possible justification for this improvement can be attributed to the fact that, as the features are split into two different streams, the dimension of each stream is essentially half the original feature dimension. This reduction in dimension can help in more robust output modeling at the mixtures level. Modeling low-dimension features is more robust than modeling high-dimension features due to the, relatively, less complex sub-feature space. Moreover, splitting features into multiple streams also enables the use of stream weights; thereby, allowing us to assign different importance to each feature. The issue of selecting optimal weights for the streams was addressed by experimenting with different weights and selecting the weight combinations that gives the best recognition performances on the development set. Using different stream weights for the two streams showed improvement in the text recognition performance as presented in Chapter 7. Figure 5.17 summarizes the multi-stream training procedure.

Multi-stream training:

1. The models are initialized and trained using the standards procedures in conjunction with the specific steps related to the modeling technique followed.
2. The mixtures are split into two streams such that the features computed from the image forms one stream and the derivative features form the second stream.
3. The stream weights are set for each stream by selecting the weight combination that performs the best—in terms of recognition rate—on the development set.

Figure 5.17: Steps for multi-stream HMMs training.

5.5.2. Contextual HMMs

In this section we will present our investigations related to contextual modeling for handwritten Arabic text recognition. Readers can refer to Section 2.7.1 for some general introduction on contextual modeling using HMMs. As indicated before, contextual modeling is an important aspect of any HMM-based speech recognition system; however, its use in the domain of text recognition is not that universal. In this section, we will, first, discuss the justification and the need for contextual modeling for handwritten Arabic text recognition. Next, we will discuss the issues related to contextual modeling for Arabic text recognition. This will be followed by studying the effect of sub-character modeling on contextual modeling and the role and benefit of the connector model. Finally, we will discuss class-based contextual modeling for Arabic text recognition.

5.5.2.1 The need of contextual modeling for handwritten Arabic text recognition

Contextual variations in Arabic text can be visualized and understood at multiple levels. As each character in Arabic can take different shapes based on its position, this is the first contextual level that need to be modelled. Figure 5.18 shows different Arabic words where the encircled glyphs in every row represent the same character. It can be clearly seen from the figure that the characters have significant variations due to their position in a word (please refer to Section 3.2 for more details). The most common way to accommodate these contextual variations between different character shapes is by treating each character shape as a separate model (please refer to Section 4.2). An alternative approach is to model a character, instead of character shape, as an HMM and use contextual HMM modeling to capture the shape-based variations as was presented by Prasad et al. for printed Arabic text recognition (Prasad et al. 2008).

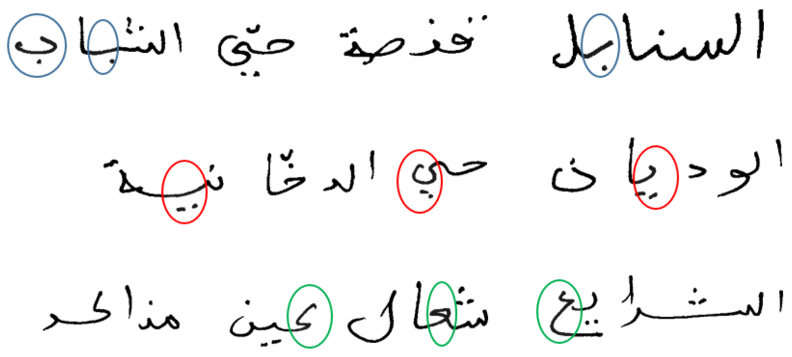


Figure 5.18: Handwritten Arabic texts illustrating the variation in characters' appearance. Glyphs encircled by a color in each row represent the same Arabic character. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

The second level of contextual variations is at the character-shape level. Even the character shapes show visual variations due to a number of reasons. As characters in Arabic script are connected to their neighboring characters in a word (a more correct term will be PAW instead of word as some characters do not connect to other characters in front of them), some stroke variations do occur when connecting a character shape to the next character shape in a word. Some variations are simply due to the different handwriting styles, but some variations seem to be due to a character's neighboring characters. A prominent example of this phenomenon is the occurrence of character pairs that are treated as special ligatures like *lām-alif* (لا). A solution to address this is by modeling these ligatures as separate models. But, the problem is that, some of these character pairs do not always appear in ligature form (please refer to Figure 3.9 and Section 3.5 for more details) and, thus, it is not always possible to model these character pairs as a special model. Another reason for variations at the character-shape level is a result of sliding window technique for feature extractions. Some characters partially overlap with other characters even though they might not, necessarily, be connected. These overlap get captured within the sliding window passing over a character and, as a result, effects the features computed for the character. Figure 5.19 illustrates contextual variation at character-shape level due to neighboring characters. Each row marks a character shape in a specific color to illustrate the variations in visual appearances due to its neighboring characters. In order to account for these variations, the most common approach is to model each character shape as a separate model and do contextual HMM modeling at the character-shape level, i.e., a contextual HMM represents a character shape in the context of its neighboring character shapes.

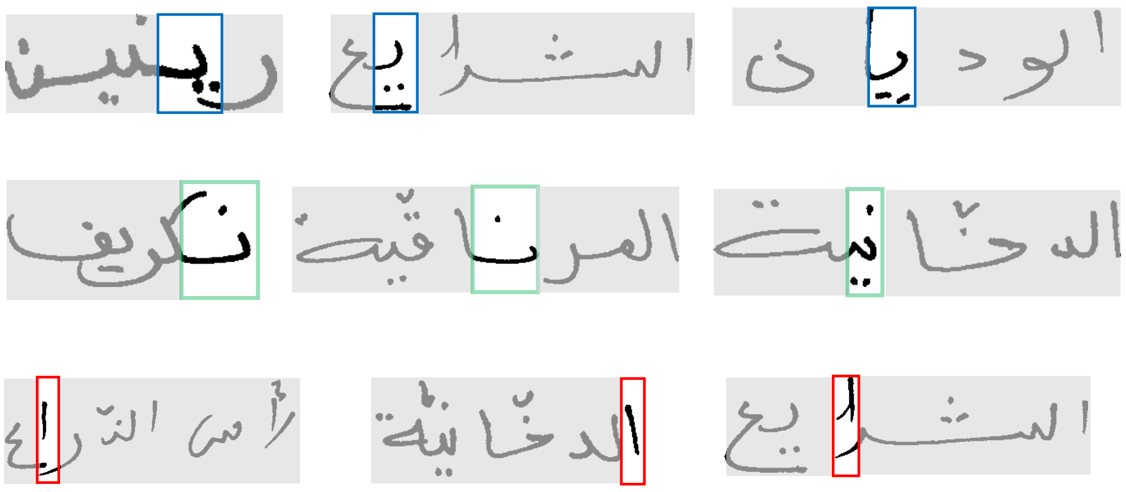


Figure 5.19: Figure illustrating the effect of neighboring characters on character shapes. Each row shows instances of a specific character shape (enclosed within a colored-edge rectangle) and the variations in its visual appearance due to the neighboring characters. (Handwritten text images source: IFN/ENIT (Pechwitz et al. 2002)).

5.5.2.2 The issues associated with contextual modeling

Because of the fact that variations at character-shape level, due to the neighboring character shapes, exists in handwritten Arabic texts, it becomes important to model them for better recognition performances. Contextual HMMs are, thus, a natural choice for Arabic text recognition. However, using tri-character (tri-character-shape in this case) HMMs for contextual modeling comes with their own issues, especially for Arabic script. The concern is related to the high number of contextual models that results from converting the mono-character-shape HMMs to tri-character-shape HMMs. Using character shapes as HMMs instead of character HMMs already led to a four-fold increase in the number of HMMs. Now, converting these character-shape HMMs into the contextual forms leads to a further increase in the number of HMMs. One can easily end up having thousands of HMMs. Having a huge number of HMMs leads to the problem of insufficient training data. Moreover, some low occurring character-shapes will have even lower number of its different contextual forms in the training data. This leads to inadequate model training. To alleviate this problem, some form of clustering is performed. The two most common approaches are the data-driven clustering and the decision-tree clustering (see Section 2.7.1 for more details). Both these techniques have been used for Arabic text recognition, as was presented in Section 4.2. However, model clustering is applied after training the contextual forms and, thus, if the training was not adequate, the clustering will not be optimal. Thus, although there are strong justifications for using contextual HMMs for Arabic text recognition, its potential has not been greatly achieved.

5.5.2.3 The effects of using sub-character HMMs and the connector model

In this section, we will present some of the observations regarding the positive effects of using the sub-character HMMs, along with the connector model, in relation to contextual modeling. Sub-character modeling, as presented in the previous sections, leads to a significant reduction in the number of HMMs in a recognition system, a direct impact of this is the large reduction of tri-character forms for these modeling units. As the number of contextual HMMs is relatively low, the problem of inadequate training is alleviated to some extent. However, the most significant effect on contextual modeling seems to be the result of using the connector model as proposed in Section 5.2.1. Use of the connector model leads to two major positive implications on the contextual modeling approach. Due to the fact that a connector appears between every two characters connected in a word, the most abundant tri-character models are, therefore, the different contextual forms of the connector model. The other models mostly have the connector and the space model as their neighbors; therefore, having fewer contextual forms. Thus, the tri-character model set is, comparatively, very small when using the sub-character modeling approach. Moreover, the variations in the characters' appearances due to different neighboring characters are mainly localized towards the periphery of the characters, and not around the core area. The periphery of the characters where two characters connect to each other is what the connector basically models. Thus, having different tri-character forms of the connector model and, at the same time, limiting the number of tri-character forms of other models, seems to be an effective approach. Experimental results related to contextual modeling, as presented in Chapter 7, supports our observations in this regard.

For all the contextual modeling experiments, we initialize the tri-character HMMs for a character by using the trained context-independent HMM for that character as an initial model. Next, we train the tri-character models by applying a number of iterations of the Baum-Welch training algorithm. After this, state level data-driven clustering is performed to tie similar states across different contextual forms of a character. State tying between different contextual forms is performed while preserving the state sequence, i.e., the corresponding states in the different contextual forms are tied if the distance between two states is within a threshold.

Training the contextual HMMs

1. The non-contextual models are initialized and trained using the standards procedures in conjunction with the specific steps related to the modeling technique followed.
2. Different tri-character* forms for each character* are generated using the training-set transcriptions.
3. Each tri-character model is initialized as a replica of its corresponding mono-character model which was trained in step–1.
4. The state-transition matrices of all the tri-character models of a character are tied.
5. The tri-character models are iteratively trained using the Baum-Welch training algorithm.
6. State level data-driven clustering is performed to tie similar states across different contextual forms of a character while preserving the state sequence. The thresholds are selected based on the recognition performance on the development set.
7. The state-tied tri-character models are, again, are iteratively trained using the Baum-Welch training algorithm.

* The word 'character' is used in a loose sense here and it can mean any modeling unit like character shape, sub-character, and core shape.

Figure 5.20: Key steps involved in training the contextual HMMs.

The distance *d(x,y)* between two states, *x* and *y*, is computed using the following equation as presented in (Young et al. 2002):

$$d(x,y) = -\frac{1}{M}\sum_{m=1}^{M} log\left[b_y(\mu_{xm})\right] + log\left[b_x(\mu_{ym})\right] \qquad (10)$$

where;

M is the number of mixture components*,*

μ_{xm} is the mean vector for the x^{th} mixture component of state m*, and*

$b_y(o)$ is the probability of generating observation *o* by state *y* which is given by Equation 1.

Figure 5.20 outlines the key steps involved in training the contextual models.

5.5.2.4 Class-based contextual modeling for Arabic text recognition

As we discussed above, one of the main problems when using contextual HMMs for Arabic text recognition is the inadequate training of the resulting high number of tri-character models. We also observed that, by using sub-character modeling approach and particularly the connector models, we can reduce the number of unique tri-characters. These observations led us to investigate class-based contextual modeling for Arabic text recognition. The core idea was to limit the number of unique tri-characters by not modeling every tri-character as a separate model, but, instead, by grouping the characters in the left and the right context into classes where characters in each class have similar effects on its neighboring characters.

The idea of class-based contextual modeling for handwritten text recognition is not new. A similar idea was presented by Fink and Plötz (Fink and Plötz 2007) for offline recognition of handwritten text in Latin script. The authors grouped the characters appearing in the left and the right context into six different categories: characters occupying core area, characters with ascenders, characters with descenders, characters with both ascenders and descenders, numerals, and the upper case characters. The groups were identical for both the left and the right context. Authors reported improvement in recognition results, as compared to the baseline context-independent system, when using this approach. At the same time, the normal approach to contextual HMMs resulted in poorer results when compared to the baseline system.

We grouped the Arabic characters into seven different classes for the left context, i.e., for character appearing after a given character, and four different classes for the right context, i.e., for the characters appearing before a given character. This grouping is subjective and was based on studying the characters' behavior in the context of other characters. Table 5.1 lists the character classes for the left context and Table 5.2 list the character classes for the right context. It should be noted that the number of classes as well as the specific characters in a given class is different for the two contexts. This is because the characters in Arabic have different influence on their neighboring characters. For example, the character *rā*′(ر) has a strong descender which affects the characters after it, i.e., to its left, but does not affect the characters before it, i.e., to its right. Figure 5.21 outlines the key steps involved in training the class-based contextual models and performing word recognition using them.

In Chapter 7, we will present the experiments we conducted and the results we obtained by employing the modeling and the training approaches presented in this chapter.

Table 5.1: List of character classes for the left context.

Left contexts	
Class	**Example character shapes**
Ascenders	ـا ـل ـلـ
Descenders	ج ح خ ع غ
Core	ـب ـبـ ـت ـتـ ـث ـثـ ـد ـر
Loop	ـص ـطـ ـفـ ـقـ ـمـ
Angular	ـجـ ـحـ ـخـ ـعـ ـغـ
kāf	ـكـ
Space	ا ب ت ن ل و د ر

Table 5.2: List of character classes for the right context.

Right contexts	
Class	**Example character shapes**
Ascenders	لـ ـلـ ـطـ ـظـ
Descenders	ـر ر و ـو د ـد
Core	بـ ـبـ تـ ـتـ ثـ ـثـ ـجـ ـحـ ـعـ
Space	ا ـا ـل ـب ـت ـث ـج
	ـح ـخ ـع ـغ ـص ـض

Data preparation:

1. Training annotation:
   - The training set annotations are converted into tri-character forms.
   - The tri-character annotations are modified such that the left and the right contexts are mapped to the respective classes.
   - A list of unique class-based tri-characters are generated from the modified annotations.
2. Dictionary:
   - Word are defined in terms of class-based tri-characters.

Model Training:

1. The non-contextual models are initialized and trained using the standards procedures in conjunction with the specific steps related to the modeling technique followed.
2. Each class-based tri-character model is initialized as a replica of its corresponding mono-character model which was trained in step–1.
3. The state-transition matrices of all the class-based tri-character models of a character are tied.
4. The class-based tri-character models are iteratively trained using the Baum-Welch training algorithm.
5. State level data-driven clustering is performed to tie similar states across different contextual forms of a character while preserving the state sequence. The thresholds are selected based on the recognition performance on the development set.
6. The state-tied tri-character models are, again, are iteratively trained using the Baum-Welch training algorithm.

Decoding:

1. The class-based tri-character models along with the modified dictionary are used for recognizing the words. The Viterbi algorithm is used for decoding.

Figure 5.21: Key steps involved in training the class-based contextual HMMs and text recognition utilizing the class-based contextual HMMs.

5.6. Summary

The standard approach, as reported in the literature, is to use position-dependent character shapes as basic modeling units for Arabic text recognition. This leads to text recognition systems with large models sets—typically around 100 models. Large number of models in a system implies the need for large amounts of data for robust training. In this chapter, three alternative modelling approaches were presented for Arabic text recognition. These alternative modeling approaches were proposed based on the observations that Arabic characters share common patterns between them and many characters differ from other characters only in diacritics. All the presented modeling approaches lead to significant reductions in the total number of models in the resulting systems as compared to the system using the standard character shapes as models. The systems using the presented modeling approaches are expected to perform more robustly when compared to the standard system, especially when little training data is available. Strategies to deal with situations where no handwritten training data is available are presented in the next chapter.

6 Handwritten Text Recognition in the Absence of Handwritten Training Set

In the last chapter, we presented the different modeling options we investigated for Arabic text recognition. In this chapter, we will present our investigations related to recognition of handwritten Arabic text when no handwritten training set is available. We will first present the motivations for our work. This is followed by techniques we investigated for machine printed Arabic text recognition in the presence of multiple font typefaces (referred to simply as font hereafter) and also recognition of machine printed text in a font which was not seen during training. The work related to machine printed text was a by-product of our efforts towards the original problem, i.e., on how to recognize handwritten text when no handwritten training data is available, as it can be regarded as a, relatively, simpler but a similar problem. Finally, we will present our approach to handwritten text recognition without handwritten training set.

6.1. Motivation for this work

Training a recognizer is one of the most important stages for any text recognition task. Availability of enough training samples of each class is very important to adequately train a recognizer. The classes can represent characters, character shapes, strokes, or other suitable representation like the ones discussed in the previous sections. To assist the research in the area, benchmark databases are developed to provide data for training and calibrating the recognizer and to evaluate a recognizer's performance on the evaluation sets (e.g., (Marti and Bunke 2003; Mahmoud et al. 2014; Pechwitz et al. 2002)). In order to ensure adequate training, huge amounts of data are collected and labelled. Data collection, labelling, and verification are labor-intensive and time-consuming activities. Researchers generally agree that quantity as well as quality of training data is important (cf., e.g., (Baird 2007; Varga and Bunke 2008)). Moreover, as the data collected in one environment is mostly not suitable for text recognition task in a different environment, there is a constant need to prepare new datasets for different text recognition scenarios. To alleviate the problem of manual data labelling and verification, some semi-supervised approaches to label the data have been investigated (e.g., (Richarz et al. 2014)), but the problem is far from being solved in this respect.

This issue has been identified by the researchers and, accordingly, some work has been published to deal with situations involving smaller training sets. The most notable approaches are related to text-image synthesis; where, training data is augmented by synthesizing text images from the original handwritten training set which was deemed not big-enough to adequately train the recognizer (e.g., (Moysset et al. 2014; Elarian et al. 2014)). Varga and Bunke (2008) presented perturbation models to synthesize text images from handwritten text images. A number of geometric distortions were applied in addition to thinning and thickening of pen strokes. The synthesized text line images augmented the training set. Experiments conducted for offline handwritten text recognition tasks showed that adding the synthetic data, to the original training set, led to improvements in recognition rates. Improvements were observed both when the original training set was small as well as large. Although, it was stated that it is easier to improve in situations where the training set is small. Miyao and Maruyama (2006) presented use of synthesized characters, in addition to the original samples, to improve the training of a Japanese Hiragana characters. Affine transformation was applied on strokes of the characters to synthesize additional characters. For Arabic text recognition, Elarian et al. (2015) presented two approaches to synthetically generate additional training data from a small set of handwritten Arabic text images. One of the approaches concatenate isolated character samples to form words. Some fitness criteria were proposed to select adequate samples for smooth concatenation. The second approach was based on connector (i.e., *Kashida*) modeling such that two adjacent characters can be joined by synthesizing the connector between them. The results from a handwritten Arabic word recognition task showed that the addition of synthetic data to the original training set can improve the text recognition results. For more details on text-image synthesis for improving text recognition performance, readers can refer to (Elarian et al. 2014).

Techniques based on semi-supervised learning where few labelled data is used to annotate the complete training set is another notable approach to deal with the problems of limited training data. Frinken et al. (2011) presented a 'co-training' approach where two separate systems were initialized using few labelled samples. The two systems were, then, iteratively trained such that a system annotates the unlabeled data which is supplied as training data to the other system after filtering the good results. An approach to train a recognizer using only unlabeled data was presented by Kozielski et al. (2014). The most challenging part was to initialize the system as no labeled data was available for training. The authors generated an initial transcription of the unlabeled data by using a language model and information related to the width of the word images. They also used some heuristics to train white-space models separate from the character models. An iterative approach was employed to fine tune the recognizer by, first, generating transcription for the training data and then, in-turn, training the recognizer on the hypothesized data in each iteration. The approach was effective based on word recognition experiments conducted on two separate datasets. The results were quite promising even though they were lower than the results of the systems trained on labeled training data.

According to the best of our knowledge, no work has been reported in the literature which deals with situations where no handwritten training set is available. This area of research has favorable implications as, in future, this can lead to greatly minimizing, if not completely removing, the need for handwritten training set. In this section, we will present our approach to handwritten Arabic text recognition where no handwritten training set is available. Our approach was based on the observation that Arabic is an inherently cursive script; therefore, it has a degree of visual similarity both in the handwritten as well as in the machine printed forms. Thus, to initialize the text recognition system (which is the most challenging part in such problems) we use computer generated machine printed text as the training data. The problem of recognizing handwritten text using a system trained on machine printed text is to a degree similar, even though much harder, to the problem of recognizing machine printed text in a font which was not seen during the training of an OCR system. Thus, to investigate this approach further, we first investigated text recognition of machine printed text in a font not seen by the recognizer during training. We will first present this work before returning back to the original problem of handwritten text recognition when no handwritten training data is available.

6.2. Machine printed text recognition

Researchers have addressed text recognition for unseen font using various approaches. The most common approach is to train the recognizer with samples from as many fonts as possible to address the variability during recognition (e.g., (Bazzi, Schwartz, and Makhoul 1999; Prasad et al. 2008; Bazzi et al. 1997; Khorsheed 2007)). This approach, normally, results in better overall recognition when compared with using a recognizer that was trained using texts from only a single font. However, the error rates are significantly higher than the error rates for text recognition for known fonts. Another approach for unseen font text recognition was proposed by Ait-Mohand et al. (Ait-Mohand, Paquet, and Ragot 2014). They proposed a supervised HMM adaptation technique in which both the model states as well as the model lengths were adapted. They demonstrated the effectiveness of their technique for mixed-font and unseen-font text recognition. However, the technique has two issues: the need for a few labeled text samples in the recognition font, i.e., the need for adaptation data, and the dependency of the technique on the assumption that all text line images to be recognized will belong to a single font.

To deal with the situation of unseen-font text recognition, we use a two-step font association based recognition. We train multiple mono-font text recognizers instead of training a recognizer on text mages from multiple fonts. Moreover, we propose a font identification module which can associate a text line image to the closest font. During recognition, the input text line image is, first, associated with a known font. As a second step, we will use the mono-font recognizer, which was trained on the associated font, to generate the recognition hypothesis. If there are multiple text lines images to be recognized using the identified recognizer, we perform unsupervised HMM adaptation during the recognition. Supervised HMM adaptations can be performed, instead, if some labelled samples are available for the input font. If the unseen font is much different than any of the trained fonts, the text line image to be recognized may not be associated to a single font with a high confidence. In such cases, a group of fonts (which is a subset of all the trained fonts) may be more representative of the input text line instead of a single font. A recognizer trained on the subset of the fonts maybe used instead of using a recognizer trained on a single font. Investigating this approach is a future work.

Based on the experimental results presented in Section B.2, this approach looks effective. Moreover, this approach enables the use of font specific aspects (if any) for feature extraction (for example, font specific features or parameters) and training (for example, different fonts can have different ligature models) which can further optimize the recognition performance. To train the font association module, one can use appropriate features and classifiers. In the present work, we present a set of simple and effective features for font identification that rely mainly on the projection profile of the text line image. These features were used with support vector machines (SVM) classifier, and the font identification results are very promising, as demonstrated in Section B.2. Below we describe our font identification features.

6.2.1. Features for font Identification

The features are extracted from height normalized (i.e., keeping the aspect ratio constant) text line images. Before introducing the features, we would like to introduce a function $p(i,j)$ that we will use frequently to define our features.

$$p(i,j) = \begin{cases} 1, & if\ row\ 'i'\ column\ 'j'\ of\ the\ image\ has\ ink\ pixel \\ 0, & otherwise \end{cases}$$

Moreover, we use the term 'h' to denote image height and 'w' to denote image width.

a. **Maximum ink projection (F₁)**: This feature calculates the maximum value of the ink projection of a text image. The value is normalized by the image width. The dimension of the feature is one.

$$F_1 = \frac{\max\limits_{i:=1\ to\ h}\left(\sum_{j=1}^{w} p(i,j)\right)}{w}$$

b. **Ratio of ink-pixels (F₂)**: It is the ratio of the number of ink-pixels in a row over the maximum ink projection. The dimension of the feature is the same as the normalized height of the image.

$$F_2(i) = \frac{\sum_{j=1}^{w} p(i,j)}{\max\limits_{i:=1\ to\ h}\left(\sum_{j=1}^{w} p(i,j)\right)}$$

c. **Percentage increase/decrease of pixel projection (F₃):** It is the percentage of increase/decrease of pixel projection in a given row as compared to the row immediately above it. The dimension of the features is one less than the normalized height of the image.

$$F_3(i) = \frac{\sum_{j=1}^{w} p(i,j) - \sum_{j=1}^{w} p(i-1,j)}{\sum_{j=1}^{w} p(i-1,j)} \qquad ; where\ 1 < i\ \leq h$$

d. **Compaction (F_4)**: It is defined as the ratio of the total number of ink-pixels in a text line image over the total area of the line image. The dimension of the feature is one.

$$F_4 = \frac{\sum_{i=1}^{h} \sum_{j=1}^{w} p(i,j)}{h \times w}$$

e. **Count of projections above average (F_5):** It is the count of the number of rows in the image whose ink-pixel count is above the average ink-pixel count of the image rows. The dimension of the feature is one.

$$F_5 = \sum_{i=1}^{h} a(i)\,; \quad where,$$

$$a(i) = \begin{cases} 1, & if \sum_{j=1}^{w} p(i,j) > \frac{\sum_{i=1}^{h} \sum_{j=1}^{w} p(i,j)}{h} \\ 0, & otherwise \end{cases}$$

We concatenate all the above defined features into one feature vector for a text line image.

6.2.2. Framework for machine printed text recognition

Based on the above discussions, we propose a framework for machine printed text recognition using the HMMs. In the first step, we train an HMM recognizer for individual fonts. We also train the font association module using the features described in Section 6.2.1 and an SVM classifier. For an input text line image that has to be recognized, we, first, associate the text image to the closest known font after extracting the font features and providing them to the font association module. Next, we extract features from the text line image for text recognition. If font-specific parameters for feature extraction (such as window width and overlap) exist, they can be employed during feature extraction. After feature extraction, we utilize the HMM recognizer (for the associated font) for decoding. If we expect a batch of text line images from a single font during decoding, we have the following two options: (i) If some labeled samples are available for the input text's font, we perform supervised HMM adaptation prior to decoding; otherwise, (ii) we perform unsupervised adaptation during decoding. If we expect random input images from a number of fonts, then we decode the text using the associated font's recognizer. Figure 6.1 illustrates the framework steps.

6.3. Approaches for handwritten text recognition

Now, we will focus back on the original problem—handwritten text recognition without the handwritten training set. As we mentioned before, initializing the recognizer is one of the most important and challenging issues in such situations. To initialize the recognition system at some reasonable level, we use computer generated machine printed text as training data and later perform unsupervised HMM adaptation during recognition. As Arabic script is cursive both in machine printed and handwritten forms, using computer generated machine printed text for training and adapting it for handwritten text recognition proves to be promising. In the following sub-sections, we will present our step-wise approach to deal with the scenario of handwritten text recognition in the absence of handwritten training data.

6.3.1. Training using computer generated text in single font

We generated Arabic text in a number of different fonts and trained separate recognition systems for each of the font. We were interested in knowing how effective a mono-font recognizer will be in recognizing handwritten Arabic texts. Due to the cursive nature of the Arabic script, machine printed texts do have a degree of visual resemblance to handwritten texts; although, the variability in handwritten texts is quite high. Moreover, we were also interested to investigate the effects of different font on the recognition performance. It will be useful to know if all the recognizers, each trained on specific-font texts, perform similar or the performance is different for different fonts. If the performance is different, can the visual complexity of a font give us some indications about its handwritten text recognition capability in the sense that more complex fonts perform better or vice-versa. Figure 6.2 shows samples of Arabic word images in one machine printed font and also samples of handwritten images of the same words. From the figure we can notice that the machine printed text (the right column) has some visual similarities to the handwritten texts (the left column); although, handwritten texts have more variability.

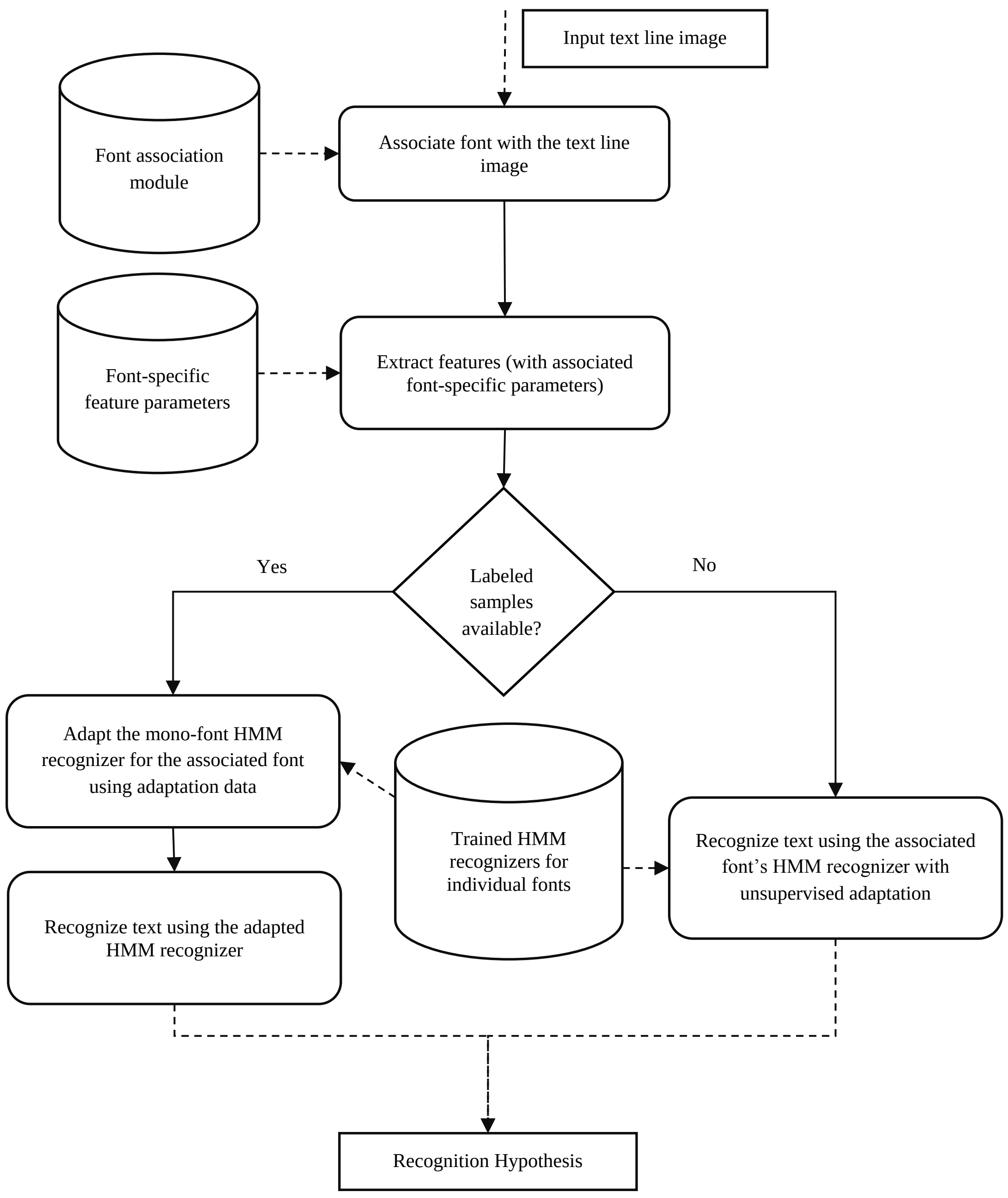

Input text line image
Font association module
Associate font with the text line image
Font-specific feature parameters
Extract features (with associated font-specific parameters)
Labeled samples available?
Yes
No
Adapt the mono-font HMM recognizer for the associated font using adaptation data
Trained HMM recognizers for individual fonts
Recognize text using the associated font's HMM recognizer with unsupervised adaptation
Recognize text using the adapted HMM recognizer
Recognition Hypothesis

Figure 6.1: Framework for machine printed text recognition. (Figure adapted from (Ahmad, Mahmoud, and Fink 2016))

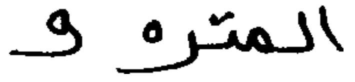
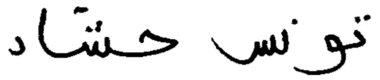
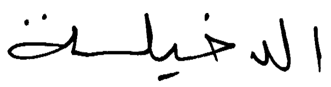
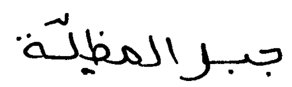
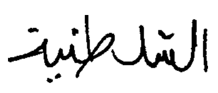
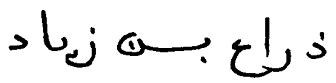
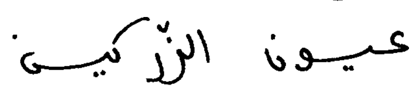
Samples handwritten word images	Computer generated images (*Naskh* font)
	المنزه 9
	تونس حشاد
	الدخيلة
	جبل املظيلة
	السلطنية
	ذراع بن زياد
	عيون الزركين

Figure 6.2: Samples of computer generated text images in one font (*Naskh*) along with samples from handwritten text images for the same texts. (Figure from (Ahmad and Fink 2015b))

6.3.2. Training using computer generated text in multiple fonts

In this step, we want to investigate the effects on handwritten text recognition when using a recognizer which is trained on multiple fonts, instead of just one font. Does the recognizer trained on text images from multiple fonts perform better than the recognizers trained on only a single font? There is a good reason to believe that the recognizer may perform better when trained on multiple fonts as this may, to some extent, enables it to model the handwriting variability better when compared to the recognizers trained on text from only one font. Figure 6.3 shows sample word images for two words in eight different fonts as well as sample handwritten word images for the same two words. We can observe from the figure that different fonts add to the variability in appearance which may help, to some extent, model the handwriting variability.

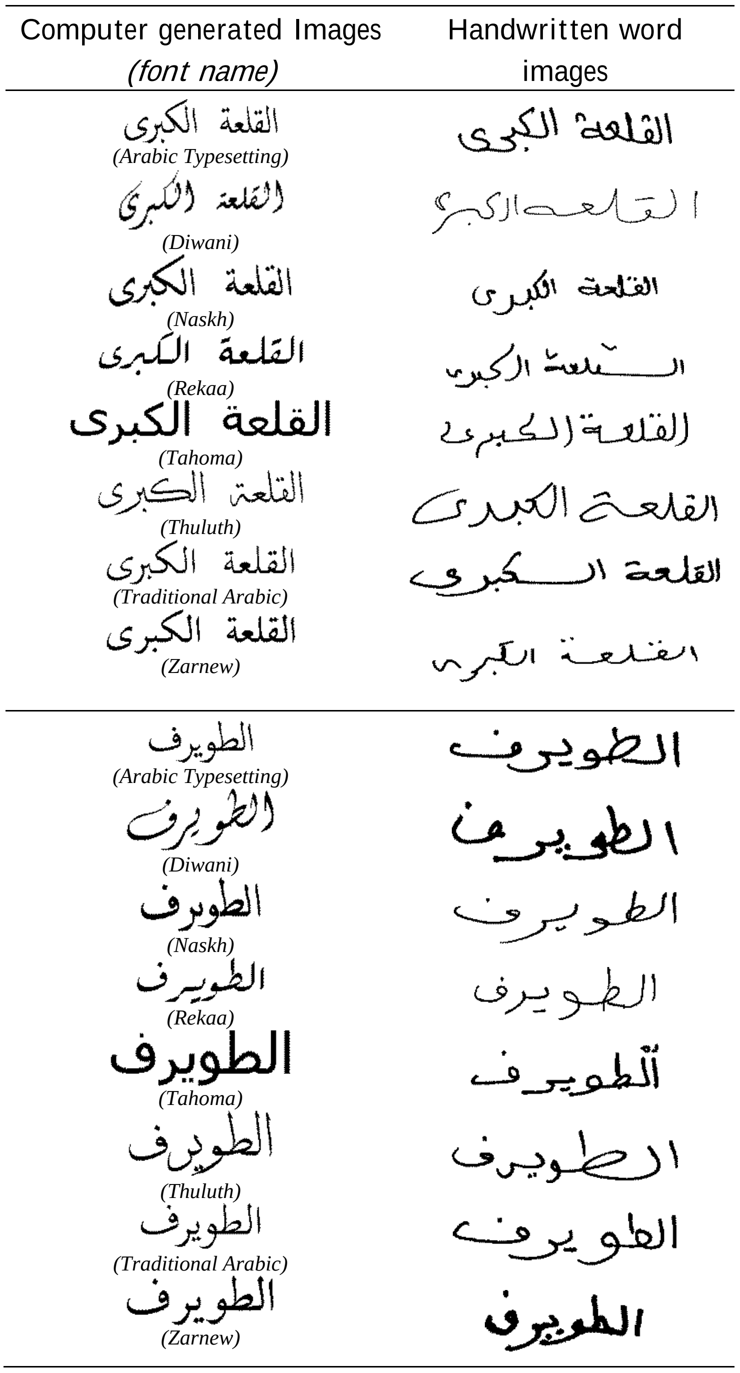


Figure 6.3: Samples of computer generated text images in different fonts along with samples from handwritten text images for the same texts. (Figure from (Ahmad and Fink 2015b)).

6.3.3. Performing unsupervised adaptation

Unsupervised HMM adaptation techniques recalibrate the trained parameters based on the new data they see during recognition (please refer to Section 2.7.2 for more details on HMM adaptation techniques). It has been used for adaptation of handwritten text recognizer for new writers (e.g., (Saleem et al. 2009)). HMM adaptations techniques were applied for adapting a multi-font text recognizer to a specific-font text recognition task in (Ait-Mohand, Paquet, and Ragot 2014). In the present work, we investigated the use of HMM adaptation to adapt a recognizer trained on printed text to handwritten text recognition task. As we do not use any labeled training data, we perform unsupervised HMM adaption during recognition.

6.3.4. Using test hypothesis as data for iterative training

In this approach, we generate recognition hypothesis for the handwritten test images using the system developed by the previous approach. Next, we use these recognition hypothesis to re-train the classifier. Once the handwritten text has been labeled 'reasonably' well, using this test data, in-turn, for training can prove to be an effective approach and may perform better than the previous approaches. In this case, the previous approaches can be regarded as initialization steps to start-up the recognizer. Clearly, for this approach to work, it will be important that the recognition hypothesis generated at this stage is reliable to, at least, some degree. Training on poorly hypothesized data can, in fact, do more harm than good and can even perform worse than the previous approaches as the classifier will be trained on huge amounts of wrongly labeled data (cf. (Baird 2007)). Thus we need to limit, if not completely remove, the mislabeled data from the correctly hypothesized data. To address this issue, we remove the bottom five percent of the hypothesized data based on the length-normalized scores. Figure 6.4 shows a typical graph for normalized recognition scores. The bottom five percent is marked to give an indication of the score range of the removed data. The bottom five percent does not necessarily mean wrongly hypothesized data; however, there is a high likelihood that many of the images were wrongly hypothesized. Another approach, which we have not yet investigated, is to use the recognition hypothesis on the test set together with the computer generated text as the training data. This will require a careful mix of the two sets so as not to over-train a classifier on one type of data.

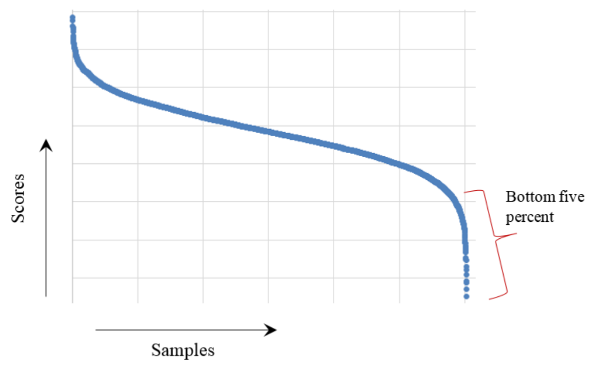


Figure 6.4: An example graph showing the normalized recognition scores (sorted) for the text images.

If the previous step works reasonably well, then we can improve the recognizer's performance by iteratively feeding the improved recognition hypothesis as training data (after removing the images having the worst length-normalized recognition scores) which in-turn can lead to better recognition. After certain iterations, the recognition performance may reach an improvement threshold and further iterations may not, necessarily, improve much. This improvement threshold can be judged by looking at the average log-probability of the frames during training. If there is no significant improvement in the average log-probability of the training feature frames, the iterative algorithm can be terminated at that step. Figure 6.5 shows the block diagram of the complete recognition process as proposed here.

6.4. Summary

A step-wise approach to system initialization and training in the absence of handwritten training data was presented in this chapter. As Arabic script is cursive both in the machine printed and the handwritten forms, there is a degree of visual similarity between the two forms. This observation led us to investigate the use of computer generated texts in multiple font typefaces as training data for system initialization. Further improvement in the system's performance can be made by adapting the system, trained on machine printed texts, on handwritten text recognition tasks. The adaptation was done in an unsupervised manner while performing text recognition. Finally, using the test hypothesis as training data in an iterative way has a potential to further improve the results. The three alternative modeling approaches presented in the previous chapter along with the approaches presented in this chapter to deal with situations where no handwritten training data is available are evaluated in the next chapter.

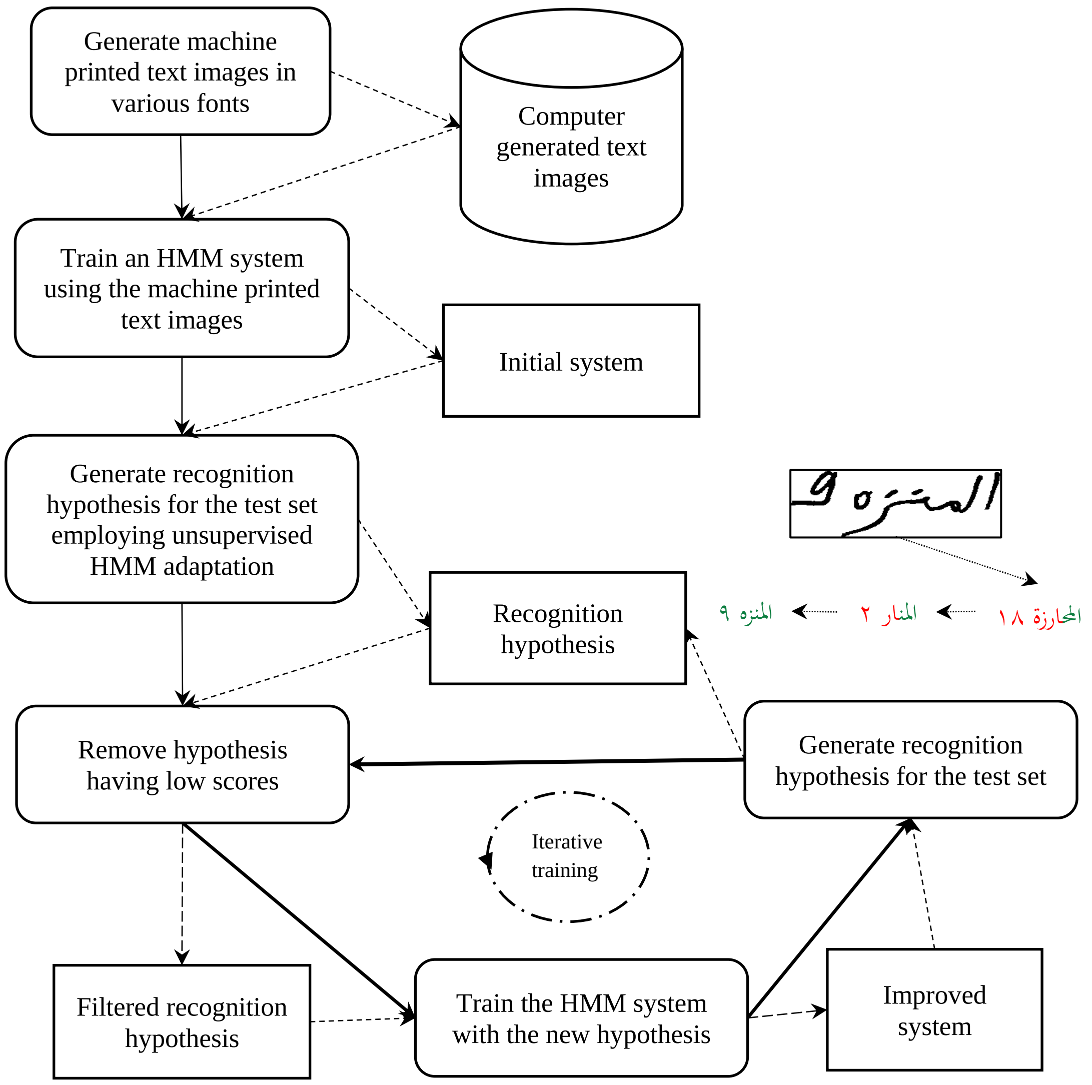

Generate machine printed text images in various fonts
Computer generated text images
Train an HMM system using the machine printed text images
Initial system
Generate recognition hypothesis for the test set employing unsupervised HMM adaptation
Recognition hypothesis
المنزه ٩
المنار ٢
المحارزة ١٨
Remove hypothesis having low scores
Generate recognition hypothesis for the test set
Iterative training
Filtered recognition hypothesis
Train the HMM system with the new hypothesis
Improved system

Figure 6.5: The process framework for Arabic text recognition without handwritten training set. (Handwritten text image source: IFN/ENIT (Pechwitz et al. 2002)).

7 Evaluations

In this chapter, we will present the evaluations for the different modeling and training options that were presented in Chapter 5 and Chapter 6. We will first present the text recognition databases we used for evaluations. Next, in Section 7.2, we will present a brief description of the text recognition tasks we performed. In Section 7.3, we present the evaluation measures we used. The details of the experiments conducted, the results obtained, and the discussions will be presented in Section 7.4. Finally, in Section 7.5 we present a summary of the experiments, the results, and the discussions. The experiments and the results related to our contributions to machine printed text recognition, as presented in Section 6.2, can be found in Appendix—B.

7.1. Databases for text recognition

We employed two handwritten Arabic text recognition databases for the evaluation. Below we present a brief description of the databases.

7.1.1. IFN/ENIT database of handwritten Arabic names

IFN/ENIT database is the most popular and the commonly used benchmark database for handwritten Arabic text recognition research (Pechwitz et al. 2002). The database consists of handwritten images of the names of Tunisian cities and towns divided into seven sets *a to f,* and *s.* The lexicon size is 937 names where each name has one or more words. Some names have two or more variations. The database originally consisted of 32,492 images divided into sets *a to e*. Later, set *f* and set *s*, consisting of 8671 and 1573 images respectively, were added. Set *s* is regarded as the most difficult set. One of the reasons being that it was collected in a different country and hence has different writing styles as compared with the other sets. Figure 7.1 shows some sample images from the IFN/ENIT database. IFN/ENIT database has been used in various text recognition competitions and the results are presented in the top conferences, related to the field, like ICDAR and ICFHR (e.g., (Märgner, Pechwitz, and Abed 2005; Märgner and Abed 2007; El Abed and Märgner 2010a; Märgner and Abed 2010; Märgner and Abed 2011)).

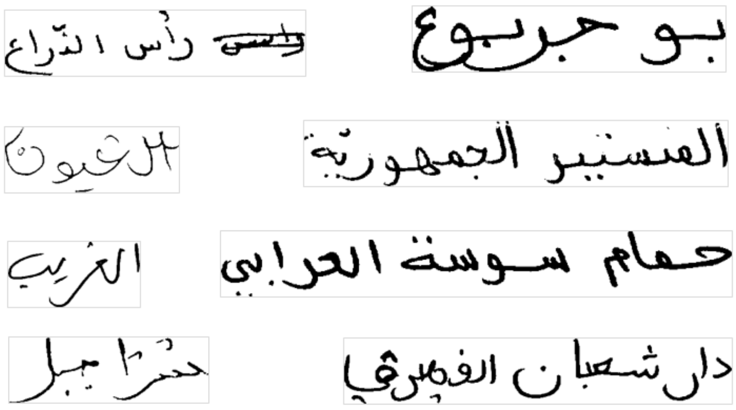


Figure 7.1: Sample text images from the IFN/ENIT database.

7.1.2. KHATT database of handwritten Arabic texts

KHATT database consists of unconstrained handwritten Arabic text images (Mahmoud et al. 2014; Mahmoud et al. 2012). A total of 1000 writers wrote the text where each writer wrote four paragraphs. Two of the paragraphs contain similar text written by all writers and are, therefore, not included in the text recognition experiments. Text line images were extracted from the paragraph images. The database is divided into three disjoint sets for training, development, and testing respectively. The train, development, and test sets currently contain 4808, 937, and 966 text line images respectively which were extracted from the paragraph images. The KHATT database is more challenging than the IFN/ENIT database as it contains unconstrained handwritten text. Figure 7.2 shows some sample text line images from the database.

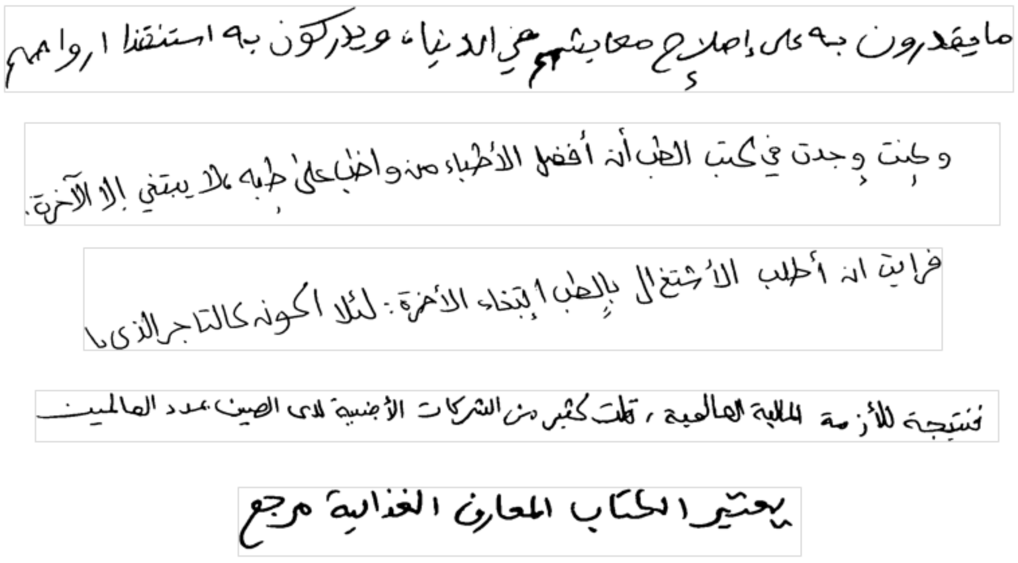


Figure 7.2: Sample text images from the KHATT database.

7.2. Text recognition tasks

In this section, we describe the different text recognition tasks we conducted using the different databases.

7.2.1. Word recognition

The first set of experiments we conducted are the word recognition tasks using the IFN/ENIT database. In a strict sense it may be termed as *name* recognition instead of *word* recognition as a name may consist of multiple words. We used the most common train–test configuration as reported in the literature including the various competitions held using the database. Apart from the common configurations, we also experimented under constrained training scenarios where we used smaller sets for training the recognizer. Unless otherwise stated, set *d* was used as the development set to optimize the parameters.

Word recognition on the IFN/ENIT database was also carried out when we investigated our proposed approach to perform handwritten text recognition without using handwritten training sets.

7.2.2. Character recognition

Apart from performing word recognition tasks using the IFN/ENIT database, we performed lexicon-free character recognition experiments—generating character hypothesis instead of word hypothesis using a system trained on text line images. Not using the lexicon, as well as the language models, makes the recognition tasks harder but it leads to more insights regarding the effects of the use of the different modeling options. Standard train–test configurations were used for experimentation.

We also performed character recognition experiments using the KHATT database. We used the same partitions of the database for training, development, and test as is provided in the KHATT database. Word recognition using the KHATT database was not experimented as it involves dealing with a number of issues related to the selection of lexicon size and appropriate n-grams which were not the focus of the present research.

7.3. Evaluation measures

In this section, we present the measures we used to evaluate the text recognition results depending on the text recognition tasks.

7.3.1. Word Error Rate (WER)

When performing word recognition task, we use Word Error Rate (WER) as the performance measure. WER is defined as follows:

$$WER\ (\%) = \frac{S + I + D}{N} \times 100$$

where;

S is the substitution error, i.e., the total number of words substituted,

I is the insertion error, i.e., the total number of words inserted,

D is the deletion error, i.e., the total number of words deleted, and

N is the total number of words in the evaluation set.

The number of words substituted, inserted, and deleted are counted after aligning the recognized word string against the transcription. The alignment is known as maximum substring matching problem (cf. (Huang et al. 2001) pages 419–421). In case of isolated word recognition, the equation simplifies to:

$$WER\ (\%) = \frac{S}{N} \times 100$$

7.3.2. Character Error Rate (CER)

For character recognition tasks, we report the results using Character Error Rate (CER) as the performance measure. Similar to WER, CER is defined as:

$$CER\ (\%) = \frac{S + I + D}{N} \times 100$$

where;

S is the substitution error, i.e., the total number of characters substituted,

I is the insertion error, i.e., the total number of characters inserted,

D is the deletion error, i.e., the total number of characters deleted, and

N is the total number of characters in the evaluation set.

The number of characters substituted, inserted, and deleted are counted after aligning the recognized character string against the transcription.

7.3.3. Significance interval of the results

In addition to reporting the results in terms of WERs or CERs, we also report the statistical significance of results when comparing the results from different systems on the same text recognition task. We use the statistical test for the difference of two proportions as presented in (Dietterich 1998) to report the significance interval of results. Thus, for a given text recognition task, we report the significance interval of the results at 95% confidence level.

7.4. Experimentations, results, and discussions

In this section, we will present the experiments we conducted, the results we obtained, and the resulting discussions. First we will present the word recognition experiments we conducted on the IFN/ENIT database. This will be followed by the character recognition experiments that were conducted using the IFN/ENIT database. Next, the character recognition experiments conducted on the KHATT database will be presented. Experiments related to handwritten text recognition without the use of handwritten training data is presented next. Finally, character recognition using machine printed databases will be presented.

7.4.1. Word recognition using the IFN/ENIT database

We will present the word recognition experiments we conducted using the IFN/ENIT database using the different modeling choices presented in Chapter 5.

7.4.1.1 Modeling choice: Character shapes

In this section, we will present the details of the recognition system that uses character shapes as models and the results we obtained using the system for the word recognition tasks. Our text recognizer is a system based on continuous HMMs. We use HTK tools (Young et al. 2002) to build our recognizer. Below, we will describe the key aspects of our system before discussing the experiments and the results.

The first step involved preprocessing of the text images. The text images were already binarized. The only preprocessing step we performed was the baseline correction of the text images. To correct the baseline, we, first, remove the small components from the text image. Next, we select the ink pixels from the lower contour of the text. The position of the selected ink pixels were, then, used to estimate a course-baseline. Next, the ink pixels which were far (more than one standard deviation) from the estimated course-baseline were removed from the selection. We then perform regression on these selected points to estimate the skew angle of the baseline. Finally, the image is rotated based on the estimated skew angle.

After correcting the baseline of the text images, we extract the features from them. We used the sliding window approach for feature extraction with the height of the window being the same as the image height and the width of the window was selected as 8 pixels. Consecutive windows overlap by 4 pixels such that the window shift is 4 pixels (i.e., 8 – 4 = 4 pixels). It should be noted that the sliding window runs from right to left across the text image as Arabic is written from right to left. We computed nine features from the sliding window frames running across the text line images. Apart from the nine features computed from the image-slices, we appended nine additional features for every window frame. These features are the derivative features. Thus, the dimension of the feature vector is 18. These feature were adapted from Wienecke et al. (Wienecke, Fink, and Sagerer 2005). Figure 7.3 lists down the features extracted from the text line images.

The IFN/ENIT database contains the transcriptions for the word image at character-shape level. In addition to the *lām-alif* ligature, few other character pairs are also represented by special ligatures. Moreover, some characters have optional *Shadda* diacritic over them and, as such, are represented by special models. We replaced some low occurring models (i.e., models whose counts were less than 30 in the training set) having *Shadda* by the models of the same character but without the *Shadda*. We ended up with a total of 157 models in our recognition system. All the models have Bakis topology. A dictionary consisting of definitions for the town and city names was constructed based on the annotations provided in the IFN/ENIT database. Some names have alternate variations and, as such, all the definitions were added to the dictionary.

Features for text recognition

Following are the list of features we computed from the text line images for text recognition tasks. These features were adapted from Wienecke et al. (Wienecke, Fink, and Sagerer 2005):

1. The average distance of the baseline to the upper contour of the ink pixels.
2. The average distance of the baseline to the lower contour of the ink pixels.
3. The average distance of the baseline to the center of gravity of the ink pixels.
4. The angle of the upper contour of the ink pixels with respect to the baseline.
5. The angle of the lower contour of the ink pixels with respect to the baseline.
6. The angle of the center of gravity of the ink pixels with respect to the baseline.
7. The average of the number of black-to-white transitions per column.
8. The percentage of ink pixels in a frame.
9. The average number of ink pixels between the upper and lower contours of the ink pixels.

Figure 7.3: The list of features extracted from the text line images for text recognition.

To initialize and train the models, we first performed model length adaptation (MLA). We used the concept of ‘slipping’ states as presented by Z. Jiang et al. (Z. Jiang et al. 2012) for MLA. All the models were initialized with large number of states (20 states with Bakis topology) and then the states having very low self-transition probabilities were removed from the model. After deciding the number of states for each model, we followed a multi-step approach to system initialization and training. In the first step, the models were initialized using the flat-start procedure (also known as uniform initialization) followed by a number of iterations of Baum-Welch training. The trained system was then used to perform forced alignment of the training data. Next, the information from forced alignment of the training samples was used to initialize individual HMMs using Viterbi initialization. This was followed by a number of iterations of Baum-Welch training. Finally, the trained system was used to decode the evaluation set using the Viterbi algorithm. It is important to note that the remaining system parameters like the number of mixtures per state and the parameters used by the HTK tools were optimally configured based on the recognition results on the development set. Set *d* was used as the development set for all the experiments involving the IFN/ENIT database.

For the first set of experiments, we used the standard *train–test* configurations. We used *abc–d* as the train–test set configuration for system development. Key system statistics related to the training the system using sets *a, b,* and *c* are presented in Table 7.1. It is interesting to note that 23 models have less than 100 samples and almost half the models have less than 200 samples.

Table 7.2 presents the summary of the results for standard train–test configurations. It can be seen from the table that our character shape system performs reasonably well when compared to the state-of-the-art systems evaluated on the IFN/ENIT database (cf. Table 4.2). The best results are reported for set *d* which is understandable given the fact that all the parameters were optimized using this evaluation set. The lowest rate (i.e., the highest WER) is reported on the evaluation set *s* which confirms that it is the most difficult set. These results will be treated as our *baseline results* and our character shape systems will be regarded as the *baseline systems* for the following experiments.

Table 7.1: Key statistics related to training the system with the IFN/ENIT training sets *a*, *b*, and *c* using character shapes as models.

Number of HMMs	157
Average number of samples per model	531
Median number of samples per model	186
Number of models having less than 100 samples	23
Number of models having less than 200 samples	80

Table 7.2: Summary of the word recognition results (in WERs) using character shapes as models on the IFN/ENIT database with standard train–test configurations.

Train–Test Configurations			
abc–d	***abcd–e***	***abcde–f***	***abcde–s***
4.01	8.47	9.87	17.74

Our next experiments were related to investigating the various contextual modeling approaches. We performed two sets of contextual modeling experiments. In the first set of experiments, we investigated the standard approach of contextual modeling along with the data-driven state clustering technique as was summarized in Figure 5.20. Again, the parameters like distance threshold (for state clustering) were calibrated based on the evaluations results on set *d*. In the second set of experiments, we investigated the class-based contextual modeling approach as proposed by us in Section 5.5.2.4 and summarized in Figure 5.21.

The evaluation results on the development set *d* for the three systems (including the results from the baseline system), in addition to key system statistics like total number of HMMs in the systems as well as the total number of states in the systems (using the training sets *a to c*), are presented in Table 7.3. The total number of states in a system is more indicative than the number of HMMs as many HMMs have their states tied to other HMMs in a system in the case of contextual modeling. The reduction in the number of HMMs after tying happens when all the corresponding states of two HMMs are tied together thereby merging the two logical HMMs into one physical HMM. It can be seen from the table that improvement in WER is reported for both the standard contextual HMM system as well as the class-based contextual HMM system. However, the improvement in WER for the class-based contextual system is higher as compared to the improvement obtained by using the standard contextual system. Moreover, there is a significant reduction in the number of HMMs and the total number of states in the class-based contextual HMM system as compared to the standard contextual HMM system. In fact, the total number of states in the class-based contextual

HMM system is only a fraction higher when compared to the non-contextual baseline system (only a 20% increase). This results in a compact contextual HMM system.

Table 7.4 presents the recognition results on all the training–test configurations. It can be seen from the table that improvements are observed in all the experiment configurations for the class-based contextual HMM systems when compared to the non-contextual HMM systems (i.e., the baseline systems) as well as when compared to the standard contextual HMM systems. For one experiment configuration (*abcd–e*) the standard contextual HMM system shows a lower performance than the baseline system whereas the class-based contextual HMM system still shows a small improvement over the baseline system. More importantly, both the contextual systems perform significantly better (significance interval of the error is ±1.53 at 95% confidence level) than the baseline system on the set *s* which is, relatively, a difficult evaluation set. Again, the class-based contextual system performs better than the plain contextual system on set *s*.

Table 7.3: Comparison of number of HMM models, total number of states, and the WERs for the different contextual HMM-based systems using the IFN/ENIT database with training sets *a to c* and evaluation on set *d*.

System Description	*Number of HMMs*	*Total Number of States*	*WER*
Character-shape HMM system (baseline)	157	1534	4.01
Contextual HMM system using the standard approach	4575 (3212 after tying)	38512 (2767 after tying)	3.86
Class-based contextual HMM system	626 (359 after tying)	5775 (1845 after tying)	3.37

Table 7.4: Summary of the results (in WERs) related to Contextual HMM modeling approaches with character shapes as models.

System Description	*WERs*			
	Train–Test Configurations			
	abc–d	*abcd–e*	*abcde–f*	*abcde–s*
Character-shape HMM system(baseline)	4.01	8.47	9.87	17.74
Contextual HMM system using the standard approach	3.86	8.81	9.81	15.89
Class-based contextual HMM system	**3.37**	**8.27**	**9.66**	**15.38**

Our final set of experiments was related to investigating the systems' performance on constrained training environments. Specifically, we were interested to see how the system will perform when only few training samples are available. Only a subset of the complete training set was used to train the system. We performed four different sets of experiments under constrained training settings by using 250, 500, 1000, and 2000 text line images respectively from set *a* of the database to train the system. It should be noted that under constrained training, we did not perform MLA as well as forced alignment based initialization. Thus, only uniform initialization in conjunction with Baum-Welch training was performed with every model having the same number of states. The number of states and other system parameters were optimized based on the evaluation results on set *d* of the database.

Table 7.5 presents the text recognition results on the four different evaluation sets under different constrained training setups. We can observe from the table that high WERs are reported for all the evaluation sets when using only 250 text line images for training. The worst results are reported for set *s*. The poor results are understandable given that very few training samples were used. The results improve as we keep increasing the number of training samples with the best results reported when using the 2000 text line images from set *a* as the training data. These results will help us compare the other modeling approaches when used under constrained training setups.

Table 7.5: Summary of the results (in WERs) using the character shapes as models under constrained training setups.

The Training Set Size	*Evaluation Sets*			
	d	*e*	*f*	*s*
250 text images	50.22	52.83	54.26	63.45
500 text images	31.54	33.90	32.67	47.23
1000 text images	20.03	26.07	25.74	39.48
2000 text images	14.30	20.90	22.64	35.73
Complete training set	4.01	8.47	9.87	17.74

7.4.1.2 Modeling choice: Sub-characters

In this section we will present the experiments we conducted and the results we obtained when using the sub-characters, as presented in Section 5.2, as our modeling units. The details of setting up the text recognition system—the preprocessing steps, feature extraction, and setting up the continuous HMM-based system—are same as the ones followed for setting up the text recognition system using character shapes as models as was presented in the previous section. The difference was in the choice of modeling units (sub-characters as HMMs in the present case), and the related training procedure it involves, as was presented in Section 5.2.

For the first set of experiments, we used the standard train–test configurations. We used *abc–d* as the train–test set configuration for system development. Key system statistics related to training the system using the training sets *a, b,* and *c* are presented in Table 7.6. The sub-character representation uses 97 HMMs to model all the characters and their shape variations. This by itself is a great improvement in terms of system compactness and efficiency. It can perform relatively well under constrained training environments as we shall see shortly. It is interesting to note that only two models have less than 100 samples and only nine models have less than 200 samples. Moreover, the median number of samples per model is more than twice the number of samples per model in the case of the character-shape system.

Table 7.7 presents the summary of the results for standard train–test configurations. It can be seen from the table that our sub-character system performs quite well when compared to the state-of-the-art systems evaluated on the IFN/ENIT database (cf. Table 4.2). The improvements in WERs are significant when compared to the results obtained when using character shapes as models (significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets *d*, *e*, *f*, and *s* respectively at 95% confidence level). In fact, the results for the sub-character systems are better than, or at least as good as, the results obtained for contextual HMMs with character shapes as modeling units (cf. Table 7.4). The best result is still reported for set *d* and the result for set *s* being the lowest.

Table 7.6: Key statistics related to training the system with the IFN/ENIT training sets *a* to *c* using sub-characters as models.

Number of HMMs	97
Average number of samples per model	1734
Median number of samples per model	492
Number of models having less than 100 samples	2
Number of models having less than 200 samples	9

Table 7.7: Summary of the word recognition results (in WERs) using sub-characters as models on the IFN/ENIT database with standard train–test configurations.

Train–Test Configurations			
abc–d	***abcd–e***	***abcde–f***	***abcde–s***
3.64	8.06	8.64	15.77

Our next set of experiments was related to investigating the use of multi-stream HMMs as presented in Section 5.5.1. The details of setting up the multi-stream HMM system was summarized in Figure 5.17. Each stream has a dimension of 9 as we split the features into two streams with the computed features forming one stream and the derivative forming the second stream. The weights for the two streams (stream–1: 0.2; stream–2: 1.0) were empirically calibrated based on the recognizer's performance on the development set *d*. The third row of Table 7.8 shows the results obtained using multi-stream HMMs. We can see that significant improvements are reported with multi-stream HMMs.

Next, we experimented with the contextual sub-character HMMs as presented in Section 5.5.2.3. The last row of Table 7.8 shows the experimental results when using contextual sub-character HMMs. From the results we can notice a relative reduction between 11% and 19% in the error rates when using contextual HMMs. This improvement is higher than the improvements achieved when using contextual HMMs with systems having character shapes as models. Thus, the experiment results corroborates our understanding that the use of sub-character models along with the *connector* model helps in contextual modeling for Arabic text recognition.

Table 7.8: Summary of the word recognition results (in WERs) related to multi-stream HMMs and contextual HMMs with sub-characters as models.

System Description	*WERs*[*]			
	Train–Test Configurations			
	abc–d	*abcd–e*	*abcde–f*	*abcde–s*
Character-shape HMM system (baseline)	4.01	8.47	9.87	17.74
Sub-character HMM system	3.64	8.06	8.64	15.77
Sub-character HMM system + multi-stream HMMs	2.97	6.66	7.22	15.00
Contextual sub-character HMM system + multi-stream HMMs	**2.44**	**5.55**	**6.40**	**12.14**

[*]Significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets *d*, *e*, *f*, and *s* respectively at 95% confidence level.

Our final set of experiments was related to investigating the systems' performance on constrained training environments. As in the case of character shapes as models, we performed four different sets of experiments under constrained training settings by using 250, 500, 1000, and 2000 text line images respectively from set *a* to train the systems. We did not perform MLA as well as forced alignment based initialization. Thus, only uniform initialization in conjunction with Baum-Welch training was performed with every model having the same number of states. The number of states and other system parameters were optimized based on the evaluation results on set *d* of the database.

Table 7.9 presents the text recognition results on the four different evaluation sets under different constrained training setups. Results are reported in terms of WERs. We can observe from the table that high WERs are reported for all the evaluation sets when using only 250 text line images for training. But it is important to note that, although the results are low when compared to using the entire training set, the results are significantly better than the results reported when using character shapes as models (cf. Table 7.5). The recognition results improve as we keep increasing the number of training samples with the best results reported when using the 2000 text line images from set *a* as the training data, which is still significantly lower than the results obtained when using the complete training sets. In all the experiments conducted under constrained training setups, the results for sub-characters models are better than the results obtained when using character shapes as models. This confirms that using sub-characters as models leads to a system which is more robust and performs significantly better under constrained training environments.

Table 7.9: Summary of the results (in WERs) using the sub-characters models under constrained training setups.

The Training Set Size	*Evaluation Set*			
	d	*e*	*f*	*s*
250 text images (from set a)	32.44	34.48	35.75	48.57
500 text images (from set a)	20.52	23.17	22.78	37.70
1000 text images (from set a)	16.08	20.74	21.19	36.81
2000 text images (from set a)	11.82	19.39	19.86	35.22
Complete training set	3.64	8.06	8.64	15.77

7.4.1.3 Modeling choice: Core shapes and diacritics with multi-stage recognition

In this section, we will present the experiments we conducted and the results we obtained using the multi-stage text recognition framework as presented in Section 5.3. The overall details of setting up the core shape recognition system—the preprocessing steps, feature extraction, and setting up the continuous HMM-based system—are same as the ones followed for setting up the text recognition system using character shapes as models as was presented in Section 7.4.1.1. The difference was in the choice of modeling units (core shapes as HMMs for the core-shape system), and the related training procedure it involves, as was presented in Section 5.25.3. Moreover, an extra preprocessing step was involved to separate the core shapes from the diacritics using the algorithm presented in Figure 5.11.

For the diacritics system, there were some differences in the system setup. Some of the features extracted from the diacritics images were different from the features listed in Figure 7.3. We did not use the three orientation features as the diacritics are too small and, most of the time, are dots. Instead, we replaced these three features by two other features—the number of components and the average size of components in an image strip. These two features seem to be more suitable as the main idea was to distinguish the number of dots. Thus, we computed a total of eight features from the diacritics image and appended the derivative features to it leading to feature vectors of 16 dimension. Figure 7.4 lists down the features extracted from the diacritics images. Moreover, we did not perform MLA as well as forced alignment based initialization for the diacritics system. Thus, only uniform initialization in conjunction with Baum-Welch training was performed with every model having the same number of states.

Features computed from diacritics images for recognition

Following are the list of features we computed from the text line images for text recognition tasks:

1. The average distance of the baseline to the upper contour of the ink pixels.
2. The average distance of the baseline to the lower contour of the ink pixels.
3. The average distance of the baseline to the center of gravity of the ink pixels.
4. The number of connected components in a frame.
5. The average size of connected components in a frame.
6. The average of the number of black-to-white transitions per column.
7. The percentage of ink pixels in a frame.
8. The average number of ink pixels between the upper and lower contours of the ink pixels.

Figure 7.4: The list of features extracted from the diacritics images for recognition.

For the first set of experiments, we used the standard train–test configurations. We used *abc–d* as the train–test set configuration for system development. Key system statistics related to training the system using the training sets *a, b,* and *c* are presented in Table 7.10. The core-shape representation uses only 71 HMMs to model all the characters and their shape variations which is less than half the number of models needed in the character shape system. In fact, the core-shape system has even fewer models as compared to the sub-character system. This helps in constrained training environments especially when very few training samples are available. Only one model has less than 100 samples and three models have less than 200 samples.

Table 7.11 presents the summary of the results for standard train–test configurations. It can be seen from the table that the multi-stage system performs quite well when compared to the state-of-the-art systems evaluated on the IFN/ENIT database (cf. Table 4.2). The improvements in WER are significant (significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets d, e, f, and s respectively at 95% confidence level) when compared to the results obtained when using character shapes as models. Although, the results from the sub-character systems are better than the multi-stage recognition system for most of the train–test configurations. The best result is still reported for set *d* and the result for set *s* being the lowest.

Table 7.10: Key statistics related to training the system with the IFN/ENIT training sets *a* to *c* on the multi-stage recognition framework using core shapes (in addition to diacritics) as models.

Number of HMMs	71 (core-shape system), 17 (diacritics system)
Average number of samples per model	2290
Median number of samples per model	972
Number of models having less than 100 samples	1
Number of models having less than 200 samples	3

Table 7.11: Summary of the experimental results on the IFN/ENIT database with multi-stage recognition framework using core shapes and diacritics as models with standard train–test configurations.

Train–Test Configurations			
abc–d	*abcd–e*	*abcde–f*	*abcde–s*
3.30	7.81	9.33	16.15

Our next set of experiments was related to investigating the use of multi-stream HMMs as presented in Section 5.5.1 and is similar to the one followed when using sub-characters as models. The weights for the two streams (stream–1: 0.2; stream–2: 1.0) were empirically calibrated based the recognizer's performance on the development set *d*. The third row of Table 7.12 shows the results obtained using multi-stream HMMs. We can see that significant improvements are reported with multi-stream HMMs. Next, we experimented with the contextual HMMs as presented in Figure 5.20. The last row of Table 7.12 shows the experimental results when using contextual HMMs. No significant change in results were observed when using contextual core-shape HMMs. In fact, two of the configurations report a drop in recognition rates when using contextual HMMs.

Our final set of experiments was related to investigating the systems' performance on constrained training environments. As with the case of character shapes as models, we performed four different sets of experiments under constrained training settings by using 250, 500, 1000, and 2000 text line images from set *a* to train the system respectively. Table 7.13 presents the text recognition results on the four different evaluation sets under different training setups. Results are reported in terms of WERs. We can observe from the table that, although the results are low as compared to using the entire training sets, the results are significantly better than the results reported when using character shapes as models (cf. Table 7.5).

Table 7.12: Summary of the word recognition results (in WERs) related to multi-stream HMMs and contextual HMMs with multi-stage recognition framework using core shapes and diacritics as models.

System Description	*WERs*[*]			
	Train–Test Configurations			
	abc–d	*abcd–e*	*abcde–f*	*abcde–s*
Character-shape HMM system (baseline)	4.01	8.47	9.87	17.74
Multi stage HMM system with core shape and diacritic models	3.30	7.81	9.33	16.15
Multi stage HMM system with core shape and diacritic models + multi-stream HMMs	2.63	6.27	7.54	14.49
Contextual Multi stage HMM system with core shape and diacritic models + multi-stream HMMs	2.46	6.27	7.74	14.88

[*]Significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets d, e, f, and s respectively at 95% confidence level.

Table 7.13: Summary of the results (in WERs) with multi-stage recognition framework using core shapes and diacritics as models under constrained training setups.

The Training Set Size	*Evaluation Set*			
	d	*e*	*f*	*s*
250 text images	30.91	36.78	38.38	48.63
500 text images	20.91	27.90	27.40	39.54
1000 text images	15.29	21.71	23.71	37.57
2000 text images	13.70	19.84	21.53	35.22
Complete training sets	3.30	7.81	9.33	16.15

7.4.1.4 Modeling choice: Sub-core shapes and diacritics with multi-stage recognition

In this section, we will present the experiments we conducted and the results we obtained using the multi-stage text recognition framework with sub-core shapes and diacritics as models as presented in Section 5.4. The overall details of setting up the sub-core shape recognition system as well as the diacritics system—the preprocessing steps, feature extraction, and setting up the continuous HMM-based system—is same as the ones followed for setting up the multi-stage system using core shapes and diacritics as models as was presented in Section 7.4.1.3. The difference was in the choice of modeling units (sub-core shapes as HMMs instead of core shapes as HMMs), and the related training procedure it involves, as was presented in Section 5.4.1.

For the first set of experiments, we used the standard train–test configurations. We used *abc–d* as the train–test set configuration for system development. Key system statistics related to the training sets *a, b,* and *c* are presented in Table 7.14. The sub-core-shape representation uses only 43 HMMs to model all the characters and their shape variations which is almost a fourth of the number of models needed in the character shape system. In fact, it even has less than half the number of models as compared to the sub-character system. Thus, the system is quite compact and is expected to perform robustly. It also helps in constrained training environments especially when very few training samples are available. Only one model has less than 100 samples and three models have less than 200 samples.

Table 7.14: Key statistics related to training the system with the IFN/ENIT training sets *a* to *c* on the multi-stage recognition framework using sub-core shapes and diacritics as models.

Number of HMMs	43 (core-shape system), 17 (diacritics system)
Average number of samples per model	4066
Median number of samples per model	1713
Number of models having less than 100 samples	1
Number of models having less than 200 samples	3

In Table 7.15, we present the summary of the results for standard train–test configurations. It can be seen from the table that the multi-stage system, with sub-core shapes as models, performs quite well when compared to the state-of-the-art systems evaluated on the IFN/ENIT database (cf. Table 4.2). The improvements in WER are significant (significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets d, e, f, and s respectively at 95% confidence level.) when compared to the results obtained when using character shapes as models. Although, the results from the sub-character systems are better than the multi-stage recognition system for most of the train–test configurations. Moreover, the results are similar to the results obtained when using multi-stage system with core shapes as models but with significantly fewer models in the system. The best result is still reported for set *d* and the result for set *s* being the lowest.

Table 7.15: Summary of the experimental results on the IFN/ENIT database with multi-stage recognition framework using sub-core shapes and diacritics as models with standard train–test configurations.

Train–Test Configuration			
abc–d	***abcd–e***	***abcde–f***	***abcde–s***
3.62	7.31	9.08	16.40

Our next set of experiments was related to investigating the use of multi-stream HMMs as presented in Section 5.5.1. The weights for the two streams (stream–1: 0.2; stream–2: 1.0) were empirically calibrated based the recognizer's performance on the development set *d*. The third row of Table 7.16 shows the results obtained using multi-stream HMMs. We can see that significant improvements are reported with multi-stream HMMs. Next, we experimented with the contextual sub-core HMMs which has similar properties to the contextual sub-character HMMs. The last row of Table 7.16 shows the experimental results when using contextual sub-core HMMs. As with the case of sub-character HMMs, significant improvements in results are observed; thereby, confirming the benefits of using the sub-character modeling approach in addition to the use of the connector model.

Table 7.16: Summary of the word recognition results (in WERs) related to multi-stream HMMs and contextual HMMs with the multi-stage recognition framework using sub-core shapes and diacritics as models.

System Description	*WERs[*]*			
	Train–Test Configuration			
	abc–d	***abcd–e***	***abcde–f***	***abcde–s***
Character-shape HMM system (baseline)	4.01	8.47	9.87	17.74
Multi stage HMM system with core shape and diacritic models	3.62	7.31	9.08	16.40
Multi stage HMM system with sub-core shape and diacritic models + multi-stream HMMs	2.91	6.36	7.68	14.94
Contextual Multi stage HMM system with sub-core shape and diacritic models + multi-stream HMMs	**2.29**	**5.24**	**6.68**	**13.48**

[*]Significance interval of the errors are ±0.38, ±0.57, ±0.51, and ±1.53 for evaluation sets *d*, *e*, *f*, and *s* respectively at 95% confidence level.

Our final set of experiments was related to investigating the systems' performance on constrained training environments. As with the case of character shapes as models, we performed four different sets of experiments under constrained training settings by using 250, 500, 1000, and 2000 text line images from set *a* to train the system respectively. Table 7.17 presents the text recognition results on the four different evaluation sets under different constrained training setups. Results are reported in terms of WERs. We can observe from the table that, although the results are low as compared to using the entire training sets, the results are significantly better than the results reported when using character shapes as models (cf. Table 7.5). In fact, for very few training data (i.e., the first row), the results are even significantly better than the sub-character system. This may be attributed to the fact that under very few training samples, having the smallest model set has a major impact in systems' performance.

Table 7.17: Summary of the results (in WERs) with multi-stage recognition framework using sub-core shapes and diacritics as models under constrained training setups.

The Training Set Size	*Evaluation Set*			
	d	*e*	*f*	*s*
250 text images	21.10	28.28	29.93	41.32
500 text images	17.83	26.26	26.47	37.76
1000 text images	15.71	22.56	24.10	37.64
2000 text images	13.62	20.16	22.30	35.28
Complete training sets	3.62	7.31	9.08	16.40

7.4.2. Character recognition using the IFN/ENIT database

In this section, we will present the character recognition experiments we conducted using the IFN/ENIT database using the four different modeling choices presented in Chapter 5. It should be noted that the corresponding systems are the same systems that were used in word recognition tasks as presented in the previous section. The only difference is that we perform plain character recognition instead of word recognition. There was no use of dictionary and language models.

Table 7.18 summarizes the character recognition results for the standard train–test configurations using the systems with the four modeling options. The first row of the table presents the results using character shapes as modeling units. We can notice that the character error rates are quite high as compared to the word error rates reported for the same system (cf. Table 7.2). The main reason behind this is the fact that no lexicon or language models are used and hence the recognition is done purely on the basis of appearance modeling. Thus, the results are more revealing regarding the effects of modeling choice on the text recognition performance. The use of lexicon or language models have an effect of masking the impact of modeling choices.

The second row of the table presents the results when using sub-character as modeling units. We can see that the improvements using the sub-character HMMs are significantly higher that the results using character shapes as HMMs (Significance interval of the errors are ±0.35, ±0.38, ±0.32, and ±0.75 for evaluation sets d, e, f, and s respectively at 95% confidence level). The reduction in error rates is proportionately much higher than the corresponding reductions observed in the word recognition tasks. Rows three and four of the table presents the results when using the multi-stage recognition with core shapes and diacritics and when using the multi-stage recognition with sub-core shapes and diacritics respectively. Again, we can observe that the reduction in error rates is much more significant when compared to the reductions in the word recognition tasks.

Table 7.18: Summary of the character recognition results (in CERs) using the IFN/ENIT database.

System Description	*CERs*[*]			
	Train–Test Configuration			
	abc–d	*abcd–e*	*abcde–f*	*abcde–s*
Character-shape HMM system (baseline)	45.12	51.98	47.20	54.21
Sub-character system	35.95	44.63	38.26	47.91
Multi stage system with core shapes and diacritics as models	37.85	46.35	39.71	48.19
Multi stage HMM system with sub-core shapes and diacritics models	36.84	43.41	40.27	49.44

[*]Significance interval of the errors are ±0.35, ±0.38, ±0.32, and ±0.75 for evaluation sets d, e, f, and s respectively at 95% confidence level.

7.4.3. Character recognition using the KHATT database

In this section, we will present the character recognition experiments we conducted using the KHATT database.

The first step involved preprocessing of the text line images. As KHATT database consists of unconstrained handwritten text, we needed to perform adequate preprocessing before the feature extraction step. Baseline and slant correction was performed based on the technique presented in (Mahmoud et al. 2014). Finally, the core text of the image was normalized before the feature were extracted from them. Same features as the one presented in Figure 7.3 were computed from the text line images. The sliding window with the width of 4 pixels and the overlap of 2 pixels were used for feature extraction. The initialization and training procedures are similar to the ones presented in Section 7.2.1. Thus, a multi-step initialization and training was performed which included MLA, a uniform initialization followed by alignment based initialization. All the parameters were optimally calibrated based on the systems' performance on the development set. We use the same training–development–test partition as provided in the database.

Accordingly, we used 4808 images for training, 937 images for development, and 966 images for the test.

For the character-shape system, there are a total of 157 models in the KHATT database that represent the various character shapes in Arabic in addition to the numerals and some punctuation marks. For the sub-character system, we ended up with 97 models representing the same text. Table 7.19 presents the character recognition results (in CERs) when using character shape as models (first row) and when using sub-characters as models (second row). Results for both the development set and the test set are presented. From the table we can see that the sub-character system outperforms the character-shape system (significance interval of the errors are ±0.37 for the development and the test sets at 95% confidence level). In addition, we also performed contextual sub-character modeling where contexts—only within a character—were modeled in terms of sub-characters. The third row of the table summarizes the results for contextual modeling experiments. Finally, we performed multi-stream HMMs based training and the results are presented as the last row in the table. We can see that improvements were observed in all the proposed modeling and training approaches besides the fact that significantly lower number of models are used by the sub-character system. The results are worse than the results we obtained for the character recognition tasks on the IFN/ENIT database which confirms that the KHATT database, in general, is more difficult and challenging than the IFN/ENIT database. However, the results are comparable to the results we obtained on set *s* of the IFN/ENIT database.

Table 7.19: Summary of the character recognition results (in CERs) using the KHATT database.

System Description	*CERs*[*]	
	Development Set	*Test Set*
Character-shape HMM system (baseline)	52.10	51.09
Sub-character system	51.52	50.69
Sub-character system + contextual modelling (within character)	50.72	50.07
Sub-character system + contextual modelling (within character) + multi-stream HMMs	49.49	49.93

[*]Significance interval of the errors are ±0.37 for the development and the test sets at 95% confidence level.

7.4.4. Word recognition without handwritten training sets

In this section, we will present the experiments we conducted related to handwritten text recognition in the absence of handwritten training sets. We first present the experiments on using a recognizer trained on computer generated texts in a single font. This is followed by experiments using a recognizer trained on computer generated text on multiple fonts. Next, we present the text recognition using unsupervised HMM adaptation. Finally, we present experiments related to the use of recognition hypothesis on the test set as training data. Our task is offline Arabic handwritten word recognition using the IFN/ENIT database (Pechwitz et al. 2002).

For the experimentation, we use the Arabic sub-character model based HMM recognizer as presented in Section 5.2 as it seems to be the most robust and effective recognition system (especially under constrained training environments) based on the experimental results presented in Section 7.4.1. However, it is important to note that only uniform initialization (flat-start) on the training set was performed. Moreover, no MLA was performed either.

In the first set of experiments, we use computer generated text from individual fonts to train our recognizer. To generate text, we use the IFN/ENIT lexicon with all its variations. Using the IFN/ENIT lexicon was not a prerequisite, as the only thing we needed was to have some training samples to train the different HMM models. Since our recognition task was on the IFN/ENIT database, we generated text using its lexicon. We generated 1929 images for each font corresponding to 1929 entries in the dictionary, i.e., we generated one sample per entry for eight different fonts. Samples of computer generated text in different fonts along with handwritten text images from the IFN/ENIT database for the same word were shown in Figure 6.3. We trained eight different recognizers, each trained on text from only one font. Once the recognizer was trained, we evaluated the recognizer by recognizing word images from set *d* of the IFN/ENIT database. The evaluation results are shown in Table 7.20. The results are shown in terms of WERs. From the results shown in the table we have following observations: Although the results were not entirely disappointing, in general the recognition rates were very low for most of the fonts, which is understandable. The character glyphs for computer generated texts are very regular with only one fixed pattern. It is very difficult for a recognizer to train the models which can cope with the huge variations found in human handwriting. Nevertheless, some fonts did relatively well; the recognizer trained on the *Naskh* font was able to achieve 73.08% WER, i.e., it was successful in recognizing approximately one-fourth of the total word images from set *d*. Another interesting observation was that, although the recognizer trained on visually simple font like *Tahoma* did worst, the recognizers trained on very complex fonts like *Rekaa* and *Diwani* did poorly as well. Thus, just having a look at the visual complexity of a font is not enough to predict its capabilities for effective training. Moreover, not all the fonts have similar behavior.

Table 7.20: Text recognition results on set *d* of the IFN/ENIT database using recognizers trained on machine printed texts from single fonts.

Font Name	WER (%)
Arabic Typesetting	88.75
Diwani	89.99
Naskh	**73.08**
Rekaa	92.72
Tahoma	95.69
Thuluth	82.33
Traditional Arabic	87.13
Zarnew	81.25

Our next goal was to study the recognizer's performance when trained with image samples from multiple fonts. In this experiment we train our recognizer with computer generated word images from all the eight fonts together. Thus, a total of 15,432 (1929 × 8) word images are used for training. Once the recognizer was trained, we evaluated the recognizer by recognizing word images from set *d* of the IFN/ENIT database. The evaluation results are shown in Table 7.21 (first row). It can be seen from the table that a significant improvement in recognition rate is achieved when we trained the recognizer on multiple fonts. Thus, the variability observed in the training samples due to the different fonts helps, to some extent, to model the variability in human handwriting in the case of Arabic script. A part of the improvement is also due to an eight-fold increase in the training data. To understand the contribution of multiple fonts alone, we carried one more experiment where we randomly selected only 1929 word images in the eight fonts for training the recognizer and evaluated the recognizer on set *d*. The recognizer was able to achieve 46.55% WER which explains that most of the improvement was indeed due to the use of multiple fonts.

Our next experiment was to use the recognizer trained on multiple fonts and perform unsupervised HMM adaptation during recognition. We used MLLR for parameter tuning (please refer to Section 2.7.2 for more details on HMM adaptation). We experimented with different number of regression classes. The evaluation results on IFN/ENIT set *d* is presented in Table 7.21 (second row). It can be seen from the table that significant improvements are achieved using unsupervised HMM adaptation. In the best configuration using 48 regression classes, it leads to improvement in recognition rate by 9.12%, i.e., a reduction in error by one-fourth approximately.

Table 7.21: Text recognition results (in WERs) on set *d* of the IFN/ENIT database using recognizers trained on machine printed texts on multiple typefaces and using unsupervised adaptation.

System	WER (%)
All fonts together	38.65
All fonts together + Unsupervised adaptation	29.53

Our next set of experiments was related to the idea of using the recognition hypothesis on the test set as training data for the recognizer. To start, we use the recognition hypothesis from the previous step (i.e., multi-fonts training and unsupervised adaptation during recognition) and use it to generate labels at the character level for each word image of the test set by forced alignment technique. An interesting aspect to investigate was to compare the results of the recognizer trained on computer generated text on multiple fonts with the recognizer trained on handwritten text images, but with imperfect labeling (as close to 30% of the word images were wrongly hypothesized). To limit the mislabeled data, we remove the bottom five percent of hypothesized data based on the length-normalized score (please refer to Section 6.3.4 for more details). After training the recognizer with the hypothesized set *d* of the IFN/ENIT database, we perform recognition on the same set. The evaluation results are presented in Table 7.22 (first row). It can be seen from the table that the results are significantly better as compared to the results from the previous approaches. As an extension to this experiment, we use this improved hypothesis to re-label the test set and use it to train our recognizer. After retraining our recognizer using the improved hypothesis for a few more iterations (until the average length-normalized scores for the hypothesis converges), we evaluate it on the same set. The results are presented in the second row of Table 7.22. We can see from the table that there is a small, but significant, improvement in the recognition rate. In our final set of experiments we use the multi-stream HMMs as was used in our previous experiments. Use of multi-stream HMMs led to a further small, but significant, improvement in the recognition rate as can be seen from the third row of Table 7.22.

Once we validated our approaches using the set *d* of the IFN/ENIT database, we replicated our experiments on sets *e*, *f*, and *s* of the database without changing the system parameters, i.e., our single font, multiple fonts, and adaptation systems were exactly the same as the ones used to evaluate set *d*. The only difference was the use of hypothesized data for the corresponding sets as training data. The summary of all the experiments are presented in Table 7.23. From the table we can see that the results, although below the state-of-the-art (cf. Table 4.2), are very promising considering that no handwritten data was used for training. The area of research seems exciting and needs further investigation. It has huge implications as this may greatly reduce, if not completely eliminate, the need for creating the handwritten training sets and its manual transcription which are very laborious and time consuming tasks.

Table 7.22: Text recognition results (in WER) on set d of the IFN/ENIT database using hypothesized test set for training.

System	WER (%)
Hypothesized data of the test set used for training	12.84
Hypothesised test-data used for training after five iterations	9.77
Hypothesised test-data used for training after five iterations + multi-stream HMMs	*8.39*

Table 7.23: Summary of the results (in WERs) for handwritten text recognition on the IFN/ENIT database without using handwritten training data.

System	*The Recognition System— Training Data*	*Evaluation Set*			
		d	*e*	*f*	*s*
1	Best individual font (*Naskh*)	73.08	77.90	75.9	72.61
2	Text images from all fonts together	38.65	44.16	44.86	48.06
3	Text images from all fonts together + unsupervised adaptation	29.53	33.47	39.07	45.26
4	Test set hypothesised using system 3 used as training data	12.84	15.57	18.56	33.4
5	Five iterations of Test set hypothesised using system 4 used as training data	9.77	12.35	15.22	29.26
6	System 5 + Multi-stream HMMs	**8.39**	**10.39**	**13.42**	**26.89**

7.5. Summary

In this section, we will present the main summary of our experiments and results and some comparisons to the state-of-the-art. For the handwritten Arabic text recognition, we experimented with four different modeling approaches—one of them being the standard modeling approach as reported in the literature and the remaining three were proposed by us in this work. Figure 7.5 presents a comparison of the number of models in the resulting systems using the four modeling approaches respectively. It can be see that our proposed approaches lead to significant reduction in the number of models defined in the systems which leads to a compact and robust system with reduced model sets. The result is that the systems are trained more robustly with the same amount of available training data.

Figure 7.6 shows the key training statistics on the IFN/ENIT training sets *a* to *c*. Again, we observe that our modeling approaches lead to higher mean and median number of training samples per model using the same amount of training data. Thus, the presented modeling approaches allow for, relatively, better training of each model and, as a result, better training for each character. This fact was shown to be even more important under constrained training environments where only a few training samples are available for training.

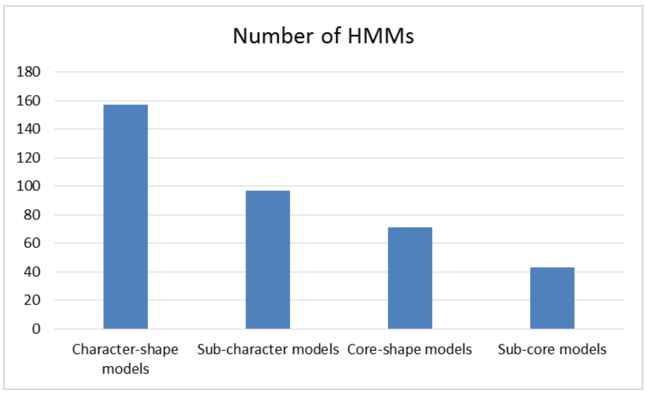


Figure 7.5: A comparison of the number of models in systems using the four modeling approaches. The numbers are based on the experiments conducted on the IFN/ENIT database.

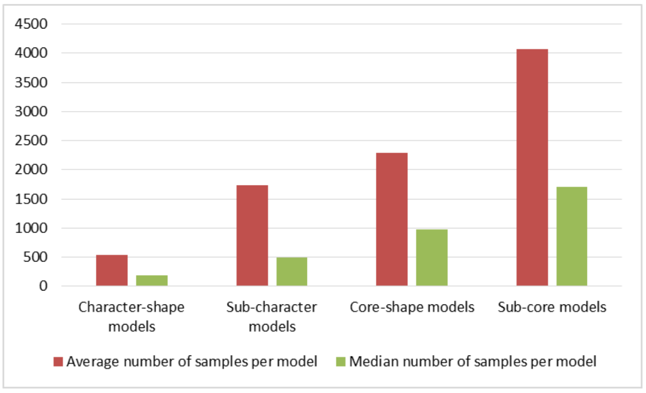
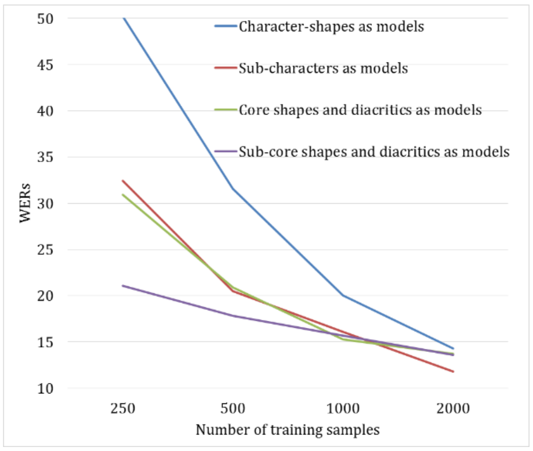
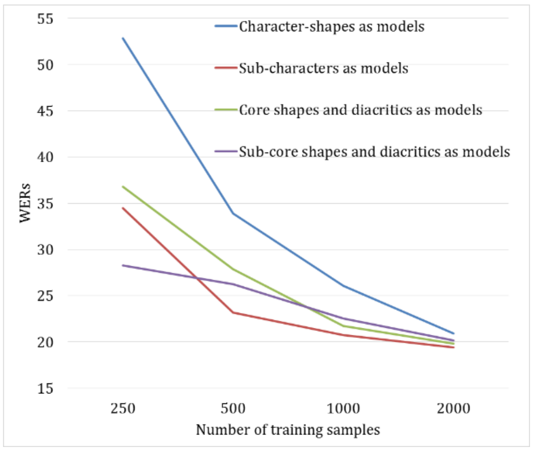


Figure 7.6: Key statistics related to training the system with the IFN/ENIT training sets *a* to *c* using the four modeling approaches.

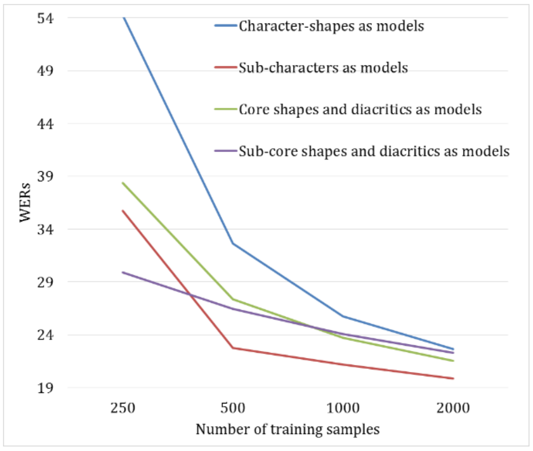
Figure 7.7 shows the performance of the four different systems on various evaluation sets from the IFN/ENIT database under constrained training environments. All the systems perform better than the standard system that uses character shapes as models. The performance of the sub-core shape system is the best when very few training data is available, but, in general, the sub-character systems perform better under most of the training configurations. One of the possible reasons for this could be the fact that the benefits of excessive sharing in sub-core shape systems, especially when enough training data is available, are balanced by losses due to the diacritics removal procedure which does not work perfect all the time. Nevertheless, the results are still significantly better when compared to the character shape systems in addition to the fact that the model-set size for the sub-core systems is only one-fourth the model-set size for the character-shape systems. Moreover, when employing contextual sub-character modeling and contextual sub-core modeling, the performance improvements are even larger.



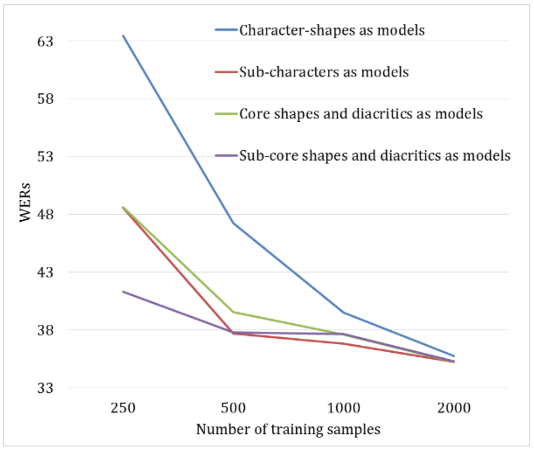
(a) Evaluation set: *d*



(b) Evaluation set: *e*



(c) Evaluation set: *f*



(d) Evaluation set: *s*

Figure 7.7: Performance of the systems with different modeling options under constrained training environments.

Table 7.24 presents a comparison of the results from the state-of-the-art systems evaluated on the IFN/ENIT database with the best results obtained by our systems using the three presented modeling approaches. We can see from the table that the overall results from our presented systems are among the best compared to the other state-of-the-art systems. In fact, our systems outperforms the best reported systems in the literature on the evaluation set *e* and set *f* of the database. We report the second best result on set *s* of the database which is regarded as the most difficult set in the database. Moreover, the best performing system on set *s*, presented by Stahlberg and Vogel (2015), uses our proposed special space and connector models for modeling. This further validates the effectiveness of our presented techniques. Set *d* of the database is used to calibrate the system and hence the systems have a tendency to over-fit on this evaluation set. The systems that are reporting the best results on set *d* are not necessarily performing very high on other evaluation sets.

Table 7.24: Comparison with other state-of-the-art systems evaluted on the IFN/ENIT database.

Systems	*WERs*			
	Train–Test Configurations			
	abc–d	*abcd–e*	*abcde–f*	*abcde–s*
Graves 2012 (Graves 2012)	-	-	6.63	18.94
Azeem and Ahmed 2013 (Azeem and Ahmed 2013)	2.3	6.56	6.9	15.2
Giménez et al. 2014 (Giménez et al. 2014)	4.7	6.1	7.80	15.38
Abandah et al. 2014 (Abandah, Jamour, and Qaralleh 2014)	**1.04**	6.54	7.54	15.20
Hamdani et al. 2014 (Märgner and Abed 2011; Hamdani et al. 2014)	-	-	7.80	15.45
Stahlberg and Vogel 2015 (Stahlberg and Vogel 2015)	2.4	6.1	6.8	**11.5**
Present Works				
Sub-character HMM system	2.44	5.55	**6.40**	12.14
Multi-stage HMM system with core shape and diacritic models	2.46	6.27	7.74	14.88
Multi-stage HMM system with sub-core shape and diacritic models	2.29	**5.24**	6.68	13.48

As for the character recognition experiments on the IFN/ENIT database, we did not find any reported work on the literature that does the same task except one system which was presented by Jiang et al. (2015). Table 7.25 compares the character recognition results of our systems with the only system presented in the literature. Again, we can observe that our systems clearly outperforms the other system. In fact, a similar system, presented by the same authors in (Z. Jiang et al. 2012), was evaluated for word recognition tasks and performed reasonably well as compared to the state-of-the-art. Unfortunately, we cannot find any system presented in the literature that performs character recognition task on the KHATT database. We did not perform word recognition task on the KHATT database as it involves using the word n-grams and dealing with OOV words which was not in the scope of our present research.

Table 7.25: Comparison with other systems evaluted on the IFN/ENIT database for the charcater recognition tasks.

Systems	*CERs*			
	Train—Test Configurations			
	abc—d	*abcd—e*	*abcde—f*	*abcde—s*
Jiang et al. (Z. Jiang et al. 2015)	50.97	58.38	-	-
Present Works				
Sub-character HMM system	**35.95**	44.63	**38.26**	**47.91**
Multi-stage HMM system with core shape and diacritic models	37.85	46.35	39.71	48.19
Multi-stage HMM system with sub-core shape and diacritic models	36.84	**43.41**	40.27	49.44

Regarding machine printed text recognition, a subjective comparison of our text recognition system with other HMM-based machine printed Arabic text recognition systems that are available in the literature is presented in Appendix—B. Moreover, a comparison of our recognition results using the APTI database with results from other systems reported in the literature, that uses the APTI database, is also presented in Appendix—B.

Finally, for handwritten text recognition without the use of handwritten training data, we present, in Table 7.26, our best results on this task and compare them to the best results obtained by our systems when using the complete training set. We can see from the table that the results are significantly lower than the results we obtain when using the full handwritten training sets. Nevertheless, the results are still quite impressive considering the fact that no handwritten training set was used. We feel that this line of research is very promising and can have favorable implications in the future by limiting, if not completely avoiding, the need for handwritten training sets.

Table 7.26: Comparison of results for handwritten text recognition tasks with and without the use of handwritten training sets.

Systems	*WERs*			
	Evaluation Set			
	d	*e*	*f*	*s*
Sub-character HMM system without the use of handwritten training sets	8.39	10.39	13.42	26.89
With Complete Training Sets				
Sub-character HMM system	2.44	5.55	**6.40**	**12.14**
Multi-stage HMM system with core shape and diacritic models	2.46	6.27	7.74	14.88
Multi-stage HMM system with sub-core shape and diacritic models	**2.29**	**5.24**	6.68	13.48

8 Conclusions

Handwritten text recognition is a challenging task. A lot of research has been done in this area with major attention to the Roman script. But, the challenges are far from over. Handwritten Arabic text recognition research has seen a huge interest in the last two decades. Research in handwritten Arabic text recognition benefitted from the extensive research that has already been carried out for other scripts like Roman and Chinese. Consequently, most of the researchers adapted the handwritten text recognition systems, already developed for other scripts, to work for the Arabic script.

HMMs have traditionally been the most successful classifier for text recognition after their success in speech recognition. They have sound theoretical and mathematical foundations. Moreover, they avoid the need to explicitly segment the text line images into smaller units like characters or strokes. This aspect is even more important for Arabic text recognition as Arabic script is inherently cursive both in handwritten and machine printed forms. Adapting the text recognition systems, developed for other scripts, to work for Arabic script led to a quick transfer of technology to this domain and recognizers with reasonably good text recognition capabilities were reported up and running in a fairly short time period. Although the progress was good in general, it led to investigating the knowledge of the Arabic script and improvising on it to build better and more efficient recognizers less explored. The peculiarities of the Arabic script poses its own issues and challenges in the area of text recognition. With challenges comes the opportunities to further enhance the research in this area. The present work was an attempt in this direction, i.e., to study the peculiarities of the Arabic script in the context of text recognition and investigate them in order to contribute to the developments in this area of research.

8.1. Summary

Among the peculiarities of the Arabic script is the aspect of modeling. Arabic characters can take different shapes based on their position in a word. Accordingly, researchers use the character shapes as models instead of using the characters as the modeling unit. This approach leads to almost a four-fold increase in the number of models in the system. Apart from leading to a system with a huge model set, the system also suffers from inadequate training because of the large number of models whose parameters need to be trained effectively from the training data. Performing contextual modeling over these character-shape models further increases the models by many times. To address these problems, we presented three alternative modeling options for Arabic text recognition. As a first option, we presented sub-character modeling where a character is split into sub-characters exploiting the similar patterns between different characters and their position-dependent shapes. The sub-character patterns are then used to reconstruct the characters leading to a huge reduction in the number of HMMs. The sub-character modeling, as presented in this work, does not need explicit segmentation of characters into the smaller units. We also investigated the impact of sub-character modeling from the perspective of contextual HMM modeling and found that the sub-character modeling, including the proposed connector model, lends favorably to contextual HMM modeling and the gains in recognition rates are higher than the gains obtained using the standard character-shape based contextual modeling.

The second modeling approach we investigated was related to separating the core shapes in Arabic texts from the diacritics and modeling the core shapes and the diacritics as separate HMMs. This approach also leads to a large reduction in the number of basic HMMs. A multi-stage text recognition framework was proposed which uses the core-shape HMM system along with the diacritics HMM system to perform text recognition.

Our third and final modeling approach integrates the idea of sub-character modeling with the idea of modeling the core shapes separately from the diacritics. Accordingly, this leads to multi-stage text recognition where sub-core shapes and diacritics are trained separately and text recognition is performed by utilizing the sub-core shape HMM system and the diacritics system. This approach leads to the greatest reduction of the number of modeling units in a system. Contextual HMMs utilizing the sub-core shapes were also investigated with favorable outcomes.

We performed a number of experiments to study the effects of different modeling approaches on the text recognition accuracy. We found that our modeling approaches performed better than the traditional character-shape modeling approach besides the fact that we end up with more compact systems with reduced model sets. This was even more important when performing text recognition under constrained training environments, i.e., when few text image samples are available for training.

Apart from the above mentioned three modeling approaches, we also presented our contributions related to white-space modeling for Arabic text recognition, multi-stream HMMs, and class-based contextual HMM modeling. White-space modeling based on our approach adds more flexibility in text recognition by skipping the space models at times and incorporating them at other times depending on the handwriting. We experimented with multi-stream HMMs by splitting our features into two streams such the features computed from the text images formed one stream and the derivative features formed the second stream. Having different stream weights for the two streams showed improvements in text recognition accuracies. Class-based contextual modeling was presented as a way to limit the number of unique tri-character models that result when using the standard contextual modeling approach. Instead of modeling every tri-character pairs, the neighboring characters are grouped into classes in a way that characters in each class have similar contextual influence to a character whose contexts are being modeled. This resulted in significantly fewer contextual models as compared to the number of contextual models resulting from the standard approach; thereby, leading to a relatively compact recognizer with better recognition performances.

Last but not the least, we investigated some approaches to initialize and train a text recognition system when no handwritten training data is available. Specifically, we studied the impact of training a recognizer with machine printed texts for the handwritten text recognition task. This approach was employed together with unsupervised HMM adaptation. Further, we use the test hypothesis to retrain our recognizer in an iterative approach. We achieved reasonably good text recognition results keeping in mind that no handwritten training set was used. These results can have favorable implication in the future as this can significantly reduce, if not completely avoid, the need of preparing large annotated training sets which is a time consuming and costly task.

8.2. Possible future works

A number of possible future works can enhance the current work. Some of the possible extensions can be as follows:

- The use of NN-based deep learning strategies is gaining a lot of attention from the researchers in the field of text recognition and the results are very promising. A number of published works have shown its superiority over HMMs in terms of text recognition accuracy. NN-based deep learning has also been successfully integrated with HMMs to benefit from both the approaches. Accordingly, an interesting future work can be investigate the effects of utilizing the modeling approaches presented in this work with NN-based deep learning frameworks. Training is, many a times, an issue in deep learning based text recognition systems and thus by significantly reducing the number of basic recognition units, the NN-based systems might benefit even more as compared to a purely HMM-based system.

- It can also be interesting to investigate the impact of the presented modeling approaches on the use of statistical n-grams as language models when using it for text recognition tasks. Specially, the impact of core shape representation of words (after removing the dots and other diacritics) on the language model estimation and even on the word lexicon may be investigated. Many words will get reduced to same core-shape representation thereby reducing the lexicon size. This can potentially lead to efficient recognition and with better overall recognition accuracy in a multi-stage recognition framework.
- For the task of text recognition without the handwritten training data, the use of text synthesis together with the approaches presented in the present work has a potential to further improve the text recognition results and seems worthy of investigation. When using the hypothesis on test data to retrain the system, one generally needs to exclude those hypothesis which have poor scores. Thus, one may end up with only a few training samples. If these handwritten test images can be used, along with its hypothesized annotations, for text image synthesis to augment the training data, more robust training can be performed. Moreover, using machine printed text images and the test images together with text synthesis in a balanced proportion might also prove to be a useful approach.
- Last but not the least, the modeling options we presented in this work needed the knowledge of the script. A possible future work could be to develop methods to automatically decide the optimal modeling units by utilizing pattern recognition techniques. Starting with a system having character-shape models trained on a dataset, techniques to investigate the state-sequence similarity between parts of HMMs maybe one way to explore this possibility. Also, using machine printed text to train the character-shape models and then deciding the ideal number of sub-patterns using state clustering algorithms could prove to be useful as well.

Appendix—A

Transliteration and Translation of Arabic Texts from the Figures in Chapter 3

ارحموا من في الأرض يرحمكم من في السماء

(alif-rā′-ḥā′-mīm-wāw-alif) (mīm-nūn) (fā′-yā′) (alif-lām-alif-rā′-ḍād) (yā′-rā′-ḥā′-mīm-kāf-mīm) (mīm-nūn) (fā′-yā′) (alif-lām-sīn-mīm-alif-hamzah)

Have mercy to those on earth; the one above (God) will have mercy on you.

Figure A–1: Arabic text (top row), its transliteration[5] (middle row), and meaning (bottom row) for Figure 3.2.

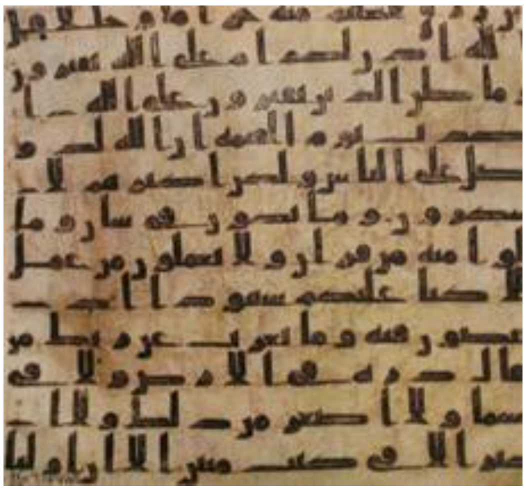
أحب لأخيك ما تحب لنفسك

(alif-ḥā′-bā′) (lām-alif-khā′-yā′-kāf) (mīm-alif) (tā′-ḥā′-bā′) (lām-nūn-fā′-sīn-kāf)

Love for your brother (i.e., others) what you love for yourself.

Figure A–2: Arabic text (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.3.

[5] The transliteration for individual characters (separated by -) are presented from left-to-right although the character sequence in Arabic is from right-to-left. Thus, the first character transliteration from the left represents the first Arabic character from the right. Moreover, the transliterations as presented here does not represent how the character sequences are pronounced. The words are enclosed in ().



—*NA*

And what will be the supposition of those who invent falsehood about Allah (God) on the Day of Resurrection? Indeed, Allah (God) is full of bounty to the people, but most of them are not grateful.

Figure A–3: Arabic text (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.4.

ناس	سلام	إنسان	مجلس
(nūn-alif-sīn)	*(sīn-lām-alif-mīm)*	*(alif-nūn-sīn-alif-nūn)*	*(mīm-jīm-lām-sīn)*
People	Peace	Human	Gathering

(a)

أسود	دنيا	مدرسة	أسد
(alif-sīn-wāw-dāl)	*(dāl-nūn-yā' -alif)*	*(mīm-dāl-rā' -sīn-tā)*	*(alif-sīn-dāl)*
Black	World	School	Lion

(b)

Figure A–4: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.6.

خليفة	سيدي	السلام
(khā’ -lām-yā’ -fā’ -tā)	*(sīn-yā’ -dāl-yā’)*	*(alif-lām-sīn-lām-alif-mīm)*
Caliph	My Master	The Peace

العروسة	الذويبات
(alif-lām-‘ayn-rā’ -wāw-sīn-tā)	*(alif-lām-dhāl-wāw-yā’ -bā’ -alif-tā’)*
The Bride	*Azzūybaat*

Figure A–5: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.7.

السلام	ثالجة	المنزه
(alif-lām-sīn-lām-alif-mīm)	*(thā’ -alif-lām-jīm-tā)*	*(alif-lām-mīm-nūn-zāy-hā’)*
The Peace	*Thaalja*	*Almunzah*

نحّال	مجلس
(nūn-ḥā’ -alif-lām)	*(mīm-jīm-lām-sīn)*
Nihhaal	Gathering

Figure A–6: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.9.

الشرايع	عين	شعال	الرضاع
(alif-lām- shin- rā’ - alif-yā’ - ‘ayn)	*(‘ayn- yā’ - nūn)*	*(shin- ‘ayn-alif- lām)*	*(alif-lām- rā’ - ḍād-alif-‘ayn)*
Asshraae’	Eye	*She’aal*	*Arridae’*

Figure A–7: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.10.

الشلّوف	بو فيشة	نكريف	الفرش
(alif-lām-shin-lām-wāw-fā′)	(bā′ -wāw-fā′ -yā′ -shin-tā)	(nūn-kāf-rā′ -yā′ -fā′)	(alif-lām-fā′ -rā′ -shin)
Assallouf	*Bofesha*	*Nakreef*	*Alfarash*

(a)

الخضراء	القباضة	الفايض	الغزلان
(alif-lām-khā′ -ḍād-rā′ -alif-hamzah)	(alif-lām-qāf-bā′ -alif-ḍād-tā)	(alif-lām-fā′ -alif-yā′ -ḍād)	(alif-lām-ghayn-zāy-lām-alif-nūn)
Green	*Alqabaadha*	*Alfaaedh*	*Algazlaan*

(b)

Figure A–8: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.11.

الشرايع	المنزه	عجنقة	سيدي	ربانة
(alif-lām- shin- rā′ - alif- yā′ - ′ayn)	(alif-lām-mīm-nūn-zāy-hā′)	(′ayn- jīm- nūn- qāf-tā)	(sīn-yā′ -dāl-yā′)	(rā′ - bā′ - alif- nūn-tā)
Asshraae′	*Almunnaza*	*E′jneka*	My Master	*Rabaana*

مارث	الخليج	بولحناش	شوّاط
(mīm-alif- rā′ - thā′)	(alif-lām- khā′ - lām- yā′ - jīm)	(bā′ - wāw- lām- ḥā′ - nūn-alif- shīn)	(shin- wāw- alif- ṭā′)
Maarth	The Gulf	*Bulhanaash*	*Shawwaat*

Figure A–9: Arabic texts (top row), its transliteration (middle row), and meaning (bottom row) for Figure 3.12.

Appendix—B

We will present the experiments and the results related to machine printed Arabic text recognition based on the approaches presented in Section 6.2. First, we will present the machine printed text databases we used for the experiments. This will be followed by the details of the experiments, the results, and the discussions.

B.1. Databases for machine printed text recognition

We used two machine printed Arabic text databases for the experiments. Below we present a brief description of both the databases.

B.1.1. P-KHATT database of machine printed texts

The P-KHATT database is the machine printed version of the KHATT database presented in Section 7.1.2. The database includes text from eight different fonts; each text is divided into three non-overlapping sets (train, development, and test). The train, development, and test sets contain 6472, 1414, and 1424 text line images respectively. The text and the divisions are similar to the text and divisions of the KHATT database. Figure B–1 presents sample text images from the P-KHATT database in eight fonts. In addition to the data and the images for the eight fonts, the P-KHATT database has text line images and their annotations for a ninth font for the purpose of text recognition on unseen fonts. The ninth font does not include the training and the development sets. The text documents were printed using a laser printer and, then, scanned at a resolution of 300 Dots-Per-Inch (DPI).

B.1.2. APTI database of machine printed texts

The APTI database is a publicly available database of machine printed Arabic texts and is free for noncommercial use (Slimane et al. 2009). The database contains low-resolution (72 DPI) synthetically generated printed Arabic word images in many fonts, sizes, and styles. The database is partitioned into six sets for each combination of font, size, and style. Five of the six sets are open, whereas; the sixth set has not been disclosed to the public and is employed in competitions to evaluate submitted OCR systems. According to the database developers, the characteristics of the sixth set is similar to the characteristics of the remaining sets (Slimane et al. 2009). Each set contains different word images but the distribution of the characters is nearly identical in every set. Figure B‒2 presents sample text images from the APTI database. Some experiments were conducted using the APTI database in order to validate our approaches on multiple databases and as well as to use it as a benchmark to compare our results to other OCR systems reported in the literature that uses the same database under similar experimental configurations.

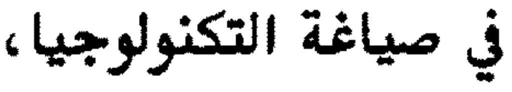
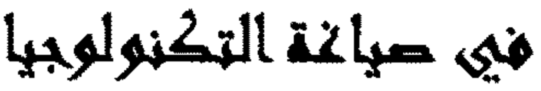
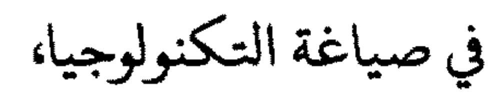
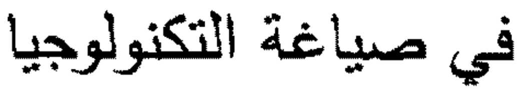
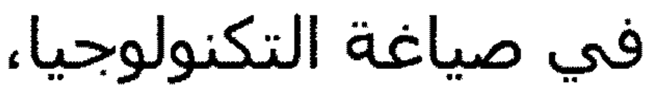
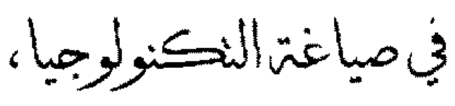
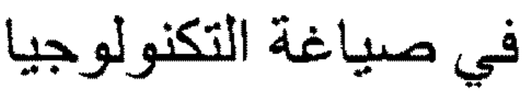
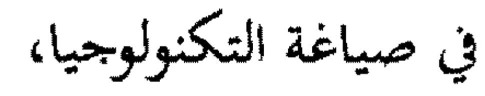
Font (Code)	*Sample Text Image*
Akhbar (AKH)	 في صياغة التكنولوجيا،
Andalus (AND)	 في صياغة التكنولوجيا،
Naskh (NAS)— (KFGQPC Uthman Taha Naskh)	 في صياغة التكنولوجيا،
Simplified Arabic (SIM)	 في صياغة التكنولوجيا،
Tahoma (TAH)	 في صياغة التكنولوجيا،
Thuluth (TLT) — DecoType Thuluth	 في صياغة التكنولوجيا،
Times New Roman (TNR)	 في صياغة التكنولوجيا،
Traditional Arabic (TRA)	 في صياغة التكنولوجيا،

Figure B‒1: Sample text line images in different fonts from the P-KHATT database. Image degradation due to the printing and scanning process is apparent (Image source (Ahmad, Mahmoud, and Fink 2016)).

B.2. Experimentations, results, and discussions

In this section, we will present our experiments and results conducted on the P-KHATT and the APTI databases of machine printed texts. First, we will present the experiments conducted on the P-KHATT database which will be followed by the experiments conducted on the APTI database.

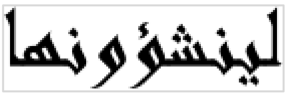
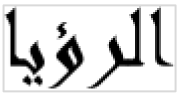
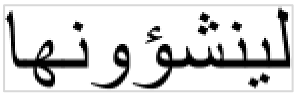
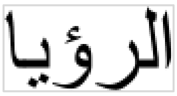
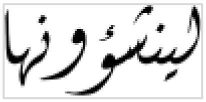
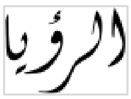
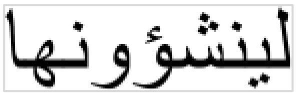
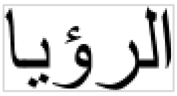
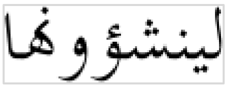
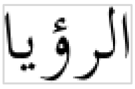
Font (Code)	*Sample Text Images*	
Andalus	 لينشؤونها	 الرؤيا
ArabicTransparent	 لينشؤونها	 الرؤيا
Diwani Letter	 لينشؤونها	 الرؤيا
Simplified Arabic	 لينشؤونها	 الرؤيا
Traditional Arabic	 لينشؤونها	 الرؤيا

Figure B-2: Sample text line images in different fonts from the APTI database.

B.2.1. Experiments using the P-KHATT database

The text line images were normalized to a fixed height of 96 pixels while maintaining the aspect ratio of individual line images. Next, sliding window technique was used to extract the features from the normalized text line images. The frames of the sliding window were further divided into individual cells (a total of 6 cells). The height of individual cells were not constant as is commonly the case. Instead, the height of the cells were adaptive to the text line such that the cells are smaller around the writing line where the pixel concentration is higher and the size gradually increase as we move away from the writing line (both below and above it). A cell is placed around the writing line and a number of cells are placed above it and below it. The number of cells below the writing line is less than the number of cells above the line as this design suits the properties of Arabic script. The algorithm for cell division of the sliding window is presented in Figure B-3. The width and the overlap for the window frames were decided based on the text recognition results on the evaluation set on one of the fonts (*Times New Roman*). The sliding window width and overlap for the *Thuluth* font was calibrated separately using its development set as the *Thuluth* font has quite different characteristics as compared to other fonts in the database. Density of ink pixels were computed from each cell of the sliding window frames running across the text line images in addition to its horizontal and vertical derivative images. The features from each frame were concatenated for all the three images such the feature vector has the dimension 18.

Our text recognition system is based on continuous HMMs. We use HTK tools (Young et al. 2002) to implement our recognizer. Each character shape is treated as an individual model with a simple right-to-left linear topology. White space was explicitly modeled using a separate HMM. We have a total of 153 different HMMs in our recognition system. Each character-shape HMM was modeled with the same number of states, with the exception of some narrow-width characters (such as *alif* ا), which were modeled with half the number of states. The optimal number of states (for each font) was determined based on the uniform initialization (flat start) recognition results on the font's development set.

```
User Inputs:= total no. of cells 'totalCells' in a frame,
               no. of cells 'cellsAbove' above the writing line,
               text-line image

•  Place the cell $cell_{wl}$ around the writing line such that the baseline
   of the text is in the middle of the cell. The width of the cell shall
   be such that the following condition is satisfied:
        $\frac{(Sum\ of\ ink\ pixels\ within\ the\ cell)}{(total\ ink\ pixels)} \cong \frac{1}{(total\ no.of\ cells\ in\ a\ frame)}$

•  Divide the area above the $cell_{wl}$ into 'cellsAbove' cells such that
   each of the cell has the same percentage of ink pixels.

•  Divide the area below the $cell_{wl}$ into 'totalCells – cellsAbove - 1'
   cells such that each of the cell has the same percentage of ink
   pixels.
```

Figure B–3: Algorithm for determining the size and the position of the cells of the sliding window frames.

We employed 2,000 text line images for training instead of the complete training set for each font. Training was conducted in two stages. In the first stage, uniform initialization (flat start) was performed using the training data. In the next stage, the alignment information from the training data was employed to initialize individual HMMs using Viterbi initialization followed by a number of iterations of Baum-Welch training. Character hypotheses for the evaluation set were generated using Viterbi decoding. The experimental results related to configuring the sliding window width and overlap are presented in Table B–1. Accordingly, based on the results, we applied a window width of 6 pixels with an overlap of 3 pixels for all other fonts (except the *Thuluth* font which has window width of 4 pixels with an overlap of 2 pixels).

Once the sliding window parameters were selected, we performed the two-step training (i.e., uniform initialization and alignment-based initialization) for all eight fonts. The recognition results for each of the eight fonts are presented in Table B–2. The best result—CER 1.04%—was achieved for the *Tahoma* font. The worst result—CER 7.55%—was achieved for the *Thuluth* font. The mean CER of 2.89% was achieved for the eight fonts on the evaluation sets.

Table B–1: Configuration of sliding window's width and overlap based on evaluations on the development set using the *Times New Roman* font from the P-KHATT database.

Window (W|O)[*]	*No. of states*	*CER (%)*
4|2	10	1.43
2|0	11	1.78
3|1	10	2.54
1|0	17	2.60
3|0	7	1.65
4|1	7	1.37
5|2	7	1.26
6|3	**7**	**1.23**
4|0	6	2.32
6|2	6	1.87
8|4	5	1.56

[*]**W**: Width; **O**: Overlap

Table B–2: Summary of the character recognition results for mono-font machine printed text recognition using the P-KHATT database.

Font	*Window (W|O)*	*No. of States*	*CER (%)*		*Statistical Significance*
			Development	*Evaluation*	
Times New Roman	6|3	7	1.23	1.20	±0.06
Andalus	6|3	8	1.20	1.35	±0.07
DecoType Thuluth	**4|2**	7	7.51	**7.55**	±0.15
Tahoma	6|3	9	1.00	**1.04**	±0.06
Traditional Arabic	6|3	6	4.75	4.35	±0.12
Naskh	6|3	6	2.61	3.06	±0.10
Akbaar	6|3	6	2.80	2.87	±0.09
Simplified Arabic	6|3	7	2.02	1.67	±0.07
Mean			2.89	**2.89**	

Our next set of experiments was related to mixed font text recognition. As an initial experiment, we trained a mixed-font recognizer (i.e., the recognizer was trained using training samples from all fonts). The optimal HMM parameters were selected based on the results from the development set, and a final evaluation was conducted on the evaluation set. A CER of 12.19% for the development set and a CER of 12.14% for the evaluation sets were achieved, which are significantly higher than the mean CER of 2.89% that was achieved for mono-font text recognition. This increase in error rates can be partly explained by the large variation in font styles and the fact that each font has individual parameters (such as number of states), which is difficult to generalize. This motivated us to explore font-identification-based recognition, as described in Section 6.2.2, in which the images' font is identified in the first step and the mono-font recognizer for the identified font is subsequently employed for text recognition in the second step.

For the font-identification-based recognition, we trained the font identification module. The font features, as described in Section 6.2.1, were computed from the training samples for each of the eight fonts. An SVM, with Radial Basis Function (RBF) as the kernel, was employed as a classifier. The font identification module was evaluated using a set that contained 1414 text line images for each font, which were randomly distributed. Table B–3 presents the font identification results and the confusion matrix. As shown in the table, we achieve reasonable results for the font identification, which demonstrates the effectiveness of our proposed features for font identification. Common confusion occurred between the *Simplified Arabic* font and the *Times New Roman* font. A closer look at the text images from the two fonts reveals that the two fonts are quite similar; this observation has been noted in other studies (cf., e.g., (Luqman, Mahmoud, and Awaida 2014)). Another observation is that both fonts employ the same number of HMM states per model, which provides clues regarding their similar properties. To confirm that the fonts are indeed similar, we recognized the text images from the *Times New Roman* font using the mono-font recognizer that was trained on the *Simplified Arabic* font. A CER of 3.68% was achieved, which confirmed that the two fonts are not only visually similar but also exhibit similar properties with respect to text recognition. When they were combined as one font, our font identification rate was 97.27%.

Table B–3: Font identification results and the confusion matrix on the P-KHATT database.

Font	*AKH*	*AND*	*NAS*	*SIM*	*TAH*	*TLT*	*TNR*	*TRA*	***Identification Rate (%)***
AKH	1337	0	20	11	2	4	1	39	94.55
AND	2	1402	0	1	2	4	1	2	99.15
NAS	9	1	1352	1	0	20	2	29	95.62
SIM	5	1	14	1269	2	3	118	2	89.75
TAH	2	0	0	0	1405	4	1	2	99.36
TLT	2	1	26	2	0	1367	2	14	96.68
TNR	2	1	3	129	3	3	1272	1	89.96
TRA	14	1	33	3	0	8	3	1352	95.62
								Mean	**95.08**

After associating the font of the input text image, we performed feature extraction and recognition using the mono-font text recognizer of the associated font. With this approach, we achieved a CER of 3.44%, which is closer to the mean CER that we achieved in the mono-font setups. Thus, the results demonstrate the effectiveness of this approach compared with the commonly employed approach of recognizing the text image using a recognizer that is trained on multiple fonts. To understand the recognition errors caused by errors in font identification, we conducted another experiment, in which we manually separated the text line images based on the font and then recognized the text images using each font's individual mono-font recognizer. The CER was 2.86%; thus, the text recognition error caused by the error in font identification was 0.58% (i.e., 3.44–2.86). Table B–4 summarizes the results of the recognition for both scenarios.

Table B–4: Summary of the results (in CER) for the mixed-font machine printed text recognition experiments using the P-KHATT database.

Setup	*CER (%)*
Recognizer trained on samples from all fonts	12.19
Using font-association-based recognition	3.44
Recognition using the mono-font recognizer after manually separating text lines into different fonts.	2.86

In the last set of experiments involving the P-KHATT database, we performed text recognition on an unseen font. We attempted different configurations to investigate the effectiveness of various approaches presented in Section 6.2.2. In the first experiment, we recognized the unknown font's text images using the recognizer trained on text line images from all the eight fonts. In the second experiment, we associated the input text line images to the closest of the eight fonts using the font association module and subsequently employed the associated font's recognizer to recognize the input text. In the next few experiments, we evaluated the HMM adaptation techniques that were presented in Section 6.2.2. In one of the experiments, we investigated unsupervised HMM adaptation, in which no labeled data for the unseen fonts were employed. Recognition was performed after the adaptation step. In another set of experiments, we employed 100 labeled text line images for the unseen font to perform MLLR-based supervised adaptation. A summary of the recognition results for the unseen fonts is presented in Table B–5. Although the supervised and unsupervised adaptation techniques improve the results, the improvements based on the supervised adaptation are optimal, which is understandable. It assumes the availability of labeled samples for the input font, which may not always be feasible.

Table B–5: Summary of the character recognition results (in CER) for the unseen font using the P-KHATT database.

System Description	*CER (%)*
Recognizer trained on samples from all fonts	19.28
Recognizer for the closest identified font	15.39
Recognizer for the closest identified font + Unsupervised adaptation	11.76
Recognizer for the closest identified font + Supervised adaptation	9.43

We present a subjective comparison of our text recognition system with other HMM-based printed Arabic text recognition systems that have been discussed in the literature. Only studies that performed text recognition using text lines instead of systems that recognized isolated characters, digits, or word images were selected. Systems that employed synthetic databases were not selected because they did not address many of the practical challenges of real and scanned databases. In Table B–6, we present a comparative study of different studies related to printed Arabic text line image recognition. This comparison was not performed to quantitatively compare different works because this task would be impossible due to the different databases utilized by different groups. Thus, this comparison should be understood from a complementary viewpoint. In the comparison presented in Table B–6, we highlight different aspects of the study, such as the selected database, which was considered to be one of the most important aspects. The nature of the database, its text sources, its characteristics (such as scanning resolution and noise level), and its division into different sets (for training, development, and evaluation) serve an important role in text recognition performance.

Another important aspect is the nature of text recognition with respect to font variability. Some studies only reported their results for mono-font or mixed-font text recognition, whereas other studies discussed the performance of both mono-font and mixed-font text recognition. Our current work focuses on mixed-font text recognition as well as text recognition of unseen fonts. Other important aspects include the decoding network and the use of language models. Some studies optionally decode at the character level using character n-grams as their language models. Other studies have employed word lexicons with the optional use of word n-grams as language models. The issue of out-of-vocabulary (OOV) words is important when using word lexicons in open vocabulary word recognition tasks. One study (Prasad et al. (Prasad et al. 2008)) also investigated the use of parts of Arabic words (PAW) language models. These models can also be used after decoding to re-score the N-best list that is generated during decoding. In addition to these aspects, other aspects can be compared between different studies, including the nature of the HMM system (continuous vs. discrete vs. systems with differing levels of tying, e.g., mixture tying and state tying), the sliding window technique and features employed for recognition.

Table B–6: A Subjective comparison of other HMM-based printed Arabic text recognition systems that perform recognition at the text line level.

Work	Characteristics of the database	Main aspects of text recognition	System description	Error rates (%)
Bazzi et al. (Bazzi, Schwartz, and Makhoul 1999)	DARPA Arabic OCR Corpus of 345 pages of Arabic text scanned at 600 DPI For mixed font text recognition: • Text line images from 30 pages were used for training • Text line images from 10 pages were used for evaluation	• Mono-font text recognition • Mixed-font text recognition where the training set and the evaluation set contains line images from four different fonts	• HMM-based OCR system • Pixels density features with vertical and horizontal derivatives in addition to local slope and correlation features across a window of two cells • lexicon obtained from a large text corpus with closed vocabulary of 30k words • A language model for recognition from the same text corpus	CER of 0.40 for mono-font text recognition CER of 2.60 for mixed-font system with closed vocabulary word recognition CER of 4.50 on mixed font open vocabulary text recognition using trigram character language model
Natarajan et al. (Premkumar Natarajan et al. 2001)	DARPA Arabic OCR Corpus of 345 pages of Arabic text scanned at 600 DPI • Text line images from 192 text zones were used for training • Text line images from 102 text zones were used for evaluation	• Mixed-font text recognition	• HMM-based OCR system with mixture tying at character level • Percentile features with vertical and horizontal derivatives in addition to local slope and correlation features	CER of 3.86
Khorsheed (Khorsheed 2007)	A database of 15,000 text line images in six different fonts, i.e., 2,500 text line images in each font. • Training set includes 1,500 text line images in each of the six font	• Mono-font text recognition for six different fonts	• Discrete-HMMs based OCR system • Pixel density features extracted from the sliding windows over the text line images and their horizontal and vertical derivatives • Contextual HMM modeling	CER ranging from 7.40 (for *Andalus* font) to 14.00 (for *Naskh* font)

	• Development Set includes 1,000 text line images in each of the six font		• Character bigrams from training transcriptions	
Prasad et al. (Prasad et al. 2008)	DARPA Arabic Machine Print (DAMP) scanned at 600 DPI • Training set includes text line images from 177 page images in addition to text line images from 380 synthetically generated page images in multiple fonts and sizes • Development set includes text line images from 60 page images • Evaluation set includes text line images from 60 page images	• Mixed-font text recognition	• HMM-based OCR system with discriminative training • Position-dependent tied mixtures where the Gaussians for corresponding states of all the presentation forms of character is tied • Contextual HMM modeling • Character, PAW, and word trigrams from 2.6 million words of Arabic newswire data in addition to the training transcriptions • Word lexicon of 65k words	Best word error rate of 9.60 using PAW language model and N-Best rescoring using contextual HMMs estimated using discriminative training procedure
Dreuw et al. (Dreuw et al. 2012)	RAMP-N printed Arabic database in 20 different fonts scanned at 600 DPI: • 222,421 text line images for training • 1,155 text line images for the development set • 3,480 text line images for the evaluation set	• Mixed-font text recognition (two of the fonts cover more than 95% of all the text line images in the evaluation set) • Word recognition task with Out Of Vocabulary rate of 2.21%	• HMM-based system with ML trained GMMs with globally pooled variances • Appearance-based image slice features along with spatial derivatives • Language model using a corpus of 228 million running words • Vocabulary size of 106k words	WER of 4.76 and CER of 0.15 on the rendered data WER of 5.79 and CER of 0.66 on the scanned data

Present Work	P-KHATT printed Arabic text database in eight different fonts scanned at 300 DPI: • Training set includes 6,472 text line images in each of the eight font (2,000 text line images used for training in current work) • Development Set includes 1,414 text line images in each of the eight font • Evaluation Set includes 1,424 text line images in each of the eight font	• Mono-font text recognition • Mixed-font text recognition • Text recognition for unseen font (i.e., having no training samples)	• HMM-based OCR system • Adaptive sliding window for feature extraction • Pixel density features and its vertical and horizontal derivatives • Font identification based text recognition • Use of supervised and unsupervised HMM adaption techniques to deal with font variability • Character bigrams from training transcriptions	CER ranging from 1.04 (for Tahoma) to 7.55 (for Thuluth) for mono-font text recognition without using any language model or word lexicon CER of 3.44 for mixed-font text recognition without using any language model or word lexicon For unseen-font text recognition: CER of 11.76 using unsupervised adaptation without any language models and lexicon CER 7.18 using supervised adaptation and character bigrams as language model

B.2.2. Experiments using the APTI database

In this section, we will present the character recognition experiments we conducted on the publically available APTI database. As mentioned before, we conducted some experiments using the APTI database in order to validate our machine printed text recognition approaches on multiple databases and as well as to use it as a benchmark to compare our results to other OCR systems reported in the literature that uses the same database under similar experimental configurations. The details of the system initialization, training, and decoding are similar to the ones presented for the experiments using the P-KHATT database. The minor differences will be presented along the discussions of the results.

In the first set of experiments involving the APTI database, we perform mono-font text recognition. We experimented with five different fonts from the APTI database—the same five fonts were selected in the first competition that was held using the APTI database (Slimane et al. 2011). For each font, we selected 24-point images in plain text. Set 1 was selected as the training data, and 3000 images from set 2 were selected as the development set to optimally configure the number of states per HMM. Set 5 was utilized to evaluate the system's performance. All images, with the exception of the images in the *Diwani Letter* font, were height-normalized to 64 pixels while maintaining a constant aspect ratio. Because *Diwani Letter* is very compact with many vertically overlapping ligatures, it was height-normalized to 96 pixels. An explicit white space model was not employed in these experiments.

The mono-font text recognition results for the individual fonts is presented in Table B–7. The best results were obtained for the *Arabic Transparent* font, whereas the results for the *Andalus* and *Simplified Arabic* fonts were also comparable. The poorest results were obtained for the *Diwani Letter* font, which is a complex and a compact font. The mean CER for all the fonts was 2.07%. For optimal performance, parameters such as the sliding window width and overlap, the number of mixtures per HMM state, and the image height can be calibrated for individual fonts using their corresponding development sets. Use of the font-specific ligature models also has the potential to improve recognition performances (e.g., (Slimane et al. 2010; Slimane et al. 2012)).

Table B–7: Summary of the character recognition results for mono-font machine printed text recognition using the APTI database.

Font	*CER (%)*
Andalus	0.76
Arabic Transparent	0.57
Diwani Letter	4.67
Simplified Arabic	0.69
Traditional Arabic	3.65
Mean	**2.07**

In the next set of experiments, we performed mixed-font text recognition. Similar to the experiments with the P-KHATT database, we investigated two approaches to this text recognition task. The first approach was to train an HMM recognizer using samples from all five fonts. The second approach was to perform font-association-based recognition. For the first approach, we selected 3000 word images from set 1 of each font at 24–point size; the training data included 15,000 word images. The optimal number of states for the HMM was selected based on the recognition performance on the development set, which included 600 word images from set 2 of each font (a total of 3000 images). The final evaluation was conducted using the evaluation set, which included 15,000 word images in the five fonts (3000 images from each font from set 5). A CER of 7.71% was obtained that was reasonable but higher than the mean CER of 2.07%, which was achieved in the mono-font experiments.

For the second approach, we train our font-association module, which utilizes an SVM classifier with RBF kernel. The font identification features, as proposed in Section 6.2.1, were extracted from the 15,000 word images in the training set. These features and the information about the word image font typefaces were employed to train the SVM classifier. The trained classifier was applied to associate the word images' font in the evaluation set. The font identification results for the evaluation set are presented in Table B–8. An average identification rate of 96.99% was obtained. After associating the input text images' font, we perform feature extraction and recognition using the mono-font text recognizer for the associated font. Using this approach, we achieved a CER of 2.92%, which demonstrates that the two-step font-association-based text recognition proved to be a better approach than performing text recognition trained on multiple font images. In Table B–9, we summarize the text recognition results for the mixed-font text recognition task. When the text line images were manually separated based on the font, the CER was 2.12%. Consequently, the recognition errors caused by the misclassified fonts was 0.80% (i.e., 2.92–2.12).

Table B–8: Font identification results and the confusion matrix on the APTI database.

Font	*Andalus*	*Arabic Transparent*	*Diwani Letter*	*Simplified Arabic*	*Traditional Arabic*	***Identification Rate (%)***
Andalus	2994	2	0	3	1	99.80
Arabic Transparent	0	2806	12	178	4	93.53
Diwani Letter	0	4	2944	0	52	98.13
Simplified Arabic	0	140	0	2856	4	95.20
Traditional Arabic	3	3	44	2	2948	98.27
					Mean	**96.99**

Table B–9: Summary of the results (in CERs) for the mixed-font machine printed text recognition experiments using the APTI database.

Setup	***CER (%)***
Recognizer trained on samples from all fonts	7.71
Using font-association based recognition	2.92
Recognition using mono-font recognizer after manually separating text lines of different fonts.	2.12

We compare our text recognition results using the APTI database with results from other HMM systems that have been reported in the literature using the APTI database. The comparison is presented in Table B–10. The comparison is based on recognition results on *Arabic Transparent* font because this font was included in the reference protocols for the text recognition competitions that employed the APTI database (Slimane et al. 2011; Slimane et al. 2013). For the remaining fonts, the results in the competitions are presented for mixed-font and multi-size text recognition scenarios. Thus, comparisons including other fonts are not possible. For the *Arabic Transparent* font, a completely objective comparison is still not possible for many reasons. One of the most important reasons is that set 6, which is not publicly available, was employed to evaluate the systems in the competitions. For the systems that utilized the APTI database and that are available in the literature, each group created individual training, development, and evaluation set partitions. Some systems applied word lexicons and n-gram language models, whereas other systems did not use any word lexicons or language models. For some systems, these details are not explicitly mentioned. Nevertheless, the comparison table can provide useful qualitative insights.

Table B–10: Comparison with other HMM-based text recognition systems evaluated using the APTI database.

OCR systems	Database setup for experimentation	Error rates (%)	System description
UPV-PRHLT (cf. (Slimane et al. 2011))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - set 1 to set 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 4.00 Word level: 15.60	- Bernoulli-mixture-based HMM system (BHMM)
Awaida and Khorsheed (Awaida and Khorsheed 2012)	APTI database of printed Arabic text - Training Set: 80,000 images Sub-Training Set: 8,000 images Development Set: 1,000 images - Evaluation Set: 14,418 images	Character level: 3.35	- Discrete HMM-based OCR system - Sliding-window-based run-length encoding (RLE) features - Number of states per model and codebook size for feature quantization were optimized using the development set
IPSARec System (cf. (Slimane et al. 2011))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - sets 1 to 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 3.20 Word level: 22.50	- Discrete HMM-based OCR system - Pixel density features from the text image and its horizontal and vertical derivatives
THOCR1 (cf. (Slimane et al. 2013))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - set 1 to 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 1.05 Word level: 8.23	- HMM-based OCR system - Statistical and structural features and their derivatives - No language model used

THOCR2 (cf. (Slimane et al. 2013))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - sets 1 to 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 0.81 Word level: 4.97	- HMM-based OCR system - Statistical and structural features and their derivatives - Four-gram language model trained on the APTI training corpus used for rescoring
Khoury et al. (Khoury et al. 2013)	APTI database of printed Arabic text. Font: Arabic Transparent, size: 24 - Training set: 10,000 images - Development set: 2000 images - Evaluation set: 3000 images	Character level: 0.30	- Bernoulli-mixture-based HMM system (BHMM) - Image height, sliding window width, number of states per model, and number of mixture components per state were optimized using the development set. - Five-gram language model at the character level
UPV-BHMM (cf. (Slimane et al. 2013))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - sets 1 to 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 0.04 Word level: 0.10	- Character-based windowed BHMMs (Bernoulli HMMs) - Image height, sliding window width, number of states per model, and number of mixture components per state were optimized using the development set. - Five-gram language model at the character level
DIVA-REGIM (cf., (Slimane et al. 2011; Slimane et al. 2010))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - sets 1 to 5: used as training and development sets - set 6 (not publicly available) used for evaluation	Character level: 0.30 Word level: 1.10	- HMM-based OCR system - Character shape as HMM models with some models merged into one model, which produced a total of 65 HMM models - Ergodic HMM topology with all possible transitions allowed - System parameters tuned using sets 1 to 5 - Number of connected black and white components, centers of gravity, density, compactness, vertical and horizontal - projection, baseline position, number of relative extrema in the vertical projection, and number of relative extrema in the horizontal projection used as features and their horizontal derivatives

SID (cf. (Slimane et al. 2013))	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - set 1 to 5: used as training and development set - set 6 (not publicly available) used for evaluation	Character level: 0.01 Word level: 2.59	- HMM-based OCR system - Sliding-window-based features
Present Work	APTI database of printed Arabic text Font: Arabic Transparent, size: 24 - Set 1 used as training set - Set 2 used as development set - Set 5 used as evaluation set	Character level: 0.57 Word level: 2.12	- HMM-based system with adaptive sliding window features and statistical feature in addition to its horizontal derivatives - Number of states per model optimized using the development set - No use of language models or word lexicons.

Bibliography

Abandah, Gheith A., Fuad T. Jamour, and Esam A. Qaralleh. 2014. Recognizing Handwritten Arabic Words Using Grapheme Segmentation and Recurrent Neural Networks. *International Journal on Document Analysis and Recognition (IJDAR)* 17 (3): 275–91. doi:10.1007/s10032-014-0218-7.

Abed, Haikal El, and Volker Märgner. 2007. Comparison of Different Preprocessing and Feature Extraction Methods for Offline Recognition of Handwritten ArabicWords. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:974–78. doi:10.1109/ICDAR.2007.4377060.

———. 2009. How to Improve a Handwriting Recognition System. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 1181–85. IEEE. doi:10.1109/ICDAR.2009.11.

Abuhaiba, Ibrahim S., M.J.J Holt, and S Datta. 1998. Recognition of Off-Line Cursive Handwriting. *Computer Vision and Image Understanding* 71 (1): 19–38. doi:10.1006/cviu.1997.0629.

Abulhab, Saad D. 2009. Roots of Modern Arabic Script: From Musnad to Jazm. *Sawt Dahesh* 50–51.

Ahmad, Irfan, and Gernot A. Fink. 2015a. Multi-Stage HMM Based Arabic Text Recognition with Rescoring. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 751–55. IEEE. doi:10.1109/ICDAR.2015.7333862.

———. 2015b. Training an Arabic Handwriting Recognizer without a Handwritten Training Data Set. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 476–80. IEEE. doi:10.1109/ICDAR.2015.7333807.

Ahmad, Irfan, Gernot A. Fink, and Sabri A. Mahmoud. 2014. Improvements in Sub-Character HMM Model Based Arabic Text Recognition. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, 537–42. Crete: IEEE. doi:10.1109/ICFHR.2014.96.

Ahmad, Irfan, Sabri A. Mahmoud, and Gernot A. Fink. 2016. Open-Vocabulary Recognition of Machine-Printed Arabic Text Using Hidden Markov Models. *Pattern Recognition* 51 (March): 97–111. doi:10.1016/j.patcog.2015.09.011.

Ahmad, Irfan, Leonard Rothacker, Gernot A. Fink, and Sabri A. Mahmoud. 2013. Novel Sub-Character HMM Models for Arabic Text Recognition. In *Proceedings* *of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, 658–62. IEEE. doi:10.1109/ICDAR.2013.135.

Ait-Mohand, Kamel, Thierry Paquet, and Nicolas Ragot. 2014. Combining Structure and Parameter Adaptation of HMMs for Printed Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (9): 1716–32. doi:10.1109/TPAMI.2014.2306423.

Al-Badr, Badr, and Robert M. Haralick. 1998. A Segmentation-Free Approach to Text Recognition with Application to Arabic Text. *International Journal on Document Analysis and Recognition (IJDAR)* 1 (3): 147–66. doi:10.1007/s100320050014.

Al-Badr, Badr, and Sabri A. Mahmoud. 1995. Survey and Bibliography of Arabic Optical Text Recognition. *Signal Processing* 41 (1). Elsevier: 49–77.

Al-Hajj Mohamad, R, Laurence Likforman-Sulem, and Chafic Mokbel. 2009. Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (7). IEEE: 1165–77.

Al-Hajj Mohamad, R, Chafic Mokbel, and Laurence Likforman-Sulem. 2007. Combination of Hmm-Based Classifiers for the Recognition of Arabic Handwritten Words. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:959–63.

Al-Maadeed, S., Dave Elliman, and Colin Higgins. 2002. A Data Base for Arabic Handwritten Text Recognition Research. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR 2002)*, 485–89. IEEE Comput. Soc. doi:10.1109/IWFHR.2002.1030957.

Al-Ohali, Yousef, Mohamed Cheriet, and Ching Y. Suen. 2003. Databases for Recognition of Handwritten Arabic Cheques. *Pattern Recognition* 36 (1): 111–21. doi:10.1016/S0031-3203(02)00064-X.

Al-Shatnawi, A M, and K Omar. 2009. A Comparative Study between Methods of Arabic Baseline Detection. In *Proceedings of the International Conference on Electrical Engineering and Informatics*, 1:73–77. doi:10.1109/ICEEI.2009.5254814.

Alamri, Huda, Chun He, and Ching Y. Suen. 2009. A New Approach for Segmentation and Recognition of Arabic Handwritten Touching Numeral Pairs. Edited by Xiaoyi Jiang and Nicolai Petkov. *Computer Analysis of Images and Patterns*, Lecture Notes in Computer Science, 5702. Berlin, Heidelberg: Springer Berlin Heidelberg: 165–72. doi:10.1007/978-3-642-03767-2.

Alma'adeed, Somaya, Colin Higgins, and Dave Elliman. 2002. Recognition of off-Line Handwritten Arabic Words Using Hidden Markov Model Approach. In *Proceedings of the Object Recognition Supported by User Interaction for Service*

Robots, 3:481–84. IEEE Comput. Soc. doi:10.1109/ICPR.2002.1047981.

———. 2004. Off-Line Recognition of Handwritten Arabic Words Using Multiple Hidden Markov Models. *Knowledge-Based Systems* 17 (2–4): 75–79. doi:http://dx.doi.org/10.1016/j.knosys.2004.03.002.

Almuallim, H, and S Yamaguchi. 1987. A Method of Recognition of Arabic Cursive Handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5. IEEE: 715–22.

Amin, Adnan, Humoud Al-Sadoun, and Stephen Fischer. 1996. Hand-Printed Arabic Character Recognition System Using an Artificial Network. *Pattern Recognition* 29 (4): 663–75. doi:10.1016/0031-3203(95)00110-7.

Awaida, Sameh M., and Mohammad S. Khorsheed. 2012. Developing Discrete Density Hidden Markov Models for Arabic Printed Text Recognition. In *Proceedings of the IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom)*, 35–39. IEEE. doi:10.1109/CyberneticsCom.2012.6381612.

Awaida, Sameh M., and Sabri A. Mahmoud. 2009. A Multiple Feature/resolution Scheme to Arabic (Indian) Numerals Recognition Using Hidden Markov Models. *Signal Processing* 89 (6): 1176–84.

Azeem, SherifAbdel, and Hany Ahmed. 2013. Effective Technique for the Recognition of Offline Arabic Handwritten Words Using Hidden Markov Models. *International Journal on Document Analysis and Recognition (IJDAR)* 16 (4). Springer Berlin Heidelberg: 399–412. doi:10.1007/s10032-013-0201-8.

Azizi, Nabiha, Nadir Farah, Mokhtar Sellami, and Adellatif Ennaji. 2010. Using Diversity in Classifier Set Selection for Arabic Handwritten Recognition. In *Proceedings of the 9th International Workshop on Multiple Classifier Systems (MCS)*, edited by Neamat Gayar, Josef Kittler, and Fabio Roli, 235–44. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-12127-2_24.

Baird, Henry S. 2007. The State of the Art of Document Image Degradation Modelling. In *Digital Document Processing*, edited by Bidyut B. Chaudhuri, 261–79. Advances in Pattern Recognition. London: Springer London. doi:10.1007/978-1-84628-726-8.

Bazzi, Issam, C. LaPre, John Makhoul, C. Raphael, and Richard Schwartz. 1997. Omnifont and Unlimited-Vocabulary OCR for English and Arabic. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR 1997)*, 2:842–46. doi:10.1109/ICDAR.1997.620630.

Bazzi, Issam, Richard Schwartz, and John Makhoul. 1999. An Omnifont Open-Vocabulary OCR System for English and Arabic. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence 21 (6): 495–504. doi:10.1109/34.771314.

Benouareth, A., Adellatif Ennaji, and Mokhtar Sellami. 2006. HMMs with Explicit State Duration Applied to Handwritten Arabic Word Recognition. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006)*, 2:897–900. IEEE. doi:10.1109/ICPR.2006.631.

———. 2008. Semi-Continuous HMMs with Explicit State Duration for Unconstrained Arabic Word Modeling and Recognition. *Pattern Recognition Letters* 29 (12). Elsevier: 1742–52.

BenZeghiba, Mohamed Faouzi, Jerome Louradour, and Christopher Kermorvant. 2015. Hybrid word/Part-of-Arabic-Word Language Models for Arabic Text Document Recognition. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 671–75. IEEE. doi:10.1109/ICDAR.2015.7333846.

Bianne-Bernard, Anne Laure, Fares Menasri, Rami Al-Hajj Mohamad, Chafic Mokbel, Christopher Kermorvant, and Laurence Likforman-Sulem. 2011. Dynamic and Contextual Information in HMM Modeling for Handwritten Word Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (10): 2066–80. doi:10.1109/TPAMI.2011.22.

Bluche, Théodore, Jerome Louradour, Maxime Knibbe, Bastien Moysset, Mohamed Faouzi Benzeghiba, and Christopher Kermorvant. 2014. The A2iA Arabic Handwritten Text Recognition System at the Open HaRT2013 Evaluation. In *Proceedings of the 11th IAPR International Workshop on Document Analysis Systems (DAS 2014)*, 161–65. IEEE. doi:10.1109/DAS.2014.40.

Bluche, Théodore, Hermann Ney, and Christopher Kermorvant. 2014. A Comparison of Sequence-Trained Deep Neural Networks and Recurrent Neural Networks Optical Modeling for Handwriting Recognition. In *Proceedings of Second International Conference on Statistical Language and Speech Processing, SLSP2014*, edited by Laurent Besacier, Adrian-Horia Dediu, and Carlos Mart'in-Vide, 199–210. Grenoble: Springer International Publishing. doi:10.1007/978-3-319-11397-5_15.

Bose, Chinmoy B, and Shyh-Shiaw Kuo. 1994. Connected and Degraded Text Recognition Using Hidden Markov Model. *Pattern Recognition* 27 (10): 1345–63. doi:10.1016/0031-3203(94)90069-8.

Caesar, T, J M Gloger, and E Mandler. 1993. Preprocessing and Feature Extraction for a Handwriting Recognition System. In *Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR 1993)*, 408–11. doi:10.1109/ICDAR.1993.395706.

Cao, Huaigu, Prem Natarajan, Xujun Peng, Krishna Subramanian, David Belanger, and Nan Li. 2014. Progress in the Raytheon BBN Arabic Offline Handwriting Recognition System. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, 555–60. IEEE. doi:10.1109/ICFHR.2014.99.

Chammas, E, Chafic Mokbel, and Laurence Likforman-Sulem. 2015. Arabic Handwritten Document Preprocessing and Recognition. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 451–55. doi:10.1109/ICDAR.2015.7333802.

Cheriet, Mohamed, Yousef Al-Ohali, N Ayat, and Ching Y. Suen. 2007. Arabic Cheque Processing System: Issues and Future Trends. In *Digital Document Processing*, edited by Bidyut B Chaudhuri, 213–34. Advances in Pattern Recognition. London: Springer London. doi:10.1007/978-1-84628-726-8.

Dehghan, M., Karim Faez, M. Ahmadi, and M. Shridhar. 2001. Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM. *Pattern Recognition* 34 (5): 1057–65. doi:10.1016/S0031-3203(00)00051-0.

Dietterich, Thomas G. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation* 10 (7). MIT Press 238 Main St., Suite 500, Cambridge, MA 02142-1046 USA journals-info@mit.edu: 1895–1923. doi:10.1162/089976698300017197.

Dreuw, Philippe, Georg Heigold, and Hermann Ney. 2009. Confidence-Based Discriminative Training for Model Adaptation in Offline Arabic Handwriting Recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 596–600. IEEE. doi:10.1109/ICDAR.2009.116.

Dreuw, Philippe, S Jonas, and Hermann Ney. 2008. White-Space Models for Offline Arabic Handwriting Recognition. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)*, 1–4.

Dreuw, Philippe, David Rybach, Christian Gollan, and Hermann Ney. 2009. Writer Adaptive Training and Writing Variant Model Refinement for Offline Arabic Handwriting Recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 21–25. IEEE. doi:10.1109/ICDAR.2009.9.

Dreuw, Philippe, David Rybach, Georg Heigold, and Hermann Ney. 2012. RWTH OCR: A Large Vocabulary Optical Character Recognition System for Arabic Scripts. In *Guide to OCR for Arabic Scripts SE - 9*, edited by Volker Märgner and Haikal El Abed, 215–54. Springer London. doi:10.1007/978-1-4471-4072-6_9.

El-Desouky, A.I., M.M. Salem, A.O. Abd El-Gwad, and H. Arafat. 1991. A Handwritten Arabic Character Recognition Technique for Machine Reader. In *Proceedings of the Third International Conference on Software Engineering for Real Time Systems*, 212–16. Cirencester: IET.

El-Hajj, Ramy, Laurence Likforman-Sulem, and Chafic Mokbel. 2005. Arabic Handwriting Recognition Using Baseline Dependant Features and Hidden Markov Modeling. In *Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR 2005)*, 893–97.

El-Hajj, Ramy, Chafic Mokbel, and Laurence Likforman-Sulem. 2008. Recognition of Arabic Handwritten Words Using Contextual Character Models. *Document Recognition and Retrieval XV-SPIE*.

El Abed, Haikal, and Volker Märgner. 2010a. ICDAR 2009-Arabic Handwriting Recognition Competition. *International Journal on Document Analysis and Recognition (IJDAR)* 14 (1): 3–13. doi:10.1007/s10032-010-0117-5.

———. 2010b. A Framework for the Combination of Different Arabic Handwritten Word Recognition Systems. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR 2010)*, 1904–7. IEEE. doi:10.1109/ICPR.2010.469.

Elarian, Yousef S., Radwan Abdel-Aal, Irfan Ahmad, Mohammad Tanvir Parvez, and Abdelmalek Zidouri. 2014. Handwriting Synthesis: Classifications and Techniques. *International Journal on Document Analysis and Recognition (IJDAR)* 17 (4): 455–69. doi:10.1007/s10032-014-0231-x.

Elarian, Yousef S., Irfan Ahmad, Sameh M. Awaida, Wasfi G. Al-Khatib, and Abdelmalek Zidouri. 2015a. Arabic Ligatures: Analysis and Application in Text Recognition. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 896–900. IEEE.

———. 2015b. An Arabic Handwriting Synthesis System. *Pattern Recognition* 48 (3): 849–61. doi:10.1016/j.patcog.2014.09.013.

Farah, Nadir, Labiba Souici-Meslati, and Mokhtar Sellami. 2006. Classifiers Combination and Syntax Analysis for Arabic Literal Amount Recognition. *Engineering Applications of Artificial Intelligence* 19 (1): 29–39. doi:10.1016/j.engappai.2005.05.005.

Fink, Gernot A. 2014. *Markov Models for Pattern Recognition*. 2nded. London: Springer London. doi:10.1007/978-1-4471-6308-4.

Fink, Gernot A., and Thomas Plötz. 2007. On the Use of Context-Dependent Modeling Units for HMM-Based Offline Handwriting Recognition. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, 2:729–33. IEEE. doi:10.1109/ICDAR.2007.4377011.

Frinken, V, A Fischer, Horst Bunke, and A Foornes. 2011. Co-Training for Handwritten Word Recognition. In *Proceedings of the Eleventh International Conference on Document Analysis and Recognition (ICDAR 2011)*, 314–18. doi:10.1109/ICDAR.2011.71.

Gales, M.J.F., and P.C. Woodland. 1996. Mean and Variance Adaptation within the MLLR Framework. *Computer Speech & Language* 10 (4): 249–64. doi:10.1006/csla.1996.0013.

Giménez, Adrià, Ihab Khoury, Jesús Andrés-Ferrer, and Alfons Juan. 2014. Handwriting Word Recognition Using Windowed Bernoulli HMMs. *Pattern Recognition Letters* 35 (January): 149–56. doi:10.1016/j.patrec.2012.09.002.

Giménez, Adrià, Ihab Khoury, and Alfons Juan. 2010. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR 2010)*, 533–38. IEEE. doi:10.1109/ICFHR.2010.88.

Goraine, H, M Usher, and S Al-Emami. 1992. Off-Line Arabic Character Recognition. *Computer* 25 (7): 71–74. doi:10.1109/2.144444.

Graves, Alex. 2012. Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Guide to OCR for Arabic Scripts*, edited by Volker Märgner and Haikal El Abed, 297–313. London: Springer London. doi:10.1007/978-1-4471-4072-6_12.

Graves, Alex, and J Schmidhuber. 2009. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, 21:545–52.

Hamdani, Mahdi, Patrick Doetsch, Michal Kozielski, Amr El-Desoky Mousa, and Hermann Ney. 2014. The RWTH Large Vocabulary Arabic Handwriting Recognition System. In *Proceedings of the 11th IAPR International Workshop on Document Analysis Systems (DAS 2014)*, 111–15. IEEE. doi:10.1109/DAS.2014.61.

Hamdani, Mahdi, Patrick Doetsch, and Hermann Ney. 2014. Improvement of Context Dependent Modeling for Arabic Handwriting Recognition. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, 494–99. IEEE. doi:10.1109/ICFHR.2014.89.

Hamdani, Mahdi, Haikal El Abed, Monji Kherallah, and Adel Mohamed Alimi. 2009. Combining Multiple HMMs Using on-Line and off-Line Features for off-Line Arabic Handwriting Recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 201–5. Ieee. doi:10.1109/ICDAR.2009.40.

Hamdani, Mahdi, Tarek M. Hamdani, Adel Mohamed Alimi, Haikal El Abed, and Volker Märgner. 2011. Unsupervised Selection of HMMs Architectures for Handwritten Text/word Recognition. In *Proceedings of the 5th International Symposium on Computational Intelligence and Intelligent Informatics (ISCIII 2011)*, 19–24. IEEE. doi:10.1109/ISCIII.2011.6069735.

Hamdani, Mahdi, Amr El-Desoky Mousa, and Hermann Ney. 2013. Open Vocabulary Arabic Handwriting Recognition Using Morphological Decomposition. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, 280–84. IEEE. doi:10.1109/ICDAR.2013.63.

Hu, J, S Gek Lim, and M K Brown. 2000. Writer Independent on-Line Handwriting Recognition Using an HMM Approach. *Pattern Recognition* 33 (1). Elsevier: 133–47.

Huang, Xuedong, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice hall PTR.

Intellaren. 2016. A Study of Arabic Letter Frequency Analysis. Accessed March 9. http://www.intellaren.com/articles/en/a-study-of-arabic-letter-frequency-analysis.

Jiang, Zhiwei, Xiaoqing Ding, Liangrui Peng, and Changsong Liu. 2012. Analyzing the Information Entropy of States to Optimize the Number of States in an HMM-Based off-Line Handwritten Arabic Word Recognizer. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR 2012)*, 697–700.

———. 2015. Exploring More Representative States of Hidden Markov Model in Optical Character Recognition: A Clustering-Based Model Pre-Training Approach. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)* 29 (3): 1550014. doi:10.1142/S0218001415500147.

Kessentini, Y, Thierry Paquet, and A M Ben Hamadou. 2010. Off-Line Handwritten Word Recognition Using Multi-Stream Hidden Markov Models. *Pattern Recognition Letters* 31 (1). Elsevier: 60–70.

Kharma, N., M. Ahmed, and R. Ward. 1999. A New Comprehensive Database of Handwritten Arabic Words, Numbers, and Signatures Used for OCR Testing. In *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering. Engineering Solutions for the Next Millennium*, 2:766–68. IEEE. doi:10.1109/CCECE.1999.808042.

Khorsheed, Mohammad S. 2002. Off-Line Arabic Character Recognition--a Review. *Pattern Analysis & Applications* 5 (1). Springer: 31–45.

———. 2003. Recognising Handwritten Arabic Manuscripts Using a Single Hidden

Markov Model. *Pattern Recognition Letters* 24 (14). Elsevier: 2235–42.

———. 2007. Offline Recognition of Omnifont Arabic Text Using the HMM ToolKit (HTK). *Pattern Recognition Letters* 28 (12): 1563–71. doi:10.1016/j.patrec.2007.03.014.

Khoury, Ihab, Adrià Giménez, Alfons Juan, and Jesús Andrés-Ferrer. 2013. Arabic Printed Word Recognition Using Windowed Bernoulli HMMs. In *Image Analysis and Processing – ICIAP 2013 SE - 34*, edited by Alfredo Petrosino, 8156:330–39. Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-41181-6_34.

Kosmala, A, J Rottland, and G Rigoll. 1997. Improved on-Line Handwriting Recognition Using Context Dependent Hidden Markov Models. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR 1997)*, 2:641–44. IEEE Comput. Soc. doi:10.1109/ICDAR.1997.620584.

Kozielski, Michal, Malte Nuhn, Patrick Doetsch, and Hermann Ney. 2014. Towards Unsupervised Learning for Handwriting Recognition. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, 549–54. doi:10.1109/ICFHR.2014.98.

Leggetter, C.J., and P.C. Woodland. 1995. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech & Language* 9 (2): 171–85. doi:10.1006/csla.1995.0010.

Letter Frequency. 2016. Accessed November 13. https://en.wikipedia.org/wiki/Letter_frequency.

Lewis, M. Paul, Gary F. Simons, and Charles D. Fennig (eds.). n.d. Ethnologue: Languages of the World, Dallas, Texas: SIL International. *Online Version: Http://www.ethnologue.com [Accessed:03-Dec-2015]*.

Likforman-Sulem, Laurence, Abderrazak Zahour, and Bruno Taconet. 2007. Text Line Segmentation of Historical Documents: A Survey. *International Journal of Document Analysis and Recognition (IJDAR)* 9 (2). Springer-Verlag: 123–38. doi:10.1007/s10032-006-0023-z.

Lorigo, Liana M, and Venu Govindaraju. 2006. Offline Arabic Handwriting Recognition: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (5): 712–24. doi:10.1109/TPAMI.2006.102.

Luettin, J, G Potamianos, and C Neti. 2001. Asynchronous Stream Modeling for Large Vocabulary Audio-Visual Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, 1:169–72 vol.1. doi:10.1109/ICASSP.2001.940794.

Luqman, Hamzah, Sabri A. Mahmoud, and Sameh M. Awaida. 2014. KAFD Arabic

Font Database. *Pattern Recognition* 47 (6): 2231–40. doi:10.1016/j.patcog.2013.12.012.

Mahmoud, Sabri A. 2008. Recognition of Writer-Independent off-Line Handwritten Arabic (Indian) Numerals Using Hidden Markov Models. *Signal Processing* 88 (4). Elsevier: 844–57.

Mahmoud, Sabri A., Irfan Ahmad, Wasfi G. Al-Khatib, Mohammed Alshayeb, Mohammad Tanvir Parvez, Volker Märgner, and Gernot A. Fink. 2014. KHATT: An Open Arabic Offline Handwritten Text Database. *Pattern Recognition* 47 (3): 1096–1112. doi:10.1016/j.patcog.2013.08.009.

Mahmoud, Sabri A., Irfan Ahmad, Mohammed Alshayeb, and Wasfi G. Al-Khatib. 2011. A Database for Offline Arabic Handwritten Text Recognition. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, edited by Mohamed Kamel and Aurélio C Campilho, 6754 LNCS:397–406. Burnaby, BC, Canada: Springer. doi:10.1007/978-3-642-21596-4_40.

Mahmoud, Sabri A., Irfan Ahmad, Mohammed Alshayeb, Wasfi G. Al-Khatib, Mohammad Tanvir Parvez, Gernot A. Fink, Volker Märgner, and Haikal EL Abed. 2012. KHATT: Arabic Offline Handwritten Text Database. In *Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*, 447–52. IEEE.

Mahmoud, Sabri A., and Wasfi G. Al-Khatib. 2010. Recognition of Arabic (Indian) Bank Check Digits Using Log-Gabor Filters. *Applied Intelligence* 35 (3). Springer Netherlands: 445–56. doi:10.1007/s10489-010-0235-2.

Manabe, H., and Z. Zhang. 2004. Multi-Stream HMM for EMG-Based Speech Recognition. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 4:4389–92. IEEE. doi:10.1109/IEMBS.2004.1404221.

Mantas, J. 1986. An Overview of Character Recognition Methodologies. *Pattern Recognition* 19 (6): 425–30. doi:10.1016/0031-3203(86)90040-3.

Märgner, Volker, and Haikal El Abed. 2007. Arabic Handwriting Recognition Competition. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, 2:1274–78. IEEE. doi:10.1109/ICDAR.2007.4377120.

———. 2010. ICFHR 2010 - Arabic Handwriting Recognition Competition. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR 2010)*, 709–14. IEEE. doi:10.1109/ICFHR.2010.115.

———. 2011. ICDAR 2011 - Arabic Handwriting Recognition Competition. In

Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011), 1444–48. IEEE. doi:10.1109/ICDAR.2011.287.

Märgner, Volker, Mario Pechwitz, and Haikal El Abed. 2005. ICDAR 2005 Arabic Handwriting Recognition Competition. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR 2005)*, 70–74 Vol. 1. IEEE. doi:10.1109/ICDAR.2005.52.

Marti, U. V., and Horst Bunke. 2003. The IAM-Database: An English Sentence Database for Offline Handwriting Recognition. *International Journal on Document Analysis and Recognition (IJDAR)* 5 (1): 39–46. doi:10.1007/s100320200071.

Menasri, F, N Vincent, E Augustin, and Mohamed Cheriet. 2007. Shape-Based Alphabet for off-Line Arabic Handwriting Recognition. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:969–73.

Miled, H, and N E B Amara. 2001. Planar Markov Modeling for Arabic Writing Recognition: Advancement State. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR 2001)*, 69–73.

Miyao, Hidetoshi, and Maruyama Minoru. 2006. Virtual Example Synthesis Based on PCA for Off-Line Handwritten Character Recognition. In *Document Analysis Systems VII*, edited by Horst Bunke and A. Lawrence Spitz, 3872:96–105. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/11669487.

Morillot, Olivier, Cristina Oprean, Laurence Likforman-Sulem, Chafic Mokbel, Edgar Chammas, and Emmanuèle Grosicki. 2013. The UOB-Telecom ParisTech Arabic Handwriting Recognition and Translation Systems for the OpenHart 2013 Competition. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, NIST. Washington DC, United States. https://hal.archives-ouvertes.fr/hal-00948985.

Moysset, Bastien, Théodore Bluche, Maxime Knibbe, Mohamed Faouzi Benzeghiba, Ronaldo Messina, Jerome Louradour, and Christopher Kermorvant. 2014. The A2iA Multi-Lingual Text Recognition System at the Second Maurdor Evaluation. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, 297–302. IEEE. doi:10.1109/ICFHR.2014.57.

Mozaffari, Saeed, and H Soltanizadeh. 2009. ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 1413–17. doi:10.1109/ICDAR.2009.283.

Nakai, M, N Akira, H Shimodaira, and S Sagayama. 2001. Substroke Approach to HMM-Based on-Line Kanji Handwriting Recognition. In *Proceedings of Sixth International Conference on Document Analysis and Recognition (ICDAR 2001)*, 491–95. IEEE Comput. Soc. doi:10.1109/ICDAR.2001.953838.

Nakai, M, H Shimodaira, and S Sagayama. 2003. Generation of Hierarchical Dictionary for Stroke-Order Free Kanji Handwriting Recognition Based on Substroke HMM. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, 1:514–18. IEEE Comput. Soc. doi:10.1109/ICDAR.2003.1227718.

Natarajan, Pradeep, David Belanger, Rohit Prasad, Matin Kamali, Krishna Subramanian, and Prem Natarajan. 2011. Baseline Dependent Percentile Features for Offline Arabic Handwriting Recognition. In *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011)*, 329–33. IEEE. doi:10.1109/ICDAR.2011.74.

Natarajan, Pradeep, Krishna Subramanian, Anurag Bhardwaj, and Rohit Prasad. 2009. Stochastic Segment Modeling for Offline Handwriting Recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 971–75.

Natarajan, Premkumar, Zhidong Lu, Richard Schwartz, Issam Bazzi, and John Makhoul. 2001. Multilingual Machine Printed OCR. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)* 15 (1). World Scientific Publishing Company: 43–63. doi:10.1142/S0218001401000745.

Natarajan, Premkumar, Rohit Prasad, Huaigu Cao, Krishna Subramanian, Shirin Saleem, David Belanger, Shiv Vitaladevuni, Matin Kamali, and Ehry MacRostie. 2012. Arabic Text Recognition Using a Script-Independent Methodology: A Unified HMM-Based Approach for Machine-Printed and Handwritten Text. In *Guide to OCR for Arabic Scripts*, edited by Volker Märgner and Haikal El Abed, 485–505. London: Springer London. doi:10.1007/978-1-4471-4072-6_20.

NIST. 2016. OpenHaRT 2013 Information Page. Accessed February 25. http://www.nist.gov/itl/iad/mig/hart2013.cfm.

Oparin, Ilya, Juliette Kahn, and Olivier Galibert. 2014. First Maurdor 2013 Evaluation Campaign in Scanned Document Image Processing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 5090–94. IEEE. doi:10.1109/ICASSP.2014.6854572.

Parvez, Mohammad Tanvir, and Sabri A. Mahmoud. 2013a. Arabic Handwriting Recognition Using Structural and Syntactic Pattern Attributes. *Pattern Recognition* 46 (1): 141–54. doi:10.1016/j.patcog.2012.07.012.

———. 2013b. Offline Arabic Handwritten Text Recognition: A Survey. *ACM*

Comput. Surv. 45 (2). New York, NY, USA: ACM: 23:1--23:35. doi:10.1145/2431211.2431222.

Pechwitz, Mario, Haikal El Abed, and Volker Märgner. 2012. Handwritten Arabic Word Recognition Using the IFN/ENIT-Database. In *Guide to OCR for Arabic Scripts*, edited by Volker Märgner and Haikal El Abed, 297–313. Springer London. doi:10.1007/978-1-4471-4072-6{_}8.

Pechwitz, Mario, Samia Snoussi Maddouri, Volker Märgner, Noureddine Ellouze, and Hamid Amiri. 2002. IFN/ENIT - Database of Handwritten Arabic Words. In *7th Colloque International Francophone Sur l'Ecrit et Le Document , CIFED 2002*, 129--136. Hammamet, Tunis.

Pechwitz, Mario, and Volker Märgner. 2003. HMM Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT-Database. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, 890–94.

Pechwitz, Mario, Volker Märgner, and Haikal El Abed. 2006. Comparison of Two Different Feature Sets for Offline Recognition of Handwritten Arabic Words. *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006)*. Suvisoft. https://hal.archives-ouvertes.fr/inria-00112643/.

Plötz, Thomas, and Gernot A. Fink. 2009. Markov Models for Offline Handwriting Recognition: A Survey. *International Journal on Document Analysis and Recognition (IJDAR)* 12 (4): 269–98. doi:10.1007/s10032-009-0098-4.

———. 2011. *Markov Model Based Handwriting Recognition*. Springer.

Prasad, Rohit, Shirin Saleem, Matin Kamali, Ralf Meermeier, and Prem Natarajan. 2008. Improvements in Hidden Markov Model Based Arabic OCR. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)*, 1–4. IEEE. doi:10.1109/ICPR.2008.4761446.

Rabiner, L.R. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* 77 (2). IEEE: 257–86. doi:10.1109/5.18626.

Rabiner, L.R., and B. Juang. 1986. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine* 3 (1): 4–16. doi:10.1109/MASSP.1986.1165342.

Richarz, Jan, Szilard Vajda, Rene Grzeszick, and Gernot A. Fink. 2014. Semi-Supervised Learning for Character Recognition in Historical Archive Documents. *Pattern Recognition* 47 (3): 1011–20. doi:10.1016/j.patcog.2013.07.013.

Rothacker, Leonard, Szilard Vajda, and Gernot A. Fink. 2012. Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script. In *Proceedings of the 13th International Conference on Frontiers in Handwriting*

Recognition (ICFHR 2012), 149–54. doi:10.1109/ICFHR.2012.185.

Safabakhsh, R, and P Adibi. 2005. Nastaaligh Handwritten Word Recognition Using a Continuous-Density Variable-Duration HMM. *Arabian Journal for Science and Engineering* 30 (1). KFUPM Dhahran: 95–118.

Saleem, Shirin, Huaigu Cao, Krishna Subramanian, Matin Kamali, Rohit Prasad, and Prem Natarajan. 2009. Improvements in BBN's HMM-Based Offline Arabic Handwriting Recognition System. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 773–77. IEEE. doi:10.1109/ICDAR.2009.282.

Schambach, M P, J Rottland, and T Alary. 2008. How to Convert a Latin Handwriting Recognition System to Arabic. In *Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008)*, 265–70.

Shih, Hsuan-Huei, S S Narayanan, and C.-C.J. Kuo. 2003. A Statistical Multidimensional Humming Transcription Using Phone Level Hidden Markov Models for Query by Humming Systems. In *Proceedings of the International Conference on Multimedia and Expo, ICME '03.*, 1:I-61-4 vol.1. doi:10.1109/ICME.2003.1220854.

Slimane, Fouad, Rolf Ingold, Slim Kanoun, Adel Mohamed Alimi, and Jean Hennebert. 2009. A New Arabic Printed Text Image Database and Evaluation Protocols. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR 2009)*, 946–50. IEEE. doi:10.1109/ICDAR.2009.155.

———. 2010. Impact of Character Models Choice on Arabic Text Recognition Performance. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR 2010)*, 670–75. IEEE. doi:10.1109/ICFHR.2010.110.

Slimane, Fouad, Slim Kanoun, Haikal El Abed, Adel Mohamed Alimi, Rolf Ingold, and Jean Hennebert. 2011. ICDAR 2011 - Arabic Recognition Competition: Multi-Font Multi-Size Digitally Represented Text. In *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011)*, 1449–53. IEEE. doi:10.1109/ICDAR.2011.288.

Slimane, Fouad, Slim Kanoun, Haikal El Abed, Adel Mohamed Alimi, Rolf Ingold, and Jean Hennebert. 2013. ICDAR2013 Competition on Multi-Font and Multi-Size Digitally Represented Arabic Text. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, 1433–37. IEEE. doi:10.1109/ICDAR.2013.289.

Slimane, Fouad, O. Zayene, Slim Kanoun, Adel Mohamed Alimi, Jean Hennebert, and Rolf Ingold. 2012. New Features for Complex Arabic Fonts in Cascading Recognition System. In *Proceedings of the of 21st International Conference on*

Pattern Recognition (ICPR 2012), 738–41.

Stahlberg, Felix, and Stephan Vogel. 2015. The QCRI Recognition System for Handwritten Arabic. In *Proceedings of the 18th International Conference on Image Analysis and Processing (ICIAP 2015)*, edited by Vittorio Murino and Enrico Puppo, 276–86. Genoa, Italy: Springer International Publishing. doi:10.1007/978-3-319-23234-8_26.

Tokuno, J, N Inami, S Matsuda, M Nakai, H Shimodaira, and S Sagayama. 2002. Context-Dependent Substroke Model for HMM-Based on-Line Handwriting Recognition. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR 2002)*, 78–83. IEEE Comput. Soc. doi:10.1109/IWFHR.2002.1030888.

Tong, Audrey, Mark Przybocki, Volker Märgner, and Haikal El Abed. 2014. NIST 2013 Open Handwriting Recognition and Translation (Open HaRT'13) Evaluation. In *Proceedings of the 11th IAPR International Workshop on Document Analysis Systems (DAS 2014)*, 81–85. IEEE. doi:10.1109/DAS.2014.43.

UNESCO. 2015. World Arabic Language Day. Accessed December 3. http://www.unesco.org/new/en/unesco/events/prizes-and-celebrations/celebrations/international-days/world-arabic-language-day/.

Varga, Tamás, and Horst Bunke. 2008. *Machine Learning in Document Analysis and Recognition*. Edited by Simone Marinai and Hiromichi Fujisawa. Vol. 90. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-76280-5.

Wienecke, M, Gernot A. Fink, and G Sagerer. 2005. Toward Automatic Video-Based Whiteboard Reading. *International Journal on Document Analysis and Recognition (IJDAR)* 7 (2). Springer: 188–200.

Young, S.J., G Evermann, T Hain, D Kershaw, G Moore, J J Odell, D Ollason, D Povey, V Valtchev, and P.C. Woodland. 2002. *The HTK Book (for HTK Version 3.2. 1)*. Cambridge University Engineering Department.

Young, S.J., and P.C. Woodland. 1994. State Clustering in Hidden Markov Model-Based Continuous Speech Recognition. *Computer Speech & Language* 8 (4): 369–83. doi:10.1006/csla.1994.1019.

Zimmermann, M, and Horst Bunke. 2002. Hidden Markov Model Length Optimization for Handwriting Recognition Systems. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR 2002)*, 369–74. doi:10.1109IWFHR.2002.1030938.